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

LNCS 6644

The Semantic Web: Research and Applications

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

2 Part II

Springer

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 II

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 978-3-642-21063-1 e-ISBN 978-3-642-21064-8
DOI 10.1007/978-3-642-21064-8

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

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

Program Committee - Mobile Web Track

Track Chairs

Ora Lassila and Alessandra Toninelli

Members

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

Program Committee - Natural Language Processing Track

Track Chairs

Philipp Cimiano and Michael Witbrock

Members

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

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

Gold Sponsors

Silver Sponsors

Best Paper Award Sponsors

Video Recording Sponsors

Table of Contents – Part II

Demo Track

PhD Symposium

Table of Contents – Part I

Semantics and Optimization of the SPARQL 1.1 Federation Extension

Carlos Buil-Aranda[1], Marcelo Arenas[2], and Oscar Corcho[1]

[1] Ontology Engineering Group, Facultad de Informática, UPM, Spain
[2] Department of Computer Science, PUC Chile

Abstract. The W3C SPARQL working group is defining the new SPARQL 1.1 query language. The current working draft of SPARQL 1.1 focuses mainly on the description of the language. In this paper, we provide a formalization of the syntax and semantics of the SPARQL 1.1 federation extension, an important fragment of the language that has not yet received much attention. Besides, we propose optimization techniques for this fragment, provide an implementation of the fragment including these techniques, and carry out a series of experiments that show that our optimization procedures significantly speed up the query evaluation process.

1 Introduction

The recent years have witnessed a constant growth in the amount of RDF data available, exposed by means of Linked Data-enabled URLs and SPARQL endpoints. Several non-exhaustive, and sometimes out-of-date, lists of SPARQL endpoints or data catalogs are available in different formats (from wiki-based HTML pages to SPARQL endpoints using data catalog description vocabularies). Besides, most of these datasets are interlinked, what allows navigating through them and facilitates building complex queries combining data from heterogeneous datasets.

These SPARQL endpoints accept queries written in SPARQL and adhere to the SPARQL protocol, as defined by the W3C recommendation. However, the current SPARQL recommendation has an important limitation in defining and executing queries that span across distributed datasets, since it only considers the possibility of executing these queries in isolated SPARQL endpoints. Hence users willing to federate queries across a number of SPARQL endpoints have been forced to create ad-hoc extensions of the query language or to include additional information about data sources in the configuration of their SPARQL endpoint servers [14,15]. This has led to the inclusion of query federation extensions in the current SPARQL 1.1 working draft [12] (together with other extensions that are out of the scope of this paper), which are studied in detail in order to generate a new W3C recommendation in the coming months.

The federation extension of SPARQL 1.1 includes two new operators in the query language: SERVICE and BINDINGS. The former allows specifying, inside a SPARQL query, the SPARQL query service in which a portion of the query will be executed. This query service may be known at the time of building the query, and hence the SERVICE operator will already specify the IRI of the SPARQL endpoint where it will be executed; or may be retrieved at query execution time after executing an initial SPARQL query fragment in one of the aforementioned RDF-enabled data catalogs, so that potential SPARQL endpoints that can answer the rest of the query can be obtained and used. The latter (BINDINGS) allows transferring results that are used to constrain a query, and which will normally come from previous executions of other queries or from constraints specified in user interfaces that then transform these into SPARQL queries.

Till now, most of the work done on federation extensions in the context of the W3C working group has been focused on the description of the language grammar. In this paper we complement this work with the formalization of the syntax and semantics of these federation extensions of SPARQL 1.1, and with the definition of the constraints that have to be considered in their use (which is currently not too restricted) in order to be able to provide pragmatic implementations of query evaluators. As an extreme example of bad performance, we may imagine a query that uses the SERVICE operator with a free variable to specify the SPARQL endpoint where the rest of the query has to be evaluated. We may imagine that a naïve implementation may need to go through all existing SPARQL endpoints on the Web evaluating that query fragment before providing a result, something that can be considered infeasible in practical terms. For our purpose, we define the notions of service-boundedness and service-safeness, which ensure that the SERVICE operator can be safely evaluated.

Besides, we implement the optimizations proposed in [11], using the notion of well-designed patterns, which prove to be effective in the optimization of queries that contain the OPTIONAL operator, the most costly operator in SPARQL [11,17]. This has also important implications in the number of tuples being transferred and joined in federated queries, and hence our implementation benefits from this.

As a result of our work, we have not only formalized these notions, but we have also implemented a system that supports the current SPARQL 1.1 federation extensions and makes use of these optimizations. This system, SPARQL-DQP (which stands for SPARQL Distributed Query Processing), is built on top of the OGSA-DAI and OGSA-DQP infrastructure [3,10], what provides additional robustness to deal with large amounts of data in distributed settings, supporting for example an indirect access mode that is normally used in the development of data-intensive workflows. We have evaluated our system using a small benchmark of real SPARQL 1.1 queries from the bioinformatics domain, and compared it with other similar systems, in some cases adapting the queries to their own ad-hoc SPARQL extensions, so that the benefits of our implementation can be illustrated.

With this work, we aim at advancing to the current state of the art hoping to include it in the next versions of the SPARQL working drafts, and providing SPARQL-DQP as one of the reference implementations of this part of the recommendation. We also hope that the initial benchmark that we have defined can be extended and stabilized in order to provide a good evaluation framework, complementing existing benchmarks.

Organization of the paper. In Section 2, we describe the syntax and semantics of the SPARQL 1.1 federation extension. In Section 3, we introduce the notions of service-safeness, which ensures that the SERVICE operator can be safely evaluated. In Section 4, we present some optimization techniques for the evaluation of the SPARQL 1.1 federation extension. Finally, in Section 5, we present our implementation as well as an experimental evaluation of it.

2 Syntax and Semantics of the SPARQL 1.1 Federation Extension

In this section, we give an algebraic formalization of the SPARQL 1.1 federation extension over simple RDF, that is, RDF without RDFS vocabulary and literal rules. Our starting point is the existing formalization of SPARQL described in [11], to which we add the operators SERVICE and BINDINGS proposed in [12].

We introduce first the necessary notions about RDF (taken mainly from [11]). Assume there are pairwise disjoint infinite sets I, B, and L (IRIs [6], Blank nodes, and Literals, respectively). Then a triple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an *RDF triple*. In this tuple, s is the *subject*, p the *predicate* and o the *object*. An *RDF graph* is a set of RDF triples. Moreover, assume the existence of an infinite set V of variables disjoint from the above sets, and leave UNBOUND to be a reserve word that does not belong to any of the sets mentioned previously.

2.1 Syntax of the Federation Extension

The official syntax of SPARQL [13] considers operators `OPTIONAL`, `UNION`, `FILTER`, `SELECT` and concatenation via a point symbol (`.`), to construct graph pattern expressions. Operators `SERVICE` and `BINDINGS` are introduced in the SPARQL 1.1 federation extension, the former for allowing users to direct a portion of a query to a particular SPARQL endpoint, and the latter for transferring results that are used to constrain a query. The syntax of the language also considers `{ }` to group patterns, and some implicit rules of precedence and association. In order to avoid ambiguities in the parsing, we follow the approach proposed in [11], and we first present the syntax of SPARQL graph patterns in a more traditional algebraic formalism, using operators AND (`.`), UNION (`UNION`), OPT (`OPTIONAL`), FILTER (`FILTER`) and SERVICE (`SERVICE`), then we introduce the syntax of BINDINGS queries, which use the BINDINGS operator (`BINDINGS`), and we conclude by defining the syntax of SELECT queries, which use the SELECT operator (`SELECT`). More precisely, a SPARQL graph pattern expression is defined recursively as follows:

(1) A tuple from $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a graph pattern (a triple pattern).
(2) If P_1 and P_2 are graph patterns, then expressions $(P_1 \text{ AND } P_2)$, $(P_1 \text{ OPT } P_2)$, and $(P_1 \text{ UNION } P_2)$ are graph patterns.
(3) If P is a graph pattern and R is a *SPARQL built-in condition*, then the expression $(P \text{ FILTER } R)$ is a graph pattern.
(4) If P is a graph pattern and $a \in (I \cup V)$, then $(\text{SERVICE } a\ P)$ is a graph pattern.

Moreover, a SPARQL BINDINGS query is defined as follows:

(5) If P is a graph pattern, S is a nonempty list of pairwise distinct variables and $\{A_1, \ldots, A_n\}$ is a nonempty set of lists such that for every $i \in \{1, \ldots, n\}$, it holds that A_i and S have the same length and each element in A_i belongs to $(I \cup L \cup \{\text{UNBOUND}\})$, then $(P\ \text{ BINDINGS }\ S\ \{A_1, \ldots, A_n\})$ is a BINDINGS query.

Finally, assuming that P is either a graph pattern or a BINDINGS query, let $\text{var}(P)$ be the set of variables mentioned in P. Then a SPARQL SELECT query is defined as:

(6) If P is either a graph pattern or a BINDINGS query, and W is a set of variables such that $W \subseteq \text{var}(P)$, then $(\text{SELECT } W\ P)$ is a SELECT query.

It is important to notice that the rules (1)–(3) above were introduced in [11], while we formalize in the rules (4)–(6) the federation extension of SPARQL proposed in [12].

In the previous definition, we use the notion of built-in condition for the filter operator. A SPARQL built-in condition is constructed using elements of the set $(I \cup L \cup V)$ and constants, logical connectives ($\neg$, $\wedge$, $\vee$), inequality symbols ($<$, $\leq$, $\geq$, $>$), the equality symbol ($=$), unary predicates like bound, isBlank, and isIRI, plus other features (see [13] for a complete list). Due to the lack of space, we restrict in this paper to the fragment of SPARQL where the built-in condition is a Boolean combination of terms constructed by using $=$ and bound, that is: (1) if $?X, ?Y \in V$ and $c \in (I \cup L)$, then $\text{bound}(?X)$, $?X = c$ and $?X = ?Y$ are built-in conditions, and (2) if R_1 and R_2 are built-in conditions, then $(\neg R_1)$, $(R_1 \vee R_2)$ and $(R_1 \wedge R_2)$ are built-in conditions. It should be noticed that the results of the paper can be easily extended to the other built-in predicates in SPARQL.

Let P be either a graph pattern or a BINDINGS query or a SELECT query. In the rest of the paper, we use $\text{var}(P)$ to denote the set of variables occurring in P. Similarly, for a built-in condition R, we use $\text{var}(R)$ to denote the set of variables occurring in R.

2.2 Semantics of the Federation Extension

To define the semantics of SPARQL queries, we need to introduce some extra terminology from [11]. A mapping μ from V to $(I \cup B \cup L)$ is a partial function $\mu : V \to (I \cup B \cup L)$. Abusing notation, for a triple pattern t we denote by $\mu(t)$ the triple obtained by replacing the variables in t according to μ. The domain of μ, denoted by $\text{dom}(\mu)$, is the subset of V where μ is defined. Two mappings μ_1 and μ_2 are compatible when for all $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, it is the case that $\mu_1(?X) = \mu_2(?X)$, i.e. when $\mu_1 \cup \mu_2$ is also a mapping.

Let Ω_1 and Ω_2 be sets of mappings. Then the join of, the union of, the difference between and the left outer-join between Ω_1 and Ω_2 are defined as follows [11]:

$$\begin{aligned}
\Omega_1 \bowtie \Omega_2 &= \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ and } \mu_1, \mu_2 \text{ are compatible mappings}\},\\
\Omega_1 \cup \Omega_2 &= \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\},\\
\Omega_1 \smallsetminus \Omega_2 &= \{\mu \in \Omega_1 \mid \text{ for all } \mu' \in \Omega_2, \mu \text{ and } \mu' \text{ are not compatible}\},\\
\Omega_1 ⟕ \Omega_2 &= (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \smallsetminus \Omega_2).
\end{aligned}$$

Next we use the preceding operators to give semantics to graph pattern expressions, BINDINGS queries and SELECT queries. More specifically, we define this semantics as a function $[\![\cdot]\!]_G$, which takes as input any of these types of queries and returns a set of mappings. In this definition, we assume given a partial function ep from the set I of IRIs such that for every $c \in I$, if $\text{ep}(c)$ is defined, then $\text{ep}(c)$ is an RDF graph. Intuitively, function ep is defined for an element $c \in I$ ($c \in \text{dom}(\text{ep})$) if and only if c is the IRI of a SPARQL endpoint, and $\text{ep}(c)$ is the default RDF graph of that endpoint[1].

[1] For simplicity, we only assume a single (default) graph and no named graphs per remote SPARQL endpoint.

Moreover, in this definition $\mu_\emptyset$ represents the mapping with empty domain (which is compatible with any other mapping).

The evaluation of a graph pattern P over an RDF graph G, denoted by $[\![P]\!]_G$, is defined recursively as follows (due to the lack of space, we refer the reader to the extended version of the paper for the definition of the semantics of the FILTER operator):

(1) If P is a triple pattern t, then $[\![P]\!]_G = \{\mu \mid \text{dom}(\mu) = \text{var}(t) \text{ and } \mu(t) \in G\}$.
(2) If P is $(P_1 \text{ AND } P_2)$, then $[\![P]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$.
(3) If P is $(P_1 \text{ OPT } P_2)$, then $[\![P]\!]_G = [\![P_1]\!]_G ⟕ [\![P_2]\!]_G$.
(4) If P is $(P_1 \text{ UNION } P_2)$, then $[\![P]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$.
(5) If P is $(\text{SERVICE } c\, P_1)$ with $c \in I$, then

$$[\![P]\!]_G = \begin{cases} [\![P_1]\!]_{\text{ep}(c)} & \text{if } c \in \text{dom(ep)} \\ \{\mu_\emptyset\} & \text{otherwise} \end{cases}$$

(6) If P is $(\text{SERVICE } ?X\ P_1)$ with $?X \in V$, then $[\![P]\!]_G$ is equal to:

$$\bigcup_{c \in I} \Big\{ \mu \mid \text{there exists } \mu' \in [\![(\text{SERVICE } c\, P_1)]\!]_G \text{ s.t. } \text{dom}(\mu) = (\text{dom}(\mu') \cup \{?X\}),$$
$$\mu(?X) = c \text{ and } \mu(?Y) = \mu'(?Y) \text{ for every } ?Y \in \text{dom}(\mu') \Big\}$$

Moreover, the semantics of BINDINGS queries is defined as follows. Given a list $S = [?X_1, \ldots, ?X_\ell]$ of pairwise distinct variables, where $\ell \geq 1$, and a list $A = [a_1, \ldots, a_\ell]$ of values from $(I \cup L \cup \{\text{UNBOUND}\})$, let $\mu_{S,A}$ be a mapping with domain $\{?X_i \mid i \in \{1, \ldots, \ell\} \text{ and } a_i \in (I \cup L)\}$ and such that $\mu_{S,A}(?X_i) = a_i$ for every $?X_i \in \text{dom}(\mu_{S,A})$. Then

(7) If $P = (P_1 \text{ BINDINGS } S\ \{A_1, \ldots, A_n\})$ is a BINDINGS query:

$$[\![P]\!]_G = [\![P_1]\!]_G \bowtie \{\mu_{S,A_1}, \ldots, \mu_{S,A_n}\}.$$

Finally, the semantics of SELECT queries is defined as follows. Given a mapping $\mu : V \to (I \cup B \cup L)$ and a set of variables $W \subseteq V$, the restriction of μ to W, denoted by $\mu_{|_W}$, is a mapping such that $\text{dom}(\mu_{|_W}) = (\text{dom}(\mu) \cap W)$ and $\mu_{|_W}(?X) = \mu(?X)$ for every $?X \in (\text{dom}(\mu) \cap W)$. Then

(8) If $P = (\text{SELECT } W\ P_1)$ is a SELECT query: $[\![P]\!]_G = \{\mu_{|_W} \mid \mu \in [\![P_1]\!]_G\}$.

It is important to notice that the rules (1)–(4) above were introduced in [11], while we propose in the rules (5)–(8) a semantics for the operators SERVICE and BINDINGS introduced in [12]. Intuitively, if $c \in I$ is the IRI of a SPARQL endpoint, then the idea behind the definition of $(\text{SERVICE } c\, P_1)$ is to evaluate query P_1 in the SPARQL endpoint specified by c. On the other hand, if $c \in I$ is not the IRI of a SPARQL endpoint, then $(\text{SERVICE } c\, P_1)$ leaves unbounded all the variables in P_1, as this query cannot be evaluated in this case. This idea is formalized by making $\mu_\emptyset$ the only mapping in the evaluation of $(\text{SERVICE } c\, P_1)$ if $c \notin \text{dom(ep)}$. In the same way, $(\text{SERVICE } ?X\ P_1)$

is defined by considering all the possible IRIs for the variable $?X$, that is, all the values $c \in I$. In fact, $(\text{SERVICE } ?X \; P_1)$ is defined as the union of the evaluation of the graph patterns $(\text{SERVICE } \; c \; P_1)$ for the values $c \in I$, but also storing in $?X$ the IRIs from where the values of the variables in P_1 are coming from. Finally, the idea behind the definition of $(P_1 \;\; \text{BINDINGS} \;\; S \; \{A_1, \ldots, A_n\})$ is to constrain the values of the variables in S to the values specified in $A_1, \ldots, A_n$.

Example 1. Assume that G is an RDF graph that uses triples of the form $(a, \text{service\_address}, b)$ to indicate that a SPARQL endpoint with name a is located at the IRI b. Moreover, let P be the following SPARQL query:

$$\Big[\text{SELECT } \{?X, ?N, ?E\} \\ \Big(\Big((?X, \text{service\_address}, ?Y) \text{ AND } (\text{SERVICE } ?Y \; (?N, \text{email}, ?E))\Big) \\ \text{BINDINGS } [?N] \; \{[\,\text{John}\,], [\,\text{Peter}\,]\}\Big)\Big]$$

Query P is used to compute the list of names and email addresses that can be retrieved from the SPARQL endpoints stored in an RDF graph. In fact, if $\mu \in [\![P]\!]_G$, then $\mu(?X)$ is the name of a SPARQL endpoint stored in G, $\mu(?N)$ is the name of a person stored in that SPARQL endpoint and $\mu(?E)$ is the email address of that person. Moreover, the operator BINDINGS in this query is used to filter the values of the variable $?N$. Specifically, if $\mu \in [\![P]\!]_G$, then $\mu(?N)$ is either John or Peter. $\square$

The goal of the rules (5)–(8) is to define in an unambiguous way what the result of evaluating an expression containing the operators SERVICE and BINDINGS should be. As such, these rules should not be considered as an implementation of the language. In fact, a direct implementation of the rule (6), that defines the semantics of a pattern of the form $(\text{SERVICE } ?X \; P_1)$, would involve evaluating a particular query in every possible SPARQL endpoint, which is obviously infeasible in practice. In the next section, we face this issue and, in particular, we introduce a syntactic condition on SPARQL queries that ensures that a pattern of the form $(\text{SERVICE } ?X \; P_1)$ can be evaluated by only considering a finite set of SPARQL endpoints, whose IRIs are actually taken from the RDF graph where the query is being evaluated.

3 On Evaluating the SERVICE Operator

As we pointed out in the previous section, the evaluation of a pattern of the form $(\text{SERVICE } ?X \; P)$ is infeasible unless the variable $?X$ is bound to a finite set of IRIs. This notion of *boundedness* is one of the most significant and unclear concepts in the SPARQL federation extension. In fact, the current version of the specification [12] only specifies that a variable $?X$ in a pattern of the form $(\text{SERVICE } ?X \; P)$ *must be bound*, but without providing a formal definition of what that means. Here we provide a formalization of this concept, studying the complexity issues associated with it.

3.1 The Notion of Boundedness

In Example 1, we present a SPARQL query containing a pattern (SERVICE $?Y$ $(?N, \text{email}, ?E)$). Given that variable $?Y$ is used to store the address of a remote SPARQL endpoint to be queried, it is important to assign a value to $?Y$ prior to the evaluation of the SERVICE pattern. In the case of the query in Example 1, this needs of a simple strategy: given an RDF graph G, first compute $[\![(?X, \text{service\_address}, ?Y)]\!]_G$, and then for every μ in this set, compute $[\![(\text{SERVICE}\ a\ (?N, \text{email}, ?E))]\!]_G$ with $a = \mu(?Y)$. More generally, SPARQL pattern $(\text{SERVICE}\ ?Y\ (?N, \text{email}, ?E))$ can be evaluated in this case as only a finite set of values from the domain of G need to be considered as the possible values of $?Y$. This idea naturally gives rise to the following notion of boundedness for the variables of a SPARQL query. In the definition of this notion, $\text{dom}(G)$ refers to the domain of G, that is, the set of elements from $(I \cup B \cup L)$ that are mentioned in G, and $\text{dom}(P)$ refers to the set of elements from $(I \cup L)$ that are mentioned in P.

Definition 1 (Boundedness). *Let P be a SPARQL query and $?X \in \text{var}(P)$. Then $?X$ is bound in P if one of the following conditions holds:*

- *P is either a graph pattern or a BINDINGS query, and for every RDF graph G and mapping $\mu \in [\![P]\!]_G$, it holds that $?X \in \text{dom}(\mu)$ and $\mu(?X) \in (\text{dom}(G) \cup \text{dom}(P))$.*
- *P is a SELECT query $(\text{SELECT}\ W\ P_1)$ and $?X$ is bound in P_1.*

The BINDINGS operator can make a variable $?X$ in a query P to be bound by assigning to it a fixed set of values. Given that these values are not necessarily mentioned in the RDF graph G where P is being evaluated, the previous definition first imposes the condition that $?X \in \text{dom}(\mu)$, and then not only considers the case $\mu(?X) \in \text{dom}(G)$ but also the case $\mu(?X) \in \text{dom}(P)$. As an example of the above definition, we note that variable $?Y$ is bound in the graph pattern

$$P_1 = ((?X, \text{service\_address}, ?Y)\ \text{AND}\ (\text{SERVICE}\ ?Y\ (?N, \text{email}, ?E))),$$

as for every RDF graph G and mapping $\mu \in [\![P_1]\!]_G$, we know that $?Y \in \text{dom}(\mu)$ and $\mu(?Y) \in \text{dom}(G)$. Moreover, we also have that variable $?Y$ is bound in $(\text{SELECT}\ \{?X, ?N, ?E\}\ P_1)$ as $?Y$ is bound in graph pattern P_1.

A natural way to ensure that a SPARQL query P can be evaluated in practice is by imposing the restriction that for every sub-pattern $(\text{SERVICE}\ ?X\ P_1)$ of P, it holds that $?X$ is bound in P. However, in the following theorem we show that such a condition is undecidable and, thus, a SPARQL query engine would not be able to check it in order to ensure that a query can be evaluated.

Theorem 1. *The problem of verifying, given a SPARQL query P and a variable $?X \in \text{var}(P)$, whether $?X$ is bound in P is undecidable.*

The fact that the notion of boundedness is undecidable prevents one from using it as a restriction over the variables in SPARQL queries. To overcome this limitation, we introduce here a syntactic condition that ensures that a variable is bound in a pattern and that can be efficiently verified.

Definition 2 (Strong boundedness). *Let P be a SPARQL query. Then the set of strongly bound variables in P, denoted by* $\mathrm{SB}(P)$*, is recursively defined as follows:*

- *if $P = t$, where t is a triple pattern, then* $\mathrm{SB}(P) = \mathrm{var}(t)$*;*
- *if $P = (P_1$ AND $P_2)$, then* $\mathrm{SB}(P) = \mathrm{SB}(P_1) \cup \mathrm{SB}(P_2)$*;*
- *if $P = (P_1$ UNION $P_2)$, then* $\mathrm{SB}(P) = \mathrm{SB}(P_1) \cap \mathrm{SB}(P_2)$*;*
- *if $P = (P_1$ OPT $P_2)$ or $P = (P_1$ FILTER $R)$, then* $\mathrm{SB}(P) = \mathrm{SB}(P_1)$*;*
- *if $P =$ (SERVICE c P_1), with $c \in I$, or $P =$ (SERVICE $?X$ P_1), with $?X \in V$, then* $\mathrm{SB}(P) = \emptyset$*;*
- *if $P = (P_1$ BINDINGS S $\{A_1, \ldots, A_n\})$, then* $\mathrm{SB}(P) = \mathrm{SB}(P_1) \cup \{?X \mid$ *$?X$ is in S and for every $i \in \{1, \ldots, n\}$, it holds that* $?X \in \mathrm{dom}(\mu_{S,A_i})\}$.
- *if $P =$ (SELECT W P_1), then* $\mathrm{SB}(P) = (W \cap \mathrm{SB}(P_1))$.

The previous definition recursively collects from a SPARQL query P a set of variables that are guaranteed to be bound in P. For example, if P is a triple pattern t, then $\mathrm{SB}(P) = \mathrm{var}(t)$ as one knows that for every variable $?X \in \mathrm{var}(t)$ and for every RDF graph G, if $\mu \in [\![t]\!]_G$, then $?X \in \mathrm{dom}(\mu)$ and $\mu(?X) \in \mathrm{dom}(G)$. In the same way, if $P = (P_1$ AND $P_2)$, then $\mathrm{SB}(P) = \mathrm{SB}(P_1) \cup \mathrm{SB}(P_2)$ as one knows that if $?X$ is bound in P_1 or in P_2, then $?X$ is bound in P. As a final example, notice that if $P = (P_1$ BINDINGS S $\{A_1, \ldots, A_n\})$ and $?X$ is a variable mentioned in S such that $?X \in \mathrm{dom}(\mu_{S,A_i})$ for every $i \in \{1, \ldots, n\}$, then $?X \in \mathrm{SB}(P)$. In this case, one knows that $?X$ is bound in P since $[\![P]\!]_G = [\![P_1]\!]_G \bowtie \{\mu_{S,A_1}, \ldots, \mu_{S,A_n}\}$ and $?X$ is in the domain of each one of the mappings μ_{S,A_i}, which implies that $\mu(?X) \in \mathrm{dom}(P)$ for every $\mu \in [\![P]\!]_G$. In the following proposition, we formally show that our intuition about $\mathrm{SB}(P)$ is correct, in the sense that every variable in this set is bound in P.

Proposition 1. *For every SPARQL query P and variable $?X \in \mathrm{var}(P)$, if $?X \in \mathrm{SB}(P)$, then $?X$ is bound in P.*

Given a SPARQL query P and a variable $?X \in \mathrm{var}(P)$, it can be efficiently verified whether $?X$ is strongly bound in P. Thus, a natural and efficiently verifiable way to ensure that a SPARQL query P can be evaluated in practice is by imposing the restriction that for every sub-pattern (SERVICE $?X$ P_1) of P, it holds that $?X$ is strongly bound in P. However, this notion still needs to be modified in order to be useful in practice, as shown by the following examples.

Example 2. Assume first that P_1 is the following graph pattern:

$$\begin{aligned} P_1 \; = \; & ((?X, \text{service\_description}, ?Z) \text{ UNION} \\ & \quad ((?X, \text{service\_address}, ?Y) \text{ AND } (\text{SERVICE } ?Y \; (?N, \text{email}, ?E)))). \end{aligned}$$

That is, either $?X$ and $?Z$ store the name of a SPARQL endpoint and a description of its functionalities, or $?X$ and $?Y$ store the name of a SPARQL endpoint and the IRI where it is located (together with a list of names and email addresses retrieved from that location). Variable $?Y$ is neither bound nor strongly bound in P_1. However, there is a simple strategy that ensures that P_1 can be evaluated over an RDF graph G: first compute $[\![(?X, \text{service\_description}, ?Z)]\!]_G$, then compute $[\![(?X, \text{service\_address}, ?Y)]\!]_G$, and finally for every μ in the set

$[\![(?X, \text{service\_address}, ?Y)]\!]_G$, compute $[\![(\text{SERVICE}\ a\ (?N, \text{email}, ?E))]\!]_G$ with $a = \mu(?Y)$. In fact, the reason why P_1 can be evaluated in this case is that $?Y$ is bound (and strongly bound) in the sub-pattern $((?X, \text{service\_address}, ?Y)\ \text{AND}\ (\text{SERVICE}\ ?Y\ (?N, \text{email}, ?E)))$ of P_1.

As a second example, assume that G is an RDF graph that uses triples of the form $(a_1, \text{related\_with}, a_2)$ to indicate that the SPARQL endpoints located at the IRIs a_1 and a_2 store related data. Moreover, assume that P_2 is the following graph pattern:

$$\begin{aligned} P_2 \ = \ & ((?U_1, \text{related\_with}, ?U_2)\ \text{AND} \\ & (\text{SERVICE}\ ?U_1\ ((?N, \text{email}, ?E)\ \text{OPT}\ (\text{SERVICE}\ ?U_2\ (?N, \text{phone}, ?F))))). \end{aligned}$$

When this query is evaluated over the RDF graph G, it returns for every tuple $(a_1, \text{related\_with}, a_2)$ in G, the list of names and email addresses that that can be retrieved from the SPARQL endpoint located at a_1, together with the phone number for each person in this list for which this data can be retrieved from the SPARQL endpoint located at a_2 (recall that graph pattern $(\text{SERVICE}\ ?U_2\ (?N, \text{phone}, ?F))$ is nested inside the first SERVICE operator in P_2). To evaluate this query over an RDF graph, first it is necessary to determine the possible values for variable $?U_1$, and then to submit the query $((?N, \text{email}, ?E)\ \text{OPT}\ (\text{SERVICE}\ ?U_2\ (?N, \text{phone}, ?F)))$ to each one of the endpoints located at the IRIs stored in $?U_1$. In this case, variable $?U_2$ is bound (and also strongly bound) in P_2. However, this variable is not bound in the graph pattern $((?N, \text{email}, ?E)\ \text{OPT}\ (\text{SERVICE}\ ?U_2\ (?N, \text{phone}, ?F)))$, which has to be evaluated in some of the SPARQL endpoints stored in the RDF graph where P_2 is being evaluated, something that is infeasible in practice. Notice that the difficulties in evaluating P_2 are caused by the nesting of SERVICE operators (more precisely, by the fact that P_2 has a sub-pattern of the form $(\text{SERVICE}\ ?X_1\ Q_1)$, where Q_1 has in turn a sub-pattern of the form $(\text{SERVICE}\ ?X_2\ Q_2)$ such that $?X_2$ is bound in P_2 but not in Q_1). $\square$

In the following section, we use the concept of strongly boundedness to define a notion that ensures that a SPARQL query containing the SERVICE operator can be evaluated in practice, and which takes into consideration the ideas presented in Example 2.

3.2 The Notion of Service-Safeness: Considering Sub-patterns and Nested Service Operators

The goal of this section is to provide a condition that ensures that a SPARQL query containing the SERVICE operator can be safely evaluated . To this end, we first need to introduce some terminology. Given a SPARQL query P, define $\mathcal{T}(P)$ as the *parse* tree of P. In this tree, every node corresponds to a sub-pattern of P. An example of a parse tree of a pattern Q is shown in Figure 1. In this figure, $u_1, u_2, u_3, u_4, u_5, u_6$ are the identifiers of the nodes of the tree, which are labeled with the sub-patterns of Q. It is important to notice that in this tree we do not make any distinction between the different operators in SPARQL, we just store the structure of the sub-patterns of a SPARQL query.

Tree $\mathcal{T}(P)$ is used to define the notion of service-boundedness, which extends the concept of boundedness, introduced in the previous section, to consider variables that

$$\begin{tikzcd}
 & u_1 : ((?Y, a, ?Z) \text{ UNION } ((?X, b, c) \text{ AND } (\text{SERVICE } ?X \ (?Y, a, ?Z)))) \arrow[dl] \arrow[dr] & & \\
u_2 : (?Y, a, ?Z) & & u_3 : ((?X, b, c) \text{ AND } (\text{SERVICE } ?X \ (?Y, a, ?Z))) \arrow[dl] \arrow[dr] & \\
 & u_4 : (?X, b, c) & & u_5 : (\text{SERVICE } ?X \ (?Y, a, ?Z)) \arrow[d] \\
 & & & u_6 : (?Y, a, ?Z)
\end{tikzcd}$$

Fig. 1. Parse tree $\mathcal{T}(Q)$ for the graph pattern $Q = ((?Y, a, ?Z) \text{ UNION } ((?X, b, c) \text{ AND } (\text{SERVICE } ?X \ (?Y, a, ?Z))))$

are bound inside sub-patterns and nested SERVICE operators. It should be noticed that these two features were identified in the previous section as important for the definition of a notion of boundedness (see Example 2).

Definition 3 (Service-boundedness). *A SPARQL query P is service-bound if for every node u of $\mathcal{T}(P)$ with label* $(\text{SERVICE } ?X \ P_1)$*, it holds that: (1) there exists a node v of $\mathcal{T}(P)$ with label P_2 such that v is an ancestor of u in $\mathcal{T}(P)$ and $?X$ is bound in P_2; (2) P_1 is service-bound.*

For example, query Q in Figure 1 is service-bound. In fact, condition (1) of Definition 3 is satisfied as u_5 is the only node in $\mathcal{T}(Q)$ having as label a SERVICE graph pattern, in this case $(\text{SERVICE } ?X \ (?Y, a, ?Z))$, and for the node u_3, it holds that: u_3 is an ancestor of u_5 in $\mathcal{T}(P)$, the label of u_3 is $P = ((?X, b, c) \text{ AND } (\text{SERVICE } ?X \ (?Y, a, ?Z)))$ and $?X$ is bound in P. Moreover, condition (2) of Definition 3 is satisfied as the sub-pattern $(?Y, a, ?Z)$ of the label of u_5 is also service-bound.

The notion of service-boundedness captures our intuition about the condition that a SPARQL query containing the SERVICE operator should satisfy. Unfortunately, the following theorem shows that such a condition is undecidable and, thus, a query engine would not be able to check it in order to ensure that a query can be evaluated.

Theorem 2. *The problem of verifying, given a SPARQL query P, whether P is service-bound is undecidable.*

As for the case of the notion of boundedness, the fact that the notion of service-boundedness is undecidable prevents one from using it as a restriction over the variables used in SERVICE calls. To overcome this limitation, we replace the restriction that the variables used in SERVICE calls are bound by the decidable restriction that they are strongly bound. In this way, we obtain a syntactic condition over SPARQL patterns that ensures that they are service-bound, and which can be efficiently verified.

Definition 4 (Service-safeness). *A SPARQL query P is service-safe if for every node u of $\mathcal{T}(P)$ with label* $(\text{SERVICE } ?X \ P_1)$*, it holds that: (1) there exists a node v of $\mathcal{T}(P)$ with label P_2 such that v is an ancestor of u in $\mathcal{T}(P)$ and $?X \in \text{SB}(P_2)$; (2) P_1 is service-safe.*

Proposition 2. *If a SPARQL query P is service-safe, then P is service-bound.*

The notion of service-safeness is used in our system to verify that a SPARQL pattern can be evaluated in practice. We conclude this section by pointing out that it can be efficiently verified whether a SPARQL query P is service-safe, by using a bottom-up approach over the parse tree $\mathcal{T}(P)$ of P.

4 Optimizing the Evaluation of the OPTIONAL Operator in SPARQL Federated Queries

If a SPARQL query Q including the SERVICE operator has to be evaluated in a SPARQL endpoint A, then some of the sub-queries of Q may have to be evaluated in some external SPARQL endpoints. Thus, the problem of optimizing the evaluation of Q in A, and, in particular, the problem of reordering Q in A to optimize this evaluation, becomes particularly relevant in this scenario, as in some cases one cannot rely on the optimizers of the external SPARQL endpoints. Motivating by this, we present in this section some optimization techniques that extend the techniques presented in [11] to the case of SPARQL queries using the SERVICE operator, and which can be applied to a considerable number of SPARQL federated queries.

4.1 Optimization via Well-Designed Patterns

In [11,17], the authors study the complexity of evaluating the fragment of SPARQL consisting of the operators AND, UNION, OPT and FILTER. One of the conclusions of these papers is that the main source of complexity in SPARQL comes from the use of the OPT operator. In light of these results, it was introduced in [11] a fragment of SPARQL that forbids a special form of interaction between variables appearing in optional parts, which rarely occurs in practice. The patterns in this fragment, which are called well-designed patterns [11], can be evaluated more efficiently and are suitable for reordering and optimization. In this section, we extend the definition of the notion of being well-designed to the case of SPARQL patterns using the SERVICE operator, and prove that the reordering rules proposed in [11], for optimizing the evaluation of well-designed patterns, also hold in this extension. The use of these rules allows to reduce the number of tuples being transferred and joined in federated queries, and hence our implementation benefits from this as shown in Section 5.

Let P be a graph pattern constructed by using the operators AND, OPT, FILTER and SERVICE, and assume that P satisfies the safety condition that for every sub-pattern $(P_1 \text{ FILTER } R)$ of P, it holds that $\text{var}(R) \subseteq \text{var}(P_1)$. Then, by following [11], we say that P is well-designed if for every sub-pattern $P' = (P_1 \text{ OPT } P_2)$ of P and for every variable $?X$ occurring in P: If $?X$ occurs both inside P_2 and outside P', then it also occurs in P_1. All the graph patterns given in the previous sections are well-designed. On the other hand, the following pattern P is not well-designed:

$$((?X, \text{nickname}, ?Y) \text{ AND } (\text{SERVICE } c\, ((?X, \text{email}, ?U) \text{ OPT } (?Y, \text{email}, ?V)))),$$

as for the sub-pattern $P' = (P_1 \text{ OPT } P_2)$ of P with $P_1 = (?X, \text{email}, ?U)$ and $P_2 = (?Y, \text{email}, ?V))$, we have that $?Y$ occurs in P_2 and outside P' in the triple pattern $(?X, \text{nickname}, ?Y)$, but it does not occur in P_1. Given an RDF graph G, graph pattern P retrieves from G a list of people with their nicknames, and retrieves from the SPARQL endpoint located at the IRI c the email addresses of these people and, optionally, the email addresses associated to their nicknames. What is unnatural about this graph pattern is the fact that $(?Y, \text{email}, ?V)$ is giving optional information for $(?X, \text{nickname}, ?Y)$, but in P appears as giving optional information for $(?X, \text{name}, ?U)$. In fact, it could happen that some of the results retrieved by using the triple pattern $(?X, \text{nickname}, ?Y)$ are not included in the final answer of P, as the value of variable $?Y$ in these intermediate results could be incompatible with the values for this variable retrieved by using the triple pattern $(?Y, \text{email}, ?V)$.

In the following proposition, we show that well-designed patterns including the SERVICE operator are suitable for reordering and, thus, for optimization.

Proposition 3. *Let P be a well-designed pattern and P' a pattern obtained from P by using one of the following reordering rules:*

$$\begin{aligned}
((P_1 \text{ OPT } P_2) \text{ FILTER } R) &\longrightarrow ((P_1 \text{ FILTER } R) \text{ OPT } P_2),\\
(P_1 \text{ AND } (P_2 \text{ OPT } P_3)) &\longrightarrow ((P_1 \text{ AND } P_2) \text{ OPT } P_3),\\
((P_1 \text{ OPT } P_2) \text{ AND } P_3) &\longrightarrow ((P_1 \text{ AND } P_3) \text{ OPT } P_2).
\end{aligned}$$

Then P' is a well-designed pattern equivalent to P.

The proof of this proposition is a simple extension of the proof of Proposition 4.10 in [11]. In the following section, we show that the use of these rules can have a considerable impact in the cost of evaluating graph patterns.

5 Implementation of SPARQL-DQP and Well-Designed Patterns Optimization

In this section, we describe how we implemented and evaluated the optimization techniques presented in the previous section. In particular, we demonstrate that they effectively decrease the processing time of SPARQL 1.1 federated queries.

5.1 Implementation: SPARQL-DQP

We have implemented the rewriting rules described in Section 4.1 in SPARQL-DQP [5], together with a bottom up algorithm for checking the condition of being well-designed. SPARQL-DQP is a query evaluation system built on top of OGSA-DAI [3] and OGSA-DQP [10]. OGSA-DAI is a generic service-based data access, integration, transformation and delivery framework that allows executing data-centric workflows involving heterogeneous data resources. OGSA-DAI is integrated in Apache Tomcat and within the Globus Toolkit, and is used in OMII-UK, the UK e-Science platform. OGSA-DQP is the Distributed Query Processing extension of OGSA-DAI, which access distributed OGSA-DAI data resources and provides parallelization mechanisms. SPARQL-DQP [5] extends this framework with new SPARQL parsers, logical query plan builders, operators and optimizers for distributed query processing. The main reason for selecting this framework is that it provides built-in infrastructure to support DQP and enables handling large datasets and tuple streams, which may result from the execution of queries in different query services and data sources. The low level technical details of our implementation can be found in [5].

5.2 Evaluation

In our evaluation, we compare the results and performance of our system with other similar systems that provide some support for SPARQL query federation. Currently, the engines supporting the official SPARQL 1.1 federation extension are: DARQ [14], Networked Graphs [15] and ARQ, which is available via an online web service (`http://www.sparql.org/`) as well as a library for Jena (`http://jena.sourceforge.net/`). Other system that supports distributed RDF querying is presented in [18]. We do not consider this system here as it uses the query language SeRQL instead of SPARQL.

The objective of our evaluation is to show first that we can handle SPARQL queries that comply with the federated extension, and second that the optimization techniques proposed in Section 4.1 actually reduce the time needed to process queries. We have checked for existing SPARQL benchmarks like the Berlin SPARQL Benchmark [4], SP^{2}Bench [16] and the benchmark proposed in [7]. Unfortunately for our purposes, the first two are not designed for a distributed environment, while the third one is based on a federated scenario but is not as comprehensive as the Berlin SPARQL Benchmark and SP^{2}Bench. Thus, we decided to base our evaluation on some queries from the life sciences domain, similar to those in [7] but using a base query and increasing its complexity like in [4]. These queries are real queries used by Bio2RDF experts.

Datasets description. The Bio2RDF datasets contains 2,3 billion triples organized around 40 datasets with sometimes overlapping information. The Bio2RDF datasets that we have used in our benchmark are: Entrez Gene (13 million triples, stored in the local endpoint sparql-pubmed), Pubmed (797 million triples), HHPID (244,021 triples) and MeSH (689,542 triples, stored in the local endpoint sparql-mesh). One of the practical problems that these benchmarks have is that public SPARQL endpoints normally restrict the amount of results that they provide. To overcome this limitation we installed Entrez Gene and MeSH in servers without these restrictions. We also divided them in files of 300,000 triples, creating endpoints for each one of them.

Queries used in the evaluation. We used 7 queries in our evaluation. The query structure follows the following path: using the Pubmed references obtained from the Entrez gene dataset, we access the Pubmed endpoint (queries Q1 and Q2). In these queries, we retrieve information about genes and their references in the Pubmed dataset. From Pubmed we access the information in the National Library of Medicine's controlled vocabulary thesaurus (queries Q3 and Q4), stored at MeSH endpoint, so we have more complete information about such genes. Finally, to increase the data retrieved by our queries we also access the HHPID endpoint (queries Q5, Q6 and Q7), which is the knowledge base for the HIV-1 protein. The queries, in increasing order of complexity, can be found at `http://www.oeg-upm.net/files/sparql-dqp/`. Next we show query Q4 to give the reader an idea of the type of queries that we are considering:

```
SELECT ?pubmed ?gene1 ?mesh ?descriptor ?meshReference
WHERE
{
  {SERVICE <http://127.0.0.1:2020/sparql-pubmed> {
     ?gene1 <http://bio2rdf.org/geneid_resource:pubmed_xref> ?pubmed .}}.
  {SERVICE <http://pubmed.bio2rdf.org/sparql> {
     ?pubmed <http://bio2rdf.org/pubmed_resource:meshref> ?mesh .
     ?mesh   <http://bio2rdf.org/pubmed_resource:descriptor> ?descriptor .}}.
   OPTIONAL { SERVICE <http://127.0.0.1:2021/sparql-mesh> {
     ?meshReference <http://www.w3.org/2002/07/owl#sameAs> ?descriptor .}}.
}
```

Results. Our evaluation was done in an Amazon EC2 instance. The instance has 2 cores and 7.5 GB of memory run by Ubuntu 10.04. The data used in this evaluation, together with the generated query plans and the original queries in Java formatting, can be found at `http://www.oeg-upm.net/files/sparql-dqp/`. The results of our evaluation are shown in the following table:

Query	Not optimized SPARQL-DQP	**Optimized SPARQL-DQP**	DARQ	NetworkedGraphs	ARQ
Q1	79,000ms.	**79,000ms.**	10+ min.	10+ min.	440,296ms.
Q2	64,179ms.	**64,179ms.**	10+ min.	10+ min.	10+ min.
Q3	134,324ms.	**134,324ms.**	10+ min.	10+ min.	10+ min.
Q4	152,559ms.	**136,482ms.**	10+ min.	10+ min.	10+ min.
Q5	146,575ms.	**146,575ms.**	10+ min.	10+ min.	10+ min.
Q6	322,792ms.	**79,178ms.**	10+ min.	10+ min.	10+ min.
Q7	350,554ms.	**83,153ms.**	10+ min.	10+ min.	10+ min.

A first clear advantage of our implementation is the ability to use asynchronous calls facilitated by the use of indirect access mode, what means that we do not get time out in any of the queries. This time out happens when accessing an online distributed query processing like in the case of ARQ (`www.sparql.org/query`). It is important to note that the ability to handle this type of queries is essential for many types of data-intensive applications, such as those based on Bio2RDF. Data transfer also plays a key role in query response times. For example, in some queries the local query engine received 150,000 results from Entrez gene, 10,000 results from Pubmed, 23,841 results from MeSH and 10,000 results from HHPID. The implemented optimizations are less noticeable when the amount of transferred data is fewer.

It is possible to observe three different sets of results from this preliminary evaluation. The first set (Q1–Q3 and Q5) are those that are not optimized because the reordering rules in Section 4.1 are not applicable. The second query group (Q4) represents the class of queries that can be optimized using our approach, but where the difference is not too relevant, because the less amount of transferred data. The last group of queries (Q6–Q7) shows a clear optimization when using the well-designed patterns rewriting rules. For example, in query 6 the amount of transferred data varies from a join of $150,000 \times 10,000$ tuples to a join of $10,000 \times 23,841$ tuples (using Entrez, Pubmed and MeSH endpoints), which highly reduces the global processing time of the query. Regarding the comparison with other systems, they do not properly handle these amounts of data. We represent as 10+ min. those queries that need more than 10 minutes to be answered.

In summary, we have shown that our implementation provides better results than other similar systems. Besides, we have also shown that our implementation, which benefits from an indirect access mode, can be more appropriate to deal with large datasets.

Acknowledgments. We thank the anonymous referees, the OGSA-DAI team (specially Ally Hume), Marc-Alexandre Nolin, Jorge Pérez and Axel Polleres for their help with this work. This research was supported by ADMIRE project FP7 ICT-215024 and FONDECYT grant 1090565.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. 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)
3. Antonioletti, M., et al.: OGSA-DAI 3.0 - The Whats and the Whys. UK e-Science All Hands Meeting, pp. 158–165 (2007)
4. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. Int. J. Semantic Web Inf. Syst. 5(2), 1–24 (2009)
5. Buil, C., Corcho, O.: Federating Queries to RDF repositories. Technical Report (2010), http://oa.upm.es/3302/
6. Durst, M,. Suignard, M.: Rfc 3987, Internationalized Resource Identifiers (IRIs), http://www.ietf.org/rfc/rfc3987.txt
7. Haase, P., Mathäß, T., Ziller, M.: An evaluation of approaches to federated query processing over linked data. In: I-SEMANTICS (2010)
8. Harris, S., Seaborne, A.: SPARQL 1.1 Query. W3C Working Draft (June 1, 2010), http://www.w3.org/TR/sparql11-query/
9. Klyne, G., Carroll, J.J., McBride, B.: Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation (February 10, 2004), http://www.w3.org/TR/rdf-concepts/
10. Lynden, S., et al.: The design and implementation of OGSA-DQP: A service-based distributed query processor. Future Generation Computer Systems 25(3), 224–236 (2009)
11. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. TODS 34(3) (2009)
12. Prud'hommeaux, E.: SPARQL 1.1 Federation Extensions. W3C Working Draft (June 1, 2010), http://www.w3.org/TR/sparql11-federated-query/
13. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (January 15, 2008), http://www.w3.org/TR/rdf-sparql-query/
14. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
15. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the Web. In: WWW, pp. 585–594 (2008)
16. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: A SPARQL Performance Benchmark. In: ICDE, pp. 222–233 (2009)
17. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL query optimization. In: ICDT, pp. 4–33 (2010)
18. Stuckenschmidt, H., Vdovjak, R., Geert-Jan, H., Broekstra, J.: Index structures and algorithms for querying distributed RDF repositories. In: WWW, pp. 631–639 (2004)

Grr: Generating Random RDF*

Daniel Blum and Sara Cohen

School of Computer Science and Engineering
The Hebrew University of Jerusalem
{daniel.blum@mail,sara@cs}.huji.ac.il

Abstract. This paper presents GRR, a powerful system for generating random RDF data, which can be used to test Semantic Web applications. GRR has a SPARQL-like syntax, which allows the system to be both powerful and convenient. It is shown that GRR can easily be used to produce intricate datasets, such as the LUBM benchmark. Optimization techniques are employed, which make the generation process efficient and scalable.

1 Introduction

Testing is one of most the critical steps of application development. For data-centric applications, testing is a challenge both due to the large volume of input data needed, and due to the intricate constraints that this data must satisfy. Thus, while finding or generating input data for testing is pivotal in the development of data-centric applications, it is often a difficult undertaking. This is a significant stumbling block in system development, since considerable resources must be expended to generate test data.

Automatic generation of data has been studied extensively for relational databases (e.g., [4,7,11,13]), and there has also been considerable progress on this problem for XML databases [1,3,8]. This paper focuses on generating test data for RDF databases. While some Semantic Web applications focus on varied and unexpected types of data, there are also many others that target specific domains. For such applications, to be useful, datasets used should have at least two properties. *First,* the data structure must conform to the schema of the target application. *Second,* the data should match the expected data distribution of the target application.

Currently, there are several distinct sources for RDF datasets. First, there are *downloadable RDF datasets* that can be found on the web, e.g., Barton libraries, UniProt catalog sequence, and WordNet. RDF Benchmarks, which include both large datasets and sample queries, have also been developed, e.g., the Lehigh University Benchmark (LUBM) [10] (which generates data about universities), the SP^{2}Bench Benchmark [14] (which provides DBLP-style data) and the Berlin SPARQL Benchmark [5] (which is built around an e-commerce use case). Such downloadable RDF datasets are usually an excellent choice when testing the efficiency of an *RDF database system*. However, they will not be suitable for experimentation and analysis of a particular *RDF application*. Specifically, since these datasets are built for a single given scenario, they may not have either of the two specified properties, for the application at hand.

* This work was partially supported by the GIF (Grant 2201-1880.6/2008) and the ISF (Grant 143/09).

Data generators are another source for datasets. A data generator is a program that generates data according to user constraints. As such, data generators are usually more flexible than benchmarks. Unfortunately, there are few data generators available for RDF (SIMILE [1], RBench[2]) and none of these programs can produce data that conforms to a specific given structure, and thus, again, will not have the specified properties.

In this paper, we present the GRR system for generating RDF that satisfies both desirable properties given above. Thus, GRR is *not* a benchmark system, but rather, a system to use for Semantic Web application testing.[3] GRR can produce data with a complex graph structure, as well as draw the data values from desirable domains. As a motivating (and running) example, we will discuss the problem of generating the data described in the LUBM Benchmark. However, GRR is not limited to creating benchmark data. For example, we also demonstrate using GRR to create FOAF [9] data (Friend-of-a-friend, social network data) in our experimentation.

Example 1. LUBM [10] is a collection of data describing university classes (i.e., entities), such as departments, faculty members, students, etc. These classes have a plethora of properties (i.e., relations) between them, e.g., faculty members work for departments and head departments, students take courses and are advised by faculty members.

In order to capture a real-world scenario, LUBM defines interdependencies of different types between the various entities. For example, the number of students in a department is a function of the number of faculty members. Specifically, LUBM requires there to be a 1:8-14 ratio of faculty members to undergraduate students. As another example, the cardinality of a property may be specified, such as each department must have a single head of department (who must be a full professor). Properties may also be required to satisfy additional constraints, e.g., courses, taught by faculty members, must be pairwise disjoint. □

Our main challenge is to provide powerful generation capabilities, while still retaining a simple enough user interface, to allow the system to be easily used. To demonstrate the capabilities of GRR, we show that GRR can easily be used to reproduce the entire LUBM benchmark. Actually, our textual commands are not much longer than the intuitive description of the LUBM benchmark! We are interested in generating very large datasets. Thus, a second challenge is to ensure that the runtime remains reasonable, and that data generation scales up well. Several optimization techniques have been employed to improve the runtime, which make GRR efficient and scalable. We note that a short demo of this work appeared in [6].

2 Abstract Generation Language

In this section we present the abstract syntax and the semantics for our data generation language. Our data generation commands are applied to a (possibly empty) label graph,

[1] `http://simile.mit.edu/`

[2] `http://139.91.183.30:9090/RDF/RBench`

[3] Our focus is on batch generation of data for testing, and not on online testing, where data generation is influenced by application responses.

to augment it with additional nodes and edges. There are four basic building blocks for a data generation command. *First*, a *composite query* $\bar{Q}$ finds portions of a label graph to augment. *Second*, a *construction command* C defines new nodes and edges to create. *Third*, a *query result sampler* defines which results of $\bar{Q}$ should be used for input to C. *Fourth*, a *natural number sampler* determines the number of times that C is applied. Each of these components are explained below.

Label Graphs. This paper presents a method of generating random graphs of RDF data. As such, graphs figure prominently in this paper, first as the target output, and second, as an integral part of the generation commands.

A *label graph* $L = (V, E)$ is a pair, where V is a set of labeled nodes and E is a set of directed, labeled edges. We use $a, b, \ldots$ to denote nodes of a label graph. Note that there can be multiple edges connecting a pair of nodes, each with a different label. Label graphs can be seen as an abstract data model for RDF.

Queries. GRR uses queries as a means for specifying data generation commands. We will apply queries to label graphs, to get sets of mappings for the output variables of the queries. The precise syntax used for queries is not of importance, and thus we only describe the abstract notation. In our implementation, queries are written in SPARQL[4].

A *query* is denoted $Q(\bar{x}; \bar{y})$ where $\bar{x}$ and $\bar{y}$ are tuples of distinct variables, called the *input variables* and *output variables*, respectively. We use $|\bar{x}|$ to denote the size of the tuple $\bar{x}$, i.e., if $\bar{x} = x_1, \ldots x_k$, then $|\bar{x}| = k$. If $\bar{x}$ is an empty tuple, then we write the query as $Q(; \bar{y})$. When no confusion can arise, we will sometimes simply write Q to denote a query.

Let L be a label graph and $\bar{a}$ be a tuple of (possibly nondistinct) nodes from L, of size equal to that of the input variables $\bar{x}$. Then, applying $Q(\bar{a}; \bar{y})$ to L results in a set $\mathcal{M}_{\bar{a}}(Q, L)$ of mappings μ from the variables in $\bar{y}$ to nodes in L. Intuitively, each mapping represents a query answer, i.e., $\mu(\bar{y})$ is a tuple in the result.

A *composite query* has the form $\bar{Q} := Q_1(; \bar{y}_1), \ldots, Q_n(\bar{x}_n; \bar{y}_n)$, where

1. all queries have different output variables $\bar{y}_i$;
2. all input variables appear as output variables in an earlier query, i.e., for all $1 < i \leq n$, and for all $x \in \bar{x}_i$, there is a $j < i$ such that $x \in \bar{y}_j$.

As a special case, we allow $\bar{Q}$ to be an empty tuple of queries, also written as $\top$.

Composite queries are used to find portions of a label graph that should be augmented by the construction stage. Note that the second requirement ensures that Q_1 will have no input variables. As explained later, queries are evaluated in a nested-loop-like fashion, where answers to $Q_1, \ldots, Q_i$ are used to provide the input to Q_{i+1}.

Intuitively, there are two types of augmentation that will be performed using the results of a composite query, during the construction stage. First, the new structure (nodes and edges) must be specified. Second, the labels of this new structure (subjects, predicates, objects) must be defined. Both of these aspects of data generation are discussed next. To ease the presentation, we start with the latter type of augmentation (randomly

[4] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query

choosing labels, using value samplers), and then discuss the former augmentation (creating new structures, using construction patterns), later on.

Value Samplers. GRR uses user-specified sampling procedures to produce random data. Value samplers, called *v-samplers* for short, are used to generate labels of an RDF graph. Using GRR, users can provide two different types of v-samplers (data and class value samplers), to guide the generation process.

A *data value sampler* returns a value from a space of values. For example, a data value sampler can randomly choose a country from a dictionary (file) of country names. In this fashion, a data value sampler can be used to provide the literal values associated with the nodes of an RDF graph. A data value sampler can also be defined in a deterministic manner, e.g., so that is always returns the same value. This is particularly useful for defining the predicates (edge labels) in an RDF graph.

When attempting to produce data conforming to a given schema, we will be familiar with the literal properties expected to be present for each class. To allow the user to succinctly request the generation of an instance, along with all its literal properties, GRR uses *class value samplers*. Class value samplers can be seen as an extended type of data value sampler. While a data value sampler returns a single value, a class value sampler returns a value for each of the literal properties associated with a given class.

Example 2. In order to produce data conforming to the LUBM schema, many *data value samplers* are needed. Each data value sampler will associate a different type of node or edge with a value. For example, to produce the age of a faculty member, a data value sampler choosing an age within a given range can be provided. To produce RDF identifiers of faculty members, a data value sampler producing a specific string, ended by a running number (e.g., `http://www.Department.Univ/FullProf14`) can be provided. Finally, to produce the predicates of an RDF graph, constant data value samplers can be used, e.g., to produce the edge label `ub:worksFor`, a data sampler which always returns this value can be provided.

A *class value sampler* for faculty, provides methods of producing values for all of the literal properties of faculty, e.g., age, email, officeNumber, telephone. □

Construction Patterns. *Construction patterns* are used to define the nodes and edges that will be generated, and are denoted as $C = (\bar{x}, \bar{z}, \bar{e}, \Pi)$, where:

– $\bar{x}$ is a tuple of *input variables*;
– $\bar{z}$ is a tuple of *construction variables*;
– $\bar{e} \subseteq (\bar{x} \uplus \bar{z}) \times (\bar{x} \uplus \bar{z})$ is a tuple of *construction edges*;
– Π, called the *sampler function*, associates each construction variable and each construction edge with a v-sampler.

We use $\bar{x} \uplus \bar{z}$ to denote the set of variables appearing in $\bar{x}$ or in $\bar{z}$.

Construction patterns are used to extend a given label graph. Specifically, let $C = (\bar{x}, \bar{z}, \bar{e}, \Pi)$ be a construction pattern, $L = (V, E)$ be a label graph and $\mu : \bar{x} \to V$ be a mapping of $\bar{x}$ to V. The *result of applying* C to L given μ, denoted $C \Downarrow_{\mu} L$, is the label graph derived by the following two stage process:

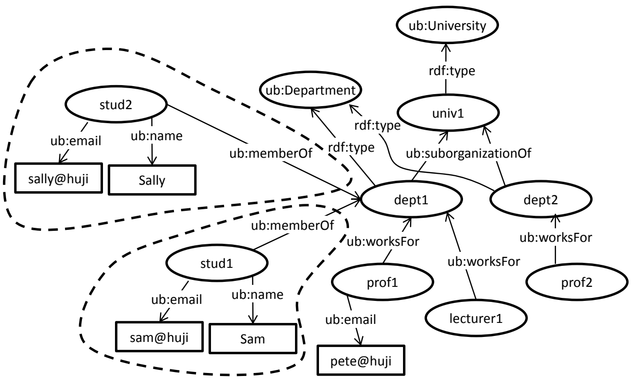


Fig. 1. Partial RDF data graph

– **Adding New Nodes:** For each variable $z \in \bar{z}$, we add a new node a to L. If $\Pi(z)$ is a data value sampler, then the label of a is randomly drawn from $\Pi(z)$. If $\Pi(z)$ is a class value sampler, then we also choose and add all literal properties associated with z by $\Pi(z)$. Define $\mu'(z) = a$.
When this stage is over, every node in $\bar{z}$ is mapped to a new node via μ'. We define μ'' as the union of μ and μ'.
– **Adding New Edges:** For each edge $(u, v) \in \bar{e}$, we add a new edge $(\mu''(u), \mu''(v))$ to L. The label of $(\mu''(u), \mu''(v))$ is chosen to as a random sample drawn from $\Pi(u, v)$.

A construction pattern can be applied several times, given the same mapping μ. Each application results in additional new nodes and edges.

Example 3. Consider[5] $C = ((?\texttt{dept}), (?\texttt{stud}), \{(?\texttt{stud}, ?\texttt{dept})\}, \Pi)$, where Π assigns `?stud` a class value sampler for students, and assigns the edge $(?\texttt{stud}, ?\texttt{dept})$ with a data value sampler that returns the constant `ub:memberOf`. Furthermore, suppose that the only literal values a student has are his name and email.

Figure 1 contains a partial label graph L of RDF data. Literal values appear in rectangles. To make the example smaller, we have omitted many of the literal properties, as well as much of the typing information (e.g., `prof1` is of type `ub:FullProfessor`, but this does not appear).

Now, consider L^-, the label graph L, without the parts appearing in the dotted circles. Let μ be the mapping which assigns input variable `?dept` to the node labeled `dept1`. Then each application of $C \Downarrow_\mu L^-$ can produce one of the circled structures. To see this, observe that each circled structure contains a new node, e.g., `stud1`, whose literal properties are added by the class value structure. The connecting edge, i.e., (`stud1`, `dept1`), will be added due to the presence of the corresponding construction edge in C, and its label `ub:memberOf` is determined by the constant data value sampler. □

[5] We use the SPARQL convention and prepend variable names with `?`.

Algorithm $\mathsf{Apply}(\bar{Q}, \bar{\pi}_q, C, \pi_n, L, L^*, \mu, j)$

```
1: if $j > |\bar{Q}|$ then
2:     choose $n$ from $\pi_n$
3:     for $i \leftarrow 1$ to $n$ do
4:         $L^* \leftarrow C \Downarrow_\mu L^*$
5: else
6:     $\bar{a} = \mu(\bar{x}_j)$
7:     for all $\mu'$ chosen by $\pi_q^j$ from $\mathcal{M}_{\bar{a}}(Q_j, L)$ do
8:         $L^* \leftarrow \mathsf{Apply}(\bar{Q}, \bar{\pi}_q, C, \pi_n, L, , L^*, \mu \cup \mu', j+1)$
9: return $L^*$
```

Fig. 2. Applying a data generation command to a label graph

Query Result Samplers. As stated earlier, there are four basic building blocks for a data generation command: composite queries, construction commands, query result samplers and natural number samplers. The first two items have been defined already. The last item, a *natural number sampler* or *n-sampler*, for short, is simply a function that returns a non-negative natural number. For example, the function π_i that always returns the number i, $\pi_{[i,j]}$ that uniformly returns a number between i and j, and $\pi_{m,v}$ which returns a value using the normal distribution, given a mean and variance, are all special types of natural number samplers.

We now define the final remaining component in our language, i.e., query result samplers. A query, along with a label graph, and an assignment μ of values for the input nodes, defines a set of assignments for the output nodes. As mentioned earlier, the results of a query guide the generation process. However, we will sometimes desire to choose a (random) subset of the results, to be used in data generation. A *query result sampler* is provided precisely for this purpose.

Given (1) a label graph L, (2) a query Q and (3) a tuple of nodes $\bar{a}$ for the input variables of Q, a query result sampler, or *q-sampler*, chooses mappings in $\mathcal{M}_{\bar{a}}(Q, L)$. Thus, applying a q-sampler π_q to $\mathcal{M}_{\bar{a}}(Q, L)$ results in a series $\mu_1, \ldots, \mu_k$ of mappings from $\mathcal{M}_{\bar{a}}(Q, L)$. A q-sampler determines both the length k of this series of samples (i.e., how many mappings are returned) and whether the sampling is with, or without, repetition.

Example 4. Consider a query Q_1 (having no input values) that returns department nodes from a label graph. Consider label graph L from Figure 1. Observe that $\mathcal{M}(Q_1, L)$ contains two mappings: μ_1 which maps `?dept` to the node labeled `dept1` and μ_2 which maps `?dept` to the node labeled `dept2`.

A q-sampler π_q is used to derive a series of mappings from $\mathcal{M}(Q_1, L)$. The q-sampler π_q can be defined to return each mapping once (e.g., the series μ_1, μ_2 or μ_2, μ_1), or to return a single random mapping (e.g., μ_2 or μ_1), or to return two random choices of mappings (e.g., one of μ_1, μ_1 or μ_1, μ_2 or μ_2, μ_1 or μ_2, μ_2), or can be defined in countless other ways. Note that regardless of how π_q is defined, a q-sampler always returns a series of mappings. The precise definition of π_q defines properties of this series (i.e., its length and repetitions). □

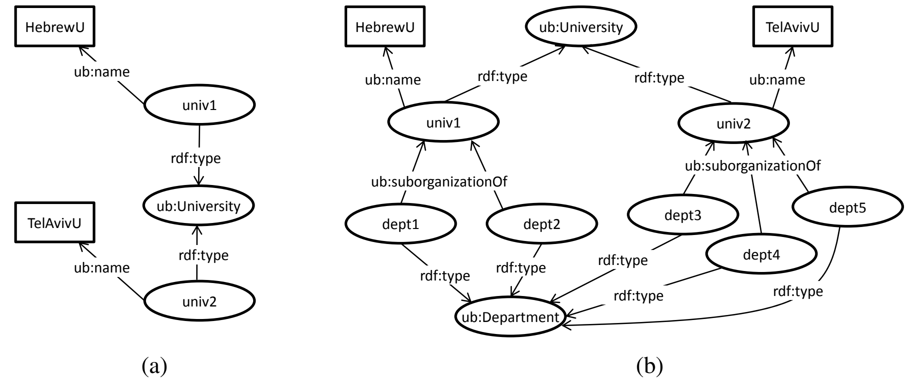


Fig. 3. (a) Possible result of application of $\mathcal{C}_1$ to the empty label graph. (b) Possible result of application of $\mathcal{C}_2$ to (a).

Data Generation Commands. To generate data, the user provides a series of *data generation commands*, each of which is a 4-tuple $\mathcal{C} = (\bar{Q}, \bar{\pi}_q, C, \pi_n)$, where

- $\bar{Q} = Q_1(\bar{x}_1; \bar{y}_1), \ldots, Q_k(\bar{x}_k, \bar{y}_k)$ is a composite query;
- $\bar{\pi}_q = (\pi_q^1, \ldots, \pi_q^k)$ is a tuple of q-samplers;
- $C = (\bar{x}, \bar{z}, \bar{e}, \mathit{\Pi})$ is a construction pattern *and*
- π_n is an n-sampler.

In addition, we require every input variable in tuple $\bar{x}$ of the construction pattern C to appear among the output variables in $\bar{Q}$.

Algorithm Apply (Figure 2) applies a data generation command $\mathcal{C}$ to a (possibly empty) label graph L. It is called with $\mathcal{C}$, L, $L^* = L$, the empty mapping $\mu_\emptyset$ and $j = 1$. Intuitively, Apply runs in a recursive fashion, described next.

We start with query Q_1, which cannot have any input variables (by definition). Therefore, Line 6 is well-defined and assigns $\bar{a}$ the empty tuple $()$. Then, we choose mappings from the result of applying Q_1 to L, using the q-sampler π_q^1. For each of these mappings μ', we recursively call Apply, now with the extended mapping $\mu \cup \mu'$ and with the index 2, which will cause us to consider Q_2 within the recursive application.

When we reach some $1 < j \leq |\bar{Q}|$, the algorithm has a mapping μ which assigns values for all output variables of Q_i, for $i < j$. Therefore, μ assigns a value to each of its input variables $\bar{x}_j$. Given this assignment for the input variables, we call Q_j, and choose some of the results in its output. This process continues recursively, until we reach $j = |\bar{Q}| + 1$.

When $j = |\bar{Q}| + 1$, the mapping μ must assign values for all of the input variables of C. We then choose a number n using the n-sampler π_n, and apply the construction pattern C to L^* a total of n times. Note that at this point, we recurse back, and may eventually return to $j = |\bar{Q}| + 1$ with a different mapping μ.

We note that Apply takes two label graphs L and L^* as parameters. Initially, these are the same. Throughout the algorithm, we use L to evaluate queries, and actually apply the construction patterns to L^* (i.e., the actual changes are made to L^*). This is important from two aspects. First, use of L^* allows us to make sure that all constructions are made to the same graph, which eventually returns a graph containing all new additions. Second, and more importantly, since we only apply queries to L, the end result is well-defined. Specifically, this means that nodes constructed during the recursive calls where $j = |\bar{Q}| + 1$, cannot be returned from queries applied when $j \leq |\bar{Q}|$. This makes the algorithm insensitive to the particular order in which we iterate over mappings in Line 7. Maintaining a copy L^* of L is costly. Hence, in practice, GRR avoids copying of L, by deferring all updates until the processing of the data generation command has reached the end. Only after no more queries will be issued are all updates performed.

Example 5. Let C_1 be the construction pattern containing no input variables, a single construction variable `?university`, no construction edges, and Π associating `?university` with the class value sampler for university. Let $\mathcal{C}_1$ be the data generation command containing the empty composite query $\top$, the empty tuple of q-samplers, the construction command C_1 and an n-sampler $\pi_{[1,5]}$ which uniformly chooses a number between 1 and 5. Since C_1 has no input variables (and $\bar{Q} = \top$), the data generation command $\mathcal{C}_1$ can be applied to an empty label graph. Assuming that universities have a single literal property `name`, applying $\mathcal{C}_1$ to the empty graph can result in the label graph appearing in Figure 3(a).

As a more complex example, let C_2 be the construction pattern containing input variable `?univ`, a single construction variable `?dept`, a construction edge (`?dept`, `?univ`), and Π associating `?dept` with the class value sampler for department. Let $\mathcal{C}_2$ be the data generation command containing (1) a query selecting nodes of type university (2) a q-sampler returning all mappings in the query result, (3) the construction pattern C_2 and (4) the n-sampler $\pi_{[2,4]}$. Then, applying $\mathcal{C}_2$ to the label graph of Figure 3(a) can result in the label graph appearing in Figure 3(b). (Literal properties of departments were omitted due to space limitations.)

Finally, the construction pattern described in Example 3, and the appropriately defined data generation command using this construction pattern (e.g., which has a query returning department mappings) could add the circled components of the graph in Figure 1 to the label graph of Figure 3(b). □

3 Concrete Generation Language

The heart of the GRR system is a Java implementation of our abstract language, which interacts with an RDF database for both evaluation of the SPARQL composite queries, and for construction and storage of the RDF data. The user can provide the data generation commands (i.e., all details of the components of our abstract commands) within an RDF file. Such files tend to be quite lengthy, and rather arduous to define.

To make data generation simple and intuitive, we provide the user with a simpler textual language within which all components of data generation commands can be defined. Our textual input is compiled into the RDF input format. Thus, the user has the flexibility of using textual commands whenever they are expressive enough for his needs, and augmenting the RDF file created with additional (more expressive) commands, when needed. We note that the textual language is quite expressive, and therefore we believe that such augmentations will be rare. For example, commands to recreate the entire LUBM benchmark were easily written in the textual interface.

The underlying assumption of the textual interface is that the user is interested in creating instances of classes and connecting between these, as opposed to arbitrary construction of nodes and edges. To further simplify the language, we allow users to use class names instead of variables, for queries that do not have self-joins. This simplification allows many data generation commands to be written without the explicit use of variables, which makes the syntax more succinct and readable. The general syntax of a textual data generation command appears below.

```
1: (FOR query sampling method
2:   [WITH (GLOBAL DISTINCT|LOCAL DISTINCT|REPEATABLE)]
3:   {list of classes}
4:   [WHERE {list of conditions}])*
5: [CREATE  n-sampler {list of classes}]
6: [CONNECT {list of connections}]
```

Observe that a data generation command can contain three types of clauses: `FOR`, `CREATE` and `CONNECT`. There can be any number (zero or more) `FOR` clauses. Each `FOR` clause defines a q-sampler (Lines 1–2), and a query (Lines 3-4). The `CREATE` and `CONNECT` clauses together determine the n-sampler (i.e., the number of times that the construction pattern will be applied), and the construction pattern. Each of `CREATE` and `CONNECT` is optional, but at least one among them must appear.

We present several example commands that demonstrate the capabilities of the language. These examples were chosen from among the commands needed to recreate the LUBM benchmark data.[6]

```
(C1) CREATE 1-5 {ub:Univ}
(C2) FOR EACH {ub:Univ}
       CREATE 15-25 {ub:Dept}
       CONNECT {ub:Dept ub:subOrg ub:Univ}
(C3) FOR EACH {ub:Faculty, ub:Dept}
       WHERE {ub:Faculty ub:worksFor ub:Dept}
       CREATE 8-14 {ub:Undergrad}
       CONNECT {ub:Undergrad ub:memberOf ub:Dept}
(C4) FOR EACH {ub:Dept}
       FOR 1 {ub:FullProf}
       WHERE {ub:FullProf ub:worksFor ub:Dept}
       CONNECT {ub:FullProf ub:headOf ub:Dept}
(C5) FOR 20%-20% {ub:Undergrad, ub:Dept}
       WHERE {ub:Undergrad ub:memberOf ub:Dept}
         FOR 1 {ub:Prof}
         WHERE {ub:Prof ub:memberOf ub:Dept}
         CONNECT {ub:Undergrad ub:advisor ub:Prof}
```

[6] In our examples, we shorten the class and predicates names to make the presentation shorter, e.g., using `Dept` instead of `Department`, and omit explicit namespace definition.

```
(C6) FOR EACH {ub:Undergrad}
        FOR 2-4 WITH LOCAL DISTINCT {ub:Course}
        CONNECT {ub:Undergrad ub:takeCourse ub:Course}
```

We start with a simple example demonstrating a command with only a CREATE clause. Command $\mathcal{C}_1$ generates instances of universities (precisely as $\mathcal{C}_1$ in Example 5 did). Note that $\mathcal{C}_1$ contains no FOR clause (and thus, it corresponds to a data generation command with the $\top$ composite query, and can be applied to an empty graph) and contains no CONNECT clause (and thus, it does not connect the instances created).

Next, we demonstrate commands containing all clauses, as well as the translation process from the textual commands to SPARQL. Command $\mathcal{C}_2$ creates between 15 and 25 departments per university. Each of the departments created for a particular university are connected to that university (precisely as with $\mathcal{C}_2$ in Example 5). Similarly, $\mathcal{C}_3$ creates 8 to 14 undergraduate students per faculty member per department, thereby ensuring a 1:8-14 ratio of faculty members to undergraduate students, as required in LUBM (Example 1).

The FOR clause of a command implicitly defines a SPARQL query Q, as follows. The select clause of Q contains a distinct variable ?vC, for each class name C in the class list. The where clause of Q is created by taking the list of conditions provided and performing two steps. First, any class names already appearing in the list of conditions is replaced by its variable name. Second, the list of conditions is augmented by adding a triple ?vC rdf:typeOf C for each class C appearing in the class list.

For $\mathcal{C}_2$ and $\mathcal{C}_3$, this translation process will result in the SPARQL queries Q_2 and Q_3, respectively, which will be applied to the database to find mappings for our construction patterns.

```
(Q2) SELECT ?univ
     WHERE {?univ rdf:typeOf ub:University .}
(Q3) SELECT ?f, ?d
     WHERE  {?f ub:worksFor ?d. ?f rdf:typeOf ub:Faculty.
             ?d rdf:typeOf ub:Dept.}
```

So far, all examples have used a single FOR clause. We show now that larger composite queries are useful. The LUBM benchmark specifies that, for each department, there is a single full professor who is the head of the department. In $\mathcal{C}_4$ two FOR clauses are used: the first, to loop over all departments, and the second, to choose one full professor working for the current department. The CONNECT clause adds the desired predicate (edge) between the department and full professor. Note that the output variable of the first FOR clause is an input variable in the second (it is mentioned in the where condition). In addition, the output variables of both clauses are input variables of the construction pattern, as defined by the CONNECT clause.

As an additional example, a fifth of the undergraduate students should have a professor as their advisor, and thus, the first FOR clause in $\mathcal{C}_5$ loops over 20% of the undergraduates (with their department), and the second FOR clause chooses a professor in the same department to serve as an advisor.

Next, we demonstrate the different repetition modes available, and how they influence the end result, i.e., the data graph created. As specified in the LUBM benchmark, each undergraduate student takes 2–4 undergraduate courses. In $\mathcal{C}_6$, the first FOR clause loops over all undergraduate students. The second FOR clause chooses 2–4 undergraduate courses for each student.

Our interface can be used to define query samplers with three different repetition modes (Line 2). In *repeatable* mode, the same query results can be sampled multiple times. Using this mode in $\mathcal{C}_6$ would allow the same course to be sampled multiple times for a given student, yielding several connections of a student to the same course. In *global distinct* mode, the same query result is *never* returned more than once, even for different samples of query results in previous FOR clauses. Using global distinct in $\mathcal{C}_6$ would ensure that the q-sampler of the second FOR clause never returns the same course, even when called for different undergraduates returned by the first FOR clause. Hence, no two students would take a common course. Finally, in *local distinct* mode, the same query result can be repeatedly returned only for different results of previous FOR clauses. However, given specific results of all previous FOR clauses, query results will not be repeated. Thus, for each undergraduate student, we will sample 2–4 *different* undergraduate courses. However for different undergraduate students we may sample the *same* courses, i.e., several students may study the same course, as is natural.

As a final example, we show how variables can be used to express queries with self-joins (i.e., using several instances of the same class). This example connects people within a FOAF RDF dataset. Note the use of FILTER in the WHERE clause, which is naturally included in our language, since our WHERE clause is immediately translated into the WHERE clause of a SPARQL query.

```
FOR EACH {foaf:Person ?p1}
  FOR 15-25 {foaf:Person ?p2}
  WHERE {FILTER ?p1 != p2}
  CONNECT {?p1 foaf:knows ?p2}
```

Finally, we must explain how the data value samplers and class value samplers are defined. An optional *class mappings* input file provides a mapping of each class to the literal properties that it has, e.g., the line

```
ub:GraduateStudent; ub:name; ub:email; ub:age;
```

associates GraduateStudents with three properties. An additional *sampler function* file states which value samplers should be used for each literal property, e.g., the line

```
CounterDictSampler; GlobalDistinctMode; ub:Person; Person
```

states that the literals which identify a Person should never be repeated, and should be created by appending Person with a running counter.

4 Optimization Techniques

Each evaluation of a data generation command requires two types of interactions with an RDF database: query evaluation of the composite queries (possibly many times), and the addition of new triples to the RDF database, as the construction patterns are applied. Thus, the OLTP speed of the RDF database used will greatly dictate the speed in which our commands can be executed. Two strategies were used to decrease evaluation time.

Caching Query Results. A data generation command has a composite query $\bar{Q} = Q_1(\bar{x}_1;\bar{y}_1),\ldots,Q_k(\bar{x}_k;\bar{y}_k)$ and a tuple of q-samplers, $\pi_q^1,\ldots,\pi_q^k$. During the execution of a data generation command, a specific query Q_i may be called several times. Specifically, a naive implementation will evaluate Q_i once, for each sample chosen by π_q^j from the result of Q_j, for all $j < i$. For example, the query determined by the second `FOR` clause of $\mathcal{C}_6$ (Section 3), returning all undergraduate courses, will be called repeatedly, once for each undergraduate student.

Sometimes, repeated evaluations of Q_i cannot be avoided. This occurs if the input parameters $\bar{x}_i$ have been given new values. However, repeated computations with the same input values can be avoided by using a caching technique. We use a hash-table to store the output of a query, for each different instantiation of the input parameters. Then, before evaluating Q_i with input values $\bar{a}$ for $\bar{x}_i$, we check whether the result of this query already appears in the hash table. If so, we use the cached values. If not, we evaluate Q_i, and add its result to the hash-table.

Avoiding Unnecessary Caching. Caching is effective in reducing the number of times queries will be applied. However, caching query results incurs a significant storage overhead. In addition, there are cases in which caching does not bring any benefit, since some queries will never be called repeatedly with the same input values.

To demonstrate, consider $\mathcal{C}_4$ (Section 3), which chooses a head for each department. As the q-sampler for the query defined by the first `FOR` clause iterates over all departments without repetition, the query defined by the second `FOR` clause will always be called with different values for its input parameter `ub:Dept`. Hence, caching the results of the second query is not useful.

In GRR, we avoid caching of results when caching is guaranteed to be useless. In particular, we will not cache the results of Q_i if, for all $j < i$ we have (1) Q_j runs in global distinct mode *and* (2) all output parameters of Q_j are input parameters of Q_i. As a special case, this also implies that caching is not employed in $\mathcal{C}_4$, but will be used for $\mathcal{C}_5$ (as only `Dept`, and not `Undergrad`, is an input parameter to the query defined by its second `FOR` clause). Note that our caching technique is in the spirit of result memoization, a well-known query optimization technique (e.g., [12,2]).

5 Experimentation

We implemented GRR within the Jena Semantic Web Framework for Java.[7] Our experimentation uses Jena's TDB database implementation (a high performance, pure-Java, non-SQL storage system). All experiments were carried out on a personal computer running Windows Vista (64 bit) with 4GB of RAM.

We created two types of data sets using GRR. First, we recreated the LUBM benchmark, with various scaling factors. Second, we created simple RDF data conforming to the FOAF (friend of a friend) schema [9]. The full input provided to GRR for these

[7] Jena–A Semantic Web Framework for Java, `http://jena.sourceforge.net`

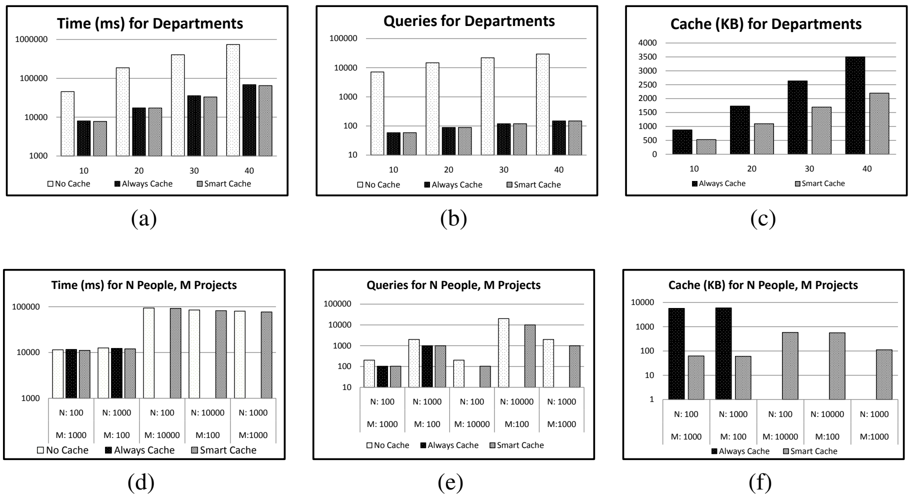


Fig. 4. Time, number of queries and cache size for generating data

experiments, as well as working downloadable code for GRR, is available online.[8] Our goal is to determine the scalability of GRR in terms of runtime and memory, as well as to (at least anecdotally) determine its ease of use.

LUBM Benchmark. In our first experiment we recreated the LUBM benchmark data. All data could be created directly using the textual interface. In total, 24 data generation commands were needed. These commands together consisted of only 284 words. This is approximately 1.8 times the number of words used in the intuitive description of the LUBM benchmark data (consisting of 158 words), which is provided in the LUBM project for users to read. We found the data generation commands to be intuitive, as demonstrated in the examples of Section 3. Thus, anecdotally, we found GRR to be quite easy to use.

Together with the class mappings and sampler function files (details not discussed in the LUBM intuitive description), 415 words were needed to recreate the LUBM benchmark. This compares quite positively to the 2644 words needed to recreate LUBM when writing these commands directly in RDF—i.e., the result of translating the textual interface into a complete RDF specification.

We consider the performance of GRR. The number of instantiations of each class type in LUBM is proportional to the number of departments.[9] We used GRR to generate the benchmark using different numbers of departments. We ran three versions of GRR to determine the effect of our optimizations. In NoCache, no caching of query results was performed. In AlwaysCache, all query results were cached. Finally, in SmartCache, only query results that can potentially be reused (as described in Section 4) are stored. Each experiment was run three times, and the average runtime was taken.

[8] www.cs.huji.ac.il/~danieb12

[9] LUBM suggests 12-25 departments in a university.

The runtime appears in Figure 4(a). As can be seen in this figure, the runtime increases linearly as the scaling factor grows. This is true for all three versions of GRR. Both AlwaysCache and SmartCache significantly outperform NoCache, and both have similar runtime. For the former two, the runtime is quite reasonable, with construction taking approximately 1 minute for the 200,000 tuples generated when 40 departments are created, while NoCache requires over 12 minutes to generate this data.

The runtime of Figure 4(a) is easily explained by the graph of Figure 4(b), which depicts the number of queries applied to the database. Since SmartCache takes a conservative approach to caching, it always caches results that can potentially be useful. Thus, AlwaysCache and SmartCache apply the same number of queries to the database. NoCache does not store any query results, and hence, must make significantly more calls to the database, which degrades its runtime.[10]

Finally, Figure 4(c) shows the size of the cache for GRR. NoCache is omitted, as is does not use a cache at all. To measure the cache size, we serialized the cache after each data generation command, and measured the size of the resulting file. The maximum cache size generated (among the 24 data generation commands) for each of AlwaysCache and SmartCache, and for different numbers of departments, is shown in the figure. Clearly, SmartCache significantly reduces the cache size.

FOAF Data. In this experiment we specifically chose data generation commands that would allow us to measure the effectiveness of our optimizations. Therefore, we used the following (rather contrived) data generation commands, with various values for M and N.

```
(C1) CREATE N {foaf:Person}
(C2) CREATE M {foaf:Project}
(C3) FOR EACH {foaf:Person}
    FOR EACH {foaf:Project}
      CONNECT {foaf:Person foaf:currProject foaf:Project}
(C4) FOR EACH {foaf:Person}
    FOR EACH {foaf:Project}
    WHERE {foaf:Person foaf:currProject foaf:Project}
      CONNECT {foaf:Person foaf:maker foaf:Project}
```

Commands $\mathcal{C}_1$ and $\mathcal{C}_2$ create N people and M projects. Command $\mathcal{C}_3$ connects all people to all projects using the `currProject` predicate, and $\mathcal{C}_4$ adds an additional edge `maker` between each person and each of his projects. Observe that caching is useful for $\mathcal{C}_3$, as the inner query has no input parameters, but not for $\mathcal{C}_4$, as it always has a different value for its input parameters.[11]

Figures 4(d) through 4(f) show the runtime, number of query applications to the RDF database and memory size, for different parameters of N and M. Note that approximately 200,000 triples are created for the first two cases, and 2 million for the

[10] This figure does not include the number of construction commands applied to the database, which is equal, for all three versions.

[11] This example is rather contrived, as a simpler method to achieve the same effect is to include both `CONNECT` clauses within $\mathcal{C}_3$.

last three cases. For the last three cases, statistics are missing for AlwaysCache, as its large cache size caused it to crash due to lack of memory. Observe that our system easily scales up to creating 2 million tuples in approximately 1.5 minutes. Interestingly, SmartCache and NoCache perform similarly in this case, as the bulk of the time is spent iterating over the instances and constructing new triples.

We conclude that SmartCache is an excellent choice for GRR, due to its speed and reasonable cache memory requirements.

6 Conclusion

We presented the GRR system for generating random RDF data. GRR is unique in that it can create data with arbitrary structure (as opposed to benchmarks, which provide data with a single specific structure). Thus, GRR is useful for generating test data for Semantic Web applications. By abstracting SPARQL queries, GRR presents a method to create data that is both natural and powerful.

Future work includes extending GRR to allow for easy generation of other types of data. For example, we are considering adding recursion to the language to add an additional level of power. User testing, to prove the simplicity of use, is also of interest.

References

1. Aboulnaga, A., Naughton, J., Zhang, C.: Generating synthetic complex-structured XML data. In: WebDB (2001)
2. Bancilhon, F., Ramakrishnan, R.: An amateur's introduction to recursive query processing strategies. In: SIGMOD, pp. 16–52 (1986)
3. Barbosa, D., Mendelzon, A., Keenleyside, J., Lyons, K.: ToXgene: an extensible template-based data generator for XML. In: WebDB (2002)
4. Binnig, C., Kossman, D., Lo, E.: Testing database applications. In: SIGMOD (2006)
5. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. IJSWIS 5(2), 1–24 (2009)
6. Blum, D., Cohen, S.: Generating RDF for application testing. In: ISWC (2010)
7. Bruno, N., Chaudhuri, S.: Flexible database generators. In: VLDB (2005)
8. Cohen, S.: Generating XML structure using examples and constraints. PVLDB 1(1), 490–501 (2008)
9. The friend of a friend (foaf) project, http://www.foaf-project.org
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. JWS 3(2-3), 158–182 (2005)
11. Houkjaer, K., Torp, K., Wind, R.: Simple and realistic data generation. In: VLDB (2006)
12. McKay, D.P., Shapiro, S.C.: Using active connection graphs for reasoning with recursive rules. In: IJCAI, pp. 368–374 (1981)
13. Neufeld, A., Moerkotte, G., Lockemann, P.C.: Generating consistent test data for a variable set of general consistency constraints. The VLDB Journal 2(2), 173–213 (1993)
14. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP^{2}Bench: a SPARQL performance benchmark. In: ICDE (March 2009)

High-Performance Computing Applied to Semantic Databases

Eric L. Goodman[1], Edward Jimenez[1], David Mizell[2], Sinan al-Saffar[3], Bob Adolf[3], and David Haglin[3]

[1] Sandia National Laboratories, Albuquerque, NM, USA
{elgoodm,esjimen}@sandia.gov
[2] Cray, Inc., Seattle, WA, USA
dmizell@cray.com
[3] Pacific Northwest National Laboratory, Richland, WA, USA
{sinan.al-saffar,robert.adolf,david.haglin}@pnl.gov

Abstract. To-date, the application of high-performance computing resources to Semantic Web data has largely focused on commodity hardware and distributed memory platforms. In this paper we make the case that more specialized hardware can offer superior scaling and close to an order of magnitude improvement in performance. In particular we examine the Cray XMT. Its key characteristics, a large, global shared-memory, and processors with a memory-latency tolerant design, offer an environment conducive to programming for the Semantic Web and have engendered results that far surpass current state of the art. We examine three fundamental pieces requisite for a fully functioning semantic database: dictionary encoding, RDFS inference, and query processing. We show scaling up to 512 processors (the largest configuration we had available), and the ability to process 20 billion triples completely in-memory.

Keywords: Semantic Web, RDFS Inference, Cray XMT, Dictionary Encoding, SPARQL, graph databases.

1 Introduction

The Semantic Web is a loosely defined notion, but generally includes such standards as the

- Resource Description Framework (RDF), a mechanism for describing entities and relationships between entities,
- RDF Schema (RDFS) and the Web Ontology Language (OWL), which provide the ability to describe ontologies that can be applied to RDF data stores,
- various data interchange formats such as RDF/XML and N-Triples, and
- SPARQL, a query language for retrieving results from RDF data sets.

An RDF statement consists of subject-predicate-object expressions known as *triples*. RDFS and OWL can be applied to triple stores to infer new facts from existing statements. This inferencing can be done at runtime for a particular query, or it can be done in batch, essentially materializing the new triples all at once in a process called *closure*.

There has been some work in taking these technologies and scaling them to data sizes on the order of a billion triples or in some cases 100 billion triples. In terms of RDFS and OWL inferencing, Urbani et al. [6] perform RDFS closure on an RDF data set gathered from various sources on the web, and then later expand to a fragment of OWL reasoning on data sets ranging up to 100 billion triples [7]. They utilize a distributed cluster with MapReduce as the programming paradigm. Weaver et al. [9] again use a distributed cluster, but develop their algorithm using MPI. In terms of querying, Husain et al. [4] perform standard queries on the Lehigh University Benchmark (LUBM) [3] and SP2Bench [5] on up to 1.1 billion triples with a cluster of 10 nodes.

A common thread to all these results is the use of commodity hardware and distributed memory architectures. In this paper we utilize more specialized hardware, specifically the Cray XMT, and present algorithms for the machine that provide close to an order of magnitude better performance for three fundamental tasks:

– Dictionary Encoding - This is the process of translating Semantic Web data from a verbose string representation to a more concise integer format. We show speedups ranging from 2.4 to 3.3.
– RDFS Closure - This step takes a set of triples and an associated ontology and materializes all inferred triples. We show speedups of around 6-9.
– Query - We examine standard queries from LUBM, and for the more complicated queries, we find between 2.12-28.0 times speedup.

All of these steps can be done almost entirely in memory. Once the raw data is loaded in as the first step of the dictionary encoding, we no longer need to touch disk until a user requests the results of a query to be sent to permanent storage. In this paper we show processing of nearly 20 billion triples, completely in memory.

The rest of the paper is organized as follows. Section 2 describes the Cray XMT and the programming environment. Section 3 describes our approach to dictionary encoding followed by Section 4 that relates our results on RDFS closure. We take a moment in Section 5 to describe our data model. Section 6 presents our results on querying. We then conclude in Section 7.

2 Cray XMT

The Cray XMT is a unique shared-memory machine with multithreaded processors especially designed to support fine-grained parallelism and perform well despite memory and network latency. Each of the custom-designed compute processors (called *Threadstorm* processors) comes equipped with 128 hardware threads, called *streams* in XMT parlance, and the processor instead of the operating system has responsibility for scheduling the streams. To allow for single-cycle context switching, each stream has a program counter, a status word, eight target registers, and thirty-two general purpose registers. At each instruction cycle, an instruction issued by one stream is moved into the execution pipeline. The large number of streams allows each processor to avoid stalls due to memory requests to a much larger extent than commodity microprocessors. For example, after a processor has processed an instruction for one stream, it can cycle through the other streams before returning to the original one, by which time some requests to memory may have completed. Each Threadstorm can currently support 8 GB of memory, all of which is globally accessible. One system we use in this study has 512 processors and 4 TB of shared memory.

Programming on the XMT consists of writing C/C++ code augmented with non-standard language features including generics, intrinsics, futures, and performance-tuning compiler directives such as pragmas. Generics are a set of functions the Cray XMT compiler supports that operate atomically on scalar values, performing either `read`, `write`, `purge`, `touch`, and `int_fetch_add` operations. Each 8-byte word of memory is associated with a full-empty bit and the read and write operations interact with these bits to provide light-weight synchronization between threads. Here are some examples of the generics provided:

– $readxx$: Returns the value of a variable without checking the full-empty bit.
– $readfe$: Returns the value of a variable when the variable is in a full state, and simultaneously sets the bit to be empty.
– $writeef$: Writes a value to a variable if the variable is in the empty state, and simultaneously sets the bit to be full.
– int_fetch_add: Atomically adds an integer value to a variable.

Parallelism is achieved explicitly through the use of futures, or implicitly when the compiler attempts to automatically parallelize `for` loops. Futures allow programmers to explicitly launch threads to perform some function. Besides explicit parallelism through futures, the compiler attempts to automatically parallelize `for` loops, enabling implicit parallelism. The programmer can also provide pragmas that give hints to the compiler on how to schedule iterations of the `for` loop on to various threads, which can be by blocks, interleaved, or dynamic. In addition, the programmer can supply hints on how many streams to use per processor, etc. We extensively use the `#pragma mta for all streams i of n` construct that allows programmers to be cognizant of the total number of streams that the runtime has assigned to the loop, as well as providing an iteration index that can be treated as the id of the stream assigned to each iteration.

2.1 Code Libraries for the XMT

Much of the results outlined below utilize the code from two open source libraries that specifically target the Cray XMT: the MultiThreaded Graph Library (MTGL)[1] and the Semantic Processing Executed Efficiently and Dynamically

[1] https://software.sandia.gov/trac/mtgl

(SPEED-MT)[2] library. The first is a set of algorithms and data structures designed to run scalably on shared-memory platforms such as the XMT. The second is a novel scalable Semantic Web processing capability being developed for the XMT.

3 Dictionary Encoding

The first aspect of semantic databases we examine is that of translating semantic data from a string representation to an integer format. To simplify the discussion, we consider only semantic web data represented in N-Triples. In this format, semantic data is presented as a sequence of lines, each line containing three elements, a subject, a predicate, and an object. An element can either be a URI, a blank node (an anonymous resource), or a literal value (a string value surrounded by quotes with optional language and datatype modifiers). In all cases, an element is a string of arbitrary length. To speed up later processing of the data and to also reduce the size of the semantic graph, a common tactic is to create a dictionary encoding – a mapping from string to integers and vice versa. On the data sets we explore in this paper, we were able to compress the raw data by a factor of between 3.2 and 4.4.

The dictionary encoding algorithm, outlined in Figure 1, is described in more detail below. The dictionary is encapsulated within a class, `RDF_Dictionary`, that has three important members: `fmap`, `rmap`, and `carray`. The `fmap`, or forward map, is an instance of a hash table class that stores the mapping from strings to integer ids. Similarly, `rmap`, or reverse map, stores the opposite mapping, from integers to strings. We use unsigned 64-bit integers in order to support data sets with more than 4 billion unique strings. The hash table implementation is similar to the linear probing method described in Goodman et al. [1]. However, we made some modifications that significantly reduces the memory footprint that will be described in the next section.

Both of `fmap` and `rmap` reference `carray`, which contains a single instance of each string, separated by null terminators. Having a single character array store the unique instances of each string reduces the memory footprint and allows for easy reading and writing of the dictionary to and from disk; however, it does add some complexity to the algorithm, as seen in Figure 1. Also, we support iteratively adding to the dictionary, which introduces further complications.

The dictionary encoding algorithm is invoked with a call to `parse_file`. The variable `ntriple_file` contains the location on disk of the file to be encoded. As of now, we only support processing files in N-Triples or N-Quads[3] format. After reading in the raw data, the algorithm tokenizes the array into individual elements (i.e. subjects, predicates, and objects) in lines 6-10. It does this by inserting a null terminator at the conclusion of each element, and storing the beginning of each element in the `words` array.

We allow for updates to an existing dictionary, so the next `for` loop on lines 11-15 extracts the subset of elements that are new this iteration. Line 11 checks

[2] https://software.sandia.gov/trac/MapReduceXMT

[3] http://sw.deri.org/2008/07/n-quads/

```
Procedure: RDF_Dictionary.parse_file(char* ntriple_file)

Relevant class member variables:
hash_table<char*, int>* fmap      ▷ Mapping from strings to ints
hash_table<int, char*>* rmap      ▷ Mapping from ints to strings
char* carray                      ▷ Contains single instance of each string

 1: char* data ← read(ntriple_file)

    Initialize:
 2: char** words
 3: char** keys
 4: hash_table<char*, int>* tmap
 5: unsigned long* output

 6: for $i$ ← 0...len(data) - 1 do
 7:     if data[$i$] == '\n' then
 8:         process_line(&data[$i+1$])
 9:     end if
10: end for

11: for all $w$ in words do
12:     if fmap->member($w$) then
13:         tmap->insert($w$,1)
14:     end if
15: end for

16: start ← get_max_value(fmap) + 1
17: assign_contiguous_ids(tmap, start)
18: num_new, keys ← get_keys(tmap)
19: plen ← consolidate(num_new, keys)
20: num_keys ← num_new + fmap->size()

21: if $\frac{num\_keys}{max\_load}$ > fmap->capacity() then
22:     exp ← $\lfloor log_2(num\_keys/max\_load) \rfloor$
23:     newsize ← $2^{exp+1}$
24:     fmap->resize(newsize)
25:     rmap->resize(newsize)
26: end if

27: for $i$ ← plen...len(carray) - 1 do
28:     if carray[$i$] == '\0' then
29:         id ← tmap->lookup(&carray [$i+1$])
30:         fmap->insert(&carray [$i+1$], id)
31:         rmap->insert(id, &carray [$i+1$])
32:     end if
33: end for

34: for $i$ ← 0...len(words) - 1 do
35:     output[$i$] ← fmap->lookup(words[$i$])
36: end for
```

Fig. 1. Overview of Dictionary Encoding Algorithm on the XMT

to see if the string is already stored in the fmap and inserts them into a function-scoped instance of the map class, tmap. Notice that for each new word we insert the value one. The actual ids that will be added to the dictionary are assigned in the next block of code. Doing so allows us to avoid memory contention on a counter variable and use efficient range iterators that come with the hash table class.

The block of lines from 16 through 20 assigns ids to the new set of elements, and then appends the new elements to the end of carray. Line 16 determines the largest id contained within the dictionary and increments that value by one, thus specifying the starting id for the new batch of strings. If the dictionary is empty, the starting id is one, reserving zero as a special value required by the hash table implementation. Line 17 calls the function assign_contiguous_ids which iterates through the keys of the hash table and assigns them values $v \in [start, start + num\_new]$, thus ensuring that regardless of how many times parse_file is called, the ids are in the range $[1, num\_keys]$, where num_keys is the total number of keys. Line 18 gathers the new elements into a contiguous array, keys. Line 19 takes keys and copies the data to the end of carray, placing null terminators between each element. The function consolidate returns the previous size of carray and assigns that value to plen. Line 20 updates the total number of unique elements.

Once we've updated the number of keys, we can then test if the forward and reverse maps need to be resized. On line 21, if the total number of keys divided by the maximum load factor exceeds the current capacity of the table (the total number of slots in the table, claimed or unclaimed), then we resize both maps. The new size is set to be the smallest power of two such that $num\_keys/capacity < max\_load$.

After the forward and reverse maps have been resized if necessary, they are then updated with the new elements and new ids in lines 27 through 33. Since the new elements have been added to the end of `carray`, we iterate through that portion of the array. Each time we find a null terminator at position i, we know that an element to be added starts at $i + 1$. We find the corresponding id from `tmap`, and then add the pair to each map. With the forward and reverse maps updated, we are finally ready to translate the elements listed in the `words` array into integers and store the result in the output buffer in lines 34 through 36.

After the data has been encoded as integers, we are then ready to move on to the next step, that of performing inferencing. An optional step is to write out the translated data and the mapping between strings and integers to disk. This is done by means of three files:

– *<dataset>.translated*: A binary file of 64-bit unsigned integers that contain the triples encoded as integer values.
– *<dataset>.chararr*: This contains the contents of `carray`.
– *<dataset>.intarr*: Another binary file of 64-bit unsigned integers. The sequence of integers corresponds to the same sequence of words found in *<dataset>.chararr*, thus preserving the mapping defined between strings and integers.

3.1 Results

We examined four data sets: *Uniprot*[4], *DBPedia*[5], *Billion Triple Challenge 2009*[6] (*BTC2009*), and the *Lehigh University Benchmark* (*LUBM(8000)*). We also ran the dictionary encoding on a LUBM data set consisting of 16.5 billion triples. This is roughly equivalent to *LUBM(120000)*, though we generated it using several different concurrent runs of the generator using different random seeds and different offsets. These sets represent a wide variety, ranging from the well-behaved, synthetic triple set of LUBM, to real-world but curated sets such as *DBPedia* and *Uniprot*, to the completely wild sources like *BTC2009*, which was formed by crawling the web.

We evaluated the dictionary encoding code using two different-sized systems, a 512-processor XMT and a 128-processor system. Each XMT comes equipped with a service partition. On the service nodes a Linux process called a file service worker (*fsworker*) coordinates the movement of data from disk to the compute nodes. Multiple fsworkers can run on multiple service nodes, providing greater

[4] http://www.uniprot.org
[5] http://wiki.dbpedia.org
[6] http://challenge.semanticweb.org

Table 1. The data sets and the compression achieved

Data set	Size(GB)	Compression Ratio	Size Dictionary On Disk (GB)	Size Dictionary In-memory (GB)
BTC2009	247	4.34	31.1	44.8
DBPedia	36.5	3.2	5.65	9.15
LUBM	185	4.37	17.7	31.7
Uniprot	250	3.94	19.6	33.2

Table 2. Comparison to Urbani et al [8]

Data set	MapReduce rate (MB/s)	XMT rate (MB/s)	Improvement
DBPedia	36.4	120	3.29
LUBM	67.1	162	2.41
Uniprot	48.8	161	3.30

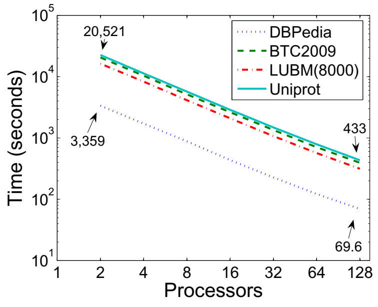
aggregate bandwidth. The 512 system has 16 service nodes and can run 16 fsworkers. However, our 128 system is limited to 2 service nodes and 2 fsworkers. Since this is an artificial constraint of our smaller installation, we decided to estimate the rate that would have been achieved had 16 fsworkers been available. Since throughput degrades linearly as the number of fsworkers is reduced, we compute I/O performance by multiplying the measured rate by a factor of eight to characterize the behavior of a machine configuration more amenable to I/O.

Table 1 shows the raw sizes of the original data sets and the compression ratio achieved. The compression ratio is calculated with

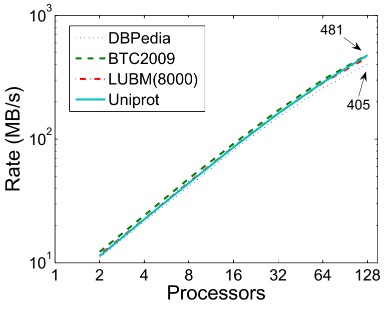
$$\frac{s_o}{s_i + s_c + s_t}$$

where s_o is the size of the original data set, s_i is the size of the dictionary integer array, s_c is the size of the dictionary character array, and s_t is the size of the encoded triples. The size of the dictionary on disk is $s_i + s_c$ while the size of the dictionary in memory is the total memory footprint of the dictionary. Going from disk to memory increases the size of the dictionary by about a factor between 1.5 and 2. This is due to the hash table implementation which requires load factors lower than 0.7 to work efficiently.

Table 2 gives a comparison to a MapReduce dictionary encoding algorithm presented by Urbani, et al. [8]. We compare rates achieved using 32 Threadstorm processors versus a 32 quad-core cluster. We range from a 2.4 to a 3.3 times improvement. Rate is calculated by dividing the size of the original data set by the total time to read the data from disk to memory, perform the encoding algorithm, and write the encoding and dictionary to disk. It should be noted that the datasets are of similar variety, but of different sizes. *DBPedia* and *Uniprot* have grown since the time when the Urbani paper was published to when we examined them. Also, we used a larger LUBM dataset. Figure 2(a) displays the



(a) Times for Dictionary Encoding



(b) Dictionary Encoding Rates

Fig. 2. (a) shows the compute times for the data sets and varying number of processors. (b) displays the encoding rates achieved. The rate is defined as the original file size divided by the time it takes to read the file into memory, perform the calculation, and write the translated triples and dictionary to disk. The file I/O times were estimated to what would be achieved using 16 fsworkers.

times obtained for the compute portion (i.e. excluding file I/O) of the dictionary encoding process. Regardless of the nature of the data, we see nearly linear speedup of 47-48x. Figure 2(b) presents the encoding rates. This includes an estimated I/O time that would have been obtained with 16 fsworkers. The rates fall within a relatively tight band except *DBPedia*, which is about 15% slower. We are unsure if this is due to the nature of the data within *DBPedia*, or due to the fact that file is significantly smaller than the other datasets.

We ran the 512 system on *LUBM(120000)*. We ran once using all 512 processors, iteratively processing a third of the data at a time. The times for each chunk were 1412, 2011, and 1694 seconds. The times of the latter files are longer than the first due to the need to check against the existing table, and also second file required a resize of the forward and reverse hash tables. Overall the rate achieved was 561 MB/s. Extrapolating from our *LUBM(8000)* 2-processor run, ideally we would have achieved 2860 MB/s, representing an efficiency of about .20. If we had run had concatenated all the data together, the rate of the 512 run would have been significantly better.

4 RDFS Closure

We presented an algorithm for RDFS closure in previous work [2]. In general the process we described is to keep a large hash table, ht, in memory and also smaller hash tables as queues for the RDFS rules, q_i. We first iterate through all the triples, adding the original set to ht, and any triples that match a given rule is added to the appropriate q_i. Then, upon invocation of a rule, we iterate through its cue instead of the entire data set. The algorithm assumes the ontology does not operate on RDFS properties. As such, a single pass through the RDFS rule set is sufficient.

The algorithm we employed in this paper is largely the same. We did make some modifications that resulted in a 50% decrease in the memory footprint, namely with

– removal of the occupied array in the hash table and hash set implementations, and
– removal of the rule queues.

In our previous work on hashing for the Cray XMT [1], we outlined an open addressing scheme with linear probing, the key contribution being a mechanism for avoiding locking except for when a slot in the hash table is declared occupied for a given key. The open addressing scheme makes use of two arrays, a `key` array and an `occupied` array. The `key` array stores the keys assigned to various slots in the hash table, while the `occupied` array handles hash collisions and thread synchronization. The `occupied` array acts as a boolean, a 1 indicating that the slot is taken and a 0 otherwise (this assumes we don't care about deleting and reclaiming values, else we need another bit). Despite the `occupied` array being a boolean, each position in the array is a 64-bit integer. Threads need to be able to interact with the full-empty bit for synchronization, and full-empty bits are only associated with each 8-byte word. However, an important observation is that the `occupied` array is only necessary for a general implementation that is agnostic to the key distribution. In situations where there is a guarantee that a particular key k will never occur, we can use the `key` array itself for thread synchronization and use k as the value indicating a slot is empty. When we initialize the `key` array, we set all the values to k. Since we control what values are assigned during the dictionary encoding, we reserve $k = 0$ as the special value indicating a slot is open.

The second change we employed is the removal of queues. In our previous implementation, we made use of queues for RDFS rules. As we processed existing triples or added new triples through inference, we would check to see if the triple under consideration matches a rule. If so, we would add it to the appropriate queue. Then, when the rule was actually evaluated, we iterated over the queue instead of the entire dataset, thus saving computation. To save on space, we removed the queues. This change did result in a small increase in computation time. We examined *LUBM(8000)* and found about a 33% increase in computation time for small processor counts, but for 128 the increase in time was only 11%.

4.1 Results

We examined performing closure on *LUBM(8000)* and *BTC2009*. For *BTC2009*, we used the higher-level ontology described by Williams et al. [10]. *BTC2009* is a collection of data crawled from the web. As such, it is questionable whether the ontological information procured from sundry sources should be applied to the entire data set. For instance, some ontological triples specified superproperties for `rdf:type`. While expansion of `rdf` and `rdfs` namespaces may be appropriate for some portion of *BTC2009*, namely the source from which the ontological information is taken, it doesn't make sense for the rest of the data. Also, this type of expansion violates the single-pass nature of our algorithm, and would require multiple passes. As such, we removed all ontological triples (i.e. any triple with `rdfs` or `owl` in the namespace of the predicate) from *BTC2009* and added the higher level ontology.

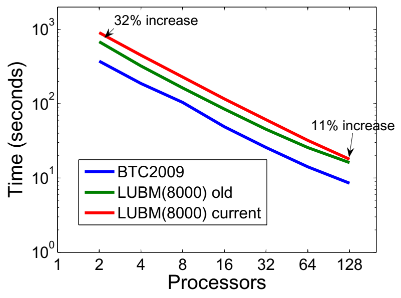


Fig. 3. This figure shows the times obtained by running RDFS closure on *LUBM(8000)* and *BTC2009*

Table 3. This table shows the speedup our RDFS closure algorithm achieved against other approaches on LUBM data sets

Query	With I/O	Without I/O
MPI	6.0	6.8
WebPIE	9.0	10.6

Figure 3 displays the results of running our RDFS closure algorithm on the two different data sets. For comparison, we also include the times using the previous approach on *LUBM(8000)*. Table 3 provides comparison with other approaches. We refer to the work of Weaver and Hendler [9] as MPI as they use an MPI-based approach. *WebPIE* refers to the work of Urbani et al. [7]. We extract the WebPIE rate for RDFS out of a larger OWL computation. In both cases we compare equal number of Threadstorm processors with quad-core nodes (32 for MPI and 64 for WebPIE). We present the comparison with and without I/O. As this part of our pipeline does not require I/O, it seems fair to consider the comparison between our non I/O numbers with the previous approaches, whose processing relies upon access to disk. Though to aid in an apples-to-apples comparison, we include estimated rates that would be garnered with I/O using 16 fsworkers.

We also ran RDFS closure on *LUBM(120000)* with 512 processors. The final triple total came in at 20.1 billion unique triples. We achieved an inference rate of 13.7 million inferences/second when we include I/O, and 21.7 million inferences/second without I/O. Again using the 2 processor run on *LUBM(8000)* as a baseline, ideally we would want to see 77.2 million inferences/second when ignoring I/O. This gives an estimate on efficiency of 0.28.

5 Data Model: A Graph

Once we have the data encoded as integers, and all RDFS inferences have been materialized, we are now ready to store the data within a data model. Previous to this step, the triples had been stored in one large array. Instead of trying to fit standard relational DBMS-style models to sets of triples, we opt to model each triple as a directed edge in a graph. The subject of a triple is a vertex on the graph, the predicate is a typed edge, with the head being the subject and the tail being the object, another vertex.

We present some basic notation to facilitate discussions of the graph data model. A graph is defined in terms of vertices, V, and edges E, i.e. $G = (V, E)$. The graphs we consider are directed, meaning that the edges point from a head vertex to a tail vertex. We use $E(v)$ to denote the edges incident on vertex v, while $E^-(v)$ denotes only the incoming edges and $E^+(v)$ signifies the outgoing edges. Similarly we define degree, the number of edges incident to vertex v as $deg(v)$, $deg^-(v)$, and $deg^+(v)$. We use $source(e)$ and $dest(e)$ to denote the head and tail vertices for an edge e. Also, we enumerate the edges, and refer to the i^{th} edge incident with v using the notations $E(v)[i]$, $E(v)^-[i]$, and $E(v)^+[i]$.

6 Querying

Once we have the data in graph form, we can now utilize that information to perform efficient SPARQL queries. LUBM [3] provides several standard queries. For the purposes of discussion we list query 1:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
    {?X rdf:type ub:GraduateStudent .
     ?X ub:takesCourse
        http://www.Department0.University0.edu/GraduateCourse0}
```

The `WHERE` clause contains a set of what are called basic graph patterns (BGPs). They are triple patterns that are applied to the data set, and those elements that fit the described constraints are returned as the result. The above SPARQL query describes formally a request to retrieve all graduate students that take a particular course. It is important to note that there are basically only two possibilities for the number of variables within a BGP, one and two. The other two cases are degenerate: a BGP with no variables has no effect on the result set, and a BGP with all variables simply matches everything.

Here we present an algorithm we call *Sprinkle SPARQL*. The algorithm begins by creating an array of size $|V|$ for each variable specified in the query (see line 1 of Figure 4). Each array is called a_w for every variable w. We then evaluate each BGP by incrementing a counter in the array for a given variable each time a node in the graph matches the BGP. For example, say we have a BGP, b, similar to the two BGPs of LUBM query 1, where the subject $b.s$ is a variable but $b.p$ and $b.o$ are fixed terms (line 2 of Figure 5). In that case, we use the graph data model (line 3 of Figure 5) and start at the object and iterate through all the $deg^{-}(b.o)$ edges in $E(b.o)$. If an edge matches $b.p$, we then increment the counter for the subject at the source of the edge $(source(E^{-}(b.o)[i])$ in the temporary array t. We use a temporary array to prevent the possibility of the counter for a node being incremented more than once during the application of a single BGP. Once we have iterated through all the edges associated with $b.o$ and found matching subjects, we then increment positions within $a_{b.s}$ that have a non-zero corresponding position in t. In the interest of space, we omit the a description of the other cases. Note that the algorithm as currently defined excludes the possibility of having a variable in the predicate position; we will leave that as future work. It is this process of iterating through the BGPs and incrementing counters that we liken to sprinkling the information from SPARQL BGPs across the array data structures.

```
Algorithm: Querying via Sprinkle SPARQL

Let $B$ be a set of Basic Graph Patterns and $W$ be the set of variables
contained in $B$. For $w \in W$, let $|w|$ denote the number of times the
variable appears in $B$. Also, for $b \in B$, let $|b|$ denote the number of
variables in $b$. To query, perform the following:
 1: $\forall w \in W$, create a set of arrays $A$ such that $\forall a_w \in A : |a_w| = |V| \wedge \forall i \in [0, |V|-1] : a_w[i] = 0$
 2: $\forall b \in B, Sprinkle(b, A)$
 3: Select $w_{min} \leftarrow \min_{w_i \in W} \sum_{j=1}^{|V|} a_{w_i}[j] = |w_i|$
 4: Create result set $R$, initially populated with all $v : a_{w_{min}}[v] = |w_{min}|$
 5: Let $B(2) = \{b | b \in B \wedge |b| = 2\}$
 6: while $B(2) \neq \emptyset$ do
 7:     Let $B_{match} = \{b | b \in B(2) \wedge \exists w \in b : w \in R\}$
 8:     Select $b_{min} \leftarrow \min_{b \in B_{match}} |GraphJoin(R, b)|$
 9:     $R \leftarrow GraphJoin(R, b_{min})$
10:     $B(2) \leftarrow B(2) - b_{min}$
11: end while
```

Fig. 4. This figure gives an overview of the *Sprinkle SPARQL* algorithm

Once we have applied each of the BGPs $b \in B$ to A, if the counter associated with node i in a_w matches the number of times that w appears in B, then that node is a candidate for inclusion. In essence, we have reduced the set of possibilities for the result set.

The next step is to iterate through all BGPs that have 2 variables, applying those constraints to the set of possible matches defined by Line 2 of Figure 4. Line 3 of Figure 4 selects the variable with the smallest number of nodes that match, beginning a greedy approach for dynamically selecting the order of execution. We populate the initial result set with these matching nodes. At this

```
**Procedure**: $Sprinkle(b, A)$
Let $B$ and $W$ be the same as above. Let $F$ be the set of fixed terms (not variables) in $B$
 1: Create a temporary array $t$ of size $|V|$ where $\forall i \in [0, |V|-1] : t[i] = 0$
 2: if $b.s \in W \wedge b.p \in F \wedge b.o \in F$ then
 3:     for $i \leftarrow 0...deg^{-}(b.o) - 1$ do
 4:         if $E^{-}(b.o)[i] = b.p$ then
 5:             $s \leftarrow source(E^{-}(b.o)[i])$
 6:             $t[s]$++
 7:         end if
 8:     end for
 9:     for $i \leftarrow 0...|V| - 1$ do
10:         if $t[i] > 0$ then
11:             $a_{b.s}[i]$++
12:         end if
13:     end for
14: else if $b.s \in F \wedge b.p \in F \wedge b.o \in W$ then
    ...
15: else if $b.s \in W \wedge b.p \in F \wedge b.o \in W$ then
    ...
16: end if
```

Fig. 5. This figure outlines the *Sprinkle* process

point, One can think of the result set as a relational table with one attribute. We then iterate through all the 2-variable BGPs in lines 6 through 11, where for each iteration we select the BGP that creates in the smallest result set. For lack of a better term, we use the term *join* to denote the combination of the result set R with a BGP b, and we use $GraphJoin(R, b)$ to represent the operation. Consider that $GraphJoin(R, b)$ has two cases:

– A variable in R matches one variable in b, and the other variable in b is unmatched.
– Two variables in R match both variables in b.

In the former case, the join adds in an additional attribute to R. In the latter case, the join further constrains the existing set of results. Our current implementation calculates the exact size of each join. An obvious improvement is to select a random sample to estimate the size of the join.

6.1 Results

Here we present the results of running *Sprinkle SPARQL* on LUBM queries 1-5 and 9. Of the queries we tested, 4, 5, and 9 require inferencing, with 9 needing *owl:intersectionOf* to infer that all graduate students are also students. Since we do not yet support OWL, we added this information as a post-processing step after caclucating RDFS closure. Figure 6 shows the times we obtained with the method. We report the time to perform the calculation together with the time to

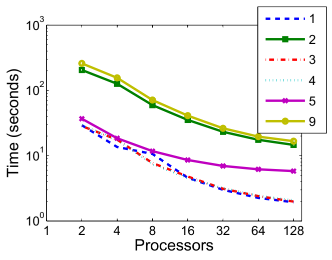


Fig. 6. This figure shows the times of running *Sprinkle SPARQL* on LUBM queries 1-5 and 9

Table 4. This table shows the speedup *Sprinkle SPARQL* achieved against other approaches for queries 2 and 9

Query	MapReduce	BigOWLIM
2	13.6	2.12
9	28.0	2.82

either print the results to the console or to store the results on disks, whichever is faster. For smaller queries, it makes sense to report the time to print to screen as a human operator can easily digest small result sets. For the larger result sets, more analysis is likely necessary, so we report the time to store the query on disk.

Queries 2 and 9 are the most complicated and they are where we see the most improvement in comparison to other approaches. We compare against a MapReduce approach by Husain et al. [4] and against the timings reported for BigOWLIM on *LUBM(8000)*[7]. This comparison is found in Table 4. For the MapReduce work, we compare 10 Threadstorm processors to an equal number of quad-core processors. For the comparison against BigOWLIM, we compare against the quad-core system, *ontosol*, dividing the times of our two-processor count runs by two to give a fair comparison. Ideally, we would like to compare against larger processor counts, but we could find nothing in the literature.

Queries 1, 3, and 4 have similar performance curves. The majority of the time is consumed during the *Sprinkle* phase, which down-selects so much that later computation (if there is any) is inconsequential. For comparison we ran a simple algorithm on query 1 that skips the *Sprinkle* phase, but instead executes each BGP in a greedy selection process, picking the BGPs based upon how many triples match the pattern. For query 1, this process chooses the second BGP, which has 4 matches, followed by the first BGP, which evaluated by itself has over 20 million matches. For this simple approach, we arrive at a time of 0.33 seconds for 2 processors as opposed to 29.28 with *Sprinkle SPARQL*, indicating that *Sprinkle SPARQL* may be overkill for simple queries. Query 5 has similar computational runtime to 1, 3, and 4, but because of a larger result set (719 versus 4, 6, and 34), takes longer to print to screen. For these simple queries, *Sprinkle SPARQL* performs admirably in comparison to the MapReduce work, ranging between 40 - 225 times faster, but comparing to the BigOWLIM results, we don't match their times of between 25 and 52 msec. As future work, we plan to investigate how we can combine the strategies of *Sprinkle SPARQL* and a simpler approach without *Sprinkle* (and perhaps other approaches) to achieve good results on both simple and complex queries.

[7] http://www.ontotext.com/owlim/benchmarking/lubm.html

7 Conclusions

In this paper we presented a unique supercomputer with architecturally-advantageous features for housing a semantic database. We showed dramatic improvement for three fundamental tasks: dictionary encoding, rdfs closure, and querying. We have shown the performance value of holding large triple stores in shared memory. We have also demonstrated scaling up to 512 processors.

Acknowledgments. This work was funded under the Center for Adaptive Supercomputing Software - Multithreaded Architectures (CASS-MT) at the Dept. of Energy's Pacific Northwest National Laboratory. Pacific Northwest National Laboratory is operated by Battelle Memorial Institute under Contract DE-ACO6-76RL01830.

References

1. Goodman, E.L., Haglin, D.J., Scherrer, C., Chavarría-Miranda, D., Mogill, J., Feo, J.: Hashing Strategies for the Cray XMT. In: Proceedings of the IEEE Workshop on Multi-Threaded Architectures and Applications, Atlanta, GA, USA (2010)
2. Goodman, E.L., Mizell, D.: Scalable In-memory RDFS Closure on Billions of Triples. In: Proceedings of the 4th International Workshop on Scalable Semantic Web Knowledge Base Systems, Shanghai, China (2010)
3. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 158–182 (2005)
4. Husain, M.F., Khan, L., Kantarcioglu, M., Thuraisingham, B.: Data Intensive Query Processing for Large RDF Graphs Using Cloud Computing Tools. In: Proceedings of the 3rd International Conference on Cloud Computing, Maimi, Florida (2010)
5. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: A SPARQL Performance Benchmark. In: Proceedings of the 25th International Conference on Data Engineering, Shanghai, China (2009)
6. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
7. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL reasoning with WebPIE: calculating the closure of 100 billion triples. In: Proceedings of the 7th Extended Semantic Web Conference, Heraklion, Greece (2010)
8. Urbani, J., Maaseen, J., Bal, H.: Massive Semantic Web data compression with MapReduce. In: Proceedings of the MapReduce Workshop at High Performance Distributed Computing Symposium, Chicago, IL, USA (2010)
9. Weaver, J., Hendler, J.A.: Parallel materialization of the finite RDFS closure for hundreds of millions of triples. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer, Heidelberg (2009)
10. Williams, G.T., Weaver, J., Atre, M., Hendler, J.: Scalable Reduction of Large Datasets to Interesting Subsets. In: Billion Triples Challenge, Washington D.C., USA (2009)

An Intermediate Algebra for Optimizing RDF Graph Pattern Matching on MapReduce

Padmashree Ravindra, HyeongSik Kim, and Kemafor Anyanwu

Department of Computer Science, North Carolina State University, Raleigh, NC
{pravind2,hkim22,kogan}@ncsu.edu

Abstract. Existing MapReduce systems support relational style join operators which translate multi-join query plans into several Map-Reduce cycles. This leads to high I/O and communication costs due to the multiple data transfer steps between *map* and *reduce* phases. SPARQL graph pattern matching is dominated by join operations, and is unlikely to be efficiently processed using existing techniques. This cost is prohibitive for RDF graph pattern matching queries which typically involve several join operations. In this paper, we propose an approach for optimizing graph pattern matching by reinterpreting certain join tree structures as grouping operations. This enables a greater degree of parallelism in join processing resulting in more “bushy” like query execution plans with fewer Map-Reduce cycles. This approach requires that the intermediate results are managed as sets of groups of triples or *TripleGroups*. We therefore propose a data model and algebra - *Nested TripleGroup Algebra* for capturing and manipulating TripleGroups. The relationship with the traditional relational style algebra used in Apache Pig is discussed. A comparative performance evaluation of the traditional Pig approach and RAPID+ (Pig extended with NTGA) for graph pattern matching queries on the BSBM benchmark dataset is presented. Results show up to 60% performance improvement of our approach over traditional Pig for some tasks.

Keywords: MapReduce, RDF graph pattern matching, optimization techniques.

1 Introduction

With the recent surge in the amount of RDF data, there is an increasing need for scalable and cost-effective techniques to exploit this data in decision-making tasks. MapReduce-based processing platforms are becoming the *de facto* standard for large scale analytical tasks. MapReduce-based systems have been explored for scalable graph pattern matching [1][2], reasoning [3], and indexing [4] of RDF graphs. In the MapReduce [5] programming model, users encode their tasks as *map* and *reduce* functions, which are executed in parallel on the Mappers and Reducers respectively. This two-phase computational model is associated with an inherent communication and I/O overhead due to the data transfer between the Mappers and the Reducers. Hadoop[1] based systems like Pig [6] and Hive [7] provide high-level query languages that improve usability and support automatic data flow optimization similar to database systems. However,

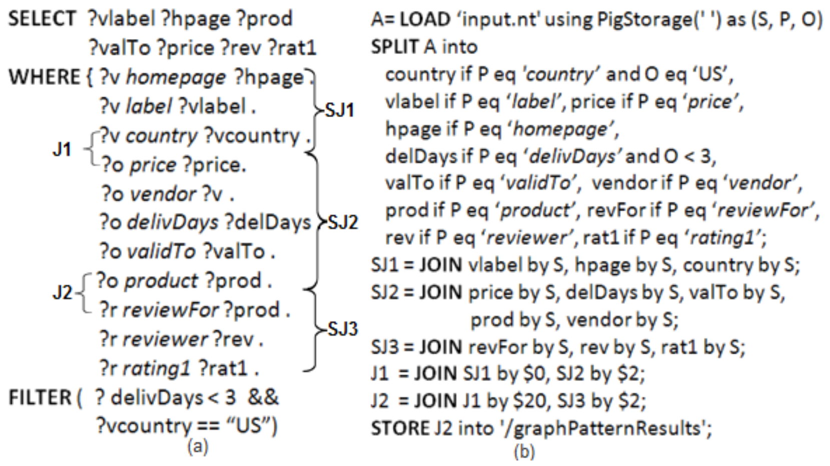
[1] http://hadoop.apache.org/core/

most of these systems are targeted at structured relational data processing workloads that require relatively few numbers of join operations as stated in [6]. On the contrary, processing RDF query patterns typically require several join operations due to the fine-grained nature of RDF data model. Currently, Hadoop supports only *partition parallelism* in which a single operator executes on different partitions of data across the nodes. As a result, the existing Hadoop-based systems with the relational style join operators translate multi-join query plans into a linear execution plan with a sequence of multiple Map-Reduce (MR) cycles. This significantly increases the overall communication and I/O overhead involved in RDF graph processing on MapReduce platforms. Existing work [8][9] directed at uniprocessor architectures exploit the fact that joins presented in RDF graph pattern queries are often organized into star patterns. In this context, they prefer bushy query execution plans over linear ones for query processing. However, supporting bushy query execution plans in Hadoop based systems would require significant modification to the task scheduling infrastructure.

In this paper, we propose an approach for increasing the degree of parallelism by enabling some form of *inter-operator parallelism*. This allows us to "sneak in" bushy like query execution plans into Hadoop by interpreting star-joins as groups of triples or *TripleGroups*. We provide the foundations for supporting TripleGroups as first class citizens. We introduce an intermediate algebra called the *Nested TripleGroup Algebra* (NTGA) that consists of TripleGroup operators as alternatives to relational style operators. We also present a data representation format called the *RDFMap* that allows for a more easy-to-use and concise representation of intermediate query results than the existing format targeted at relational tuples. RDFMap aids in efficient management of schema-data associations, which is important while querying *schema-last* data models like RDF. Specifically, we propose the following:

- A TripleGroup data model and an intermediate algebra called *Nested TripleGroup Algebra* (NTGA), that leads to efficient representation and manipulation of RDF graphs.
- A compact data representation format (*RDFMap*) that supports efficient TripleGroup-based processing.
- An extension to Pig's computational infrastructure to support NTGA operators, and compilation of NTGA logical plans to MapReduce execution plans. Operator implementation strategies are integrated into Pig to minimize costs involved in RDF graph processing.
- A comparative performance evaluation of Pig and RAPID+ (Pig extended with NTGA operators) for graph pattern queries on a benchmark dataset is presented.

This paper is organized as follows: In section 2, we review the basics of RDF graph pattern matching, and the issues involved in processing such pattern queries in systems like Pig. We also summarize the optimization strategies presented in our previous work, which form a base for the algebra proposed in this paper. In section 3.1, we present the TripleGroup data model and the supported operations. In 3.2, we discuss the integration of NTGA operators into Pig. In section 4, we present the evaluation results comparing the performance of RAPID+ with the existing Pig implementation.



```
SELECT ?vlabel ?hpage ?prod
       ?valTo ?price ?rev ?rat1
WHERE { ?v homepage ?hpage .
        ?v label ?vlabel .
        ?v country ?vcountry .
        ?o price ?price.
        ?o vendor ?v .
        ?o delivDays ?delDays
        ?o validTo ?valTo .
        ?o product ?prod .
        ?r reviewFor ?prod .
        ?r reviewer ?rev .
        ?r rating1 ?rat1 .
FILTER ( ? delivDays < 3 &&
        ?vcountry == "US")
```

```
A= LOAD 'input.nt' using PigStorage(' ') as (S, P, O)
SPLIT A into
  country if P eq 'country' and O eq 'US',
  vlabel if P eq 'label', price if P eq 'price',
  hpage if P eq 'homepage',
  delDays if P eq 'delivDays' and O < 3,
  valTo if P eq 'validTo', vendor if P eq 'vendor',
  prod if P eq 'product', revFor if P eq 'reviewFor',
  rev if P eq 'reviewer', rat1 if P eq 'rating1';
SJ1 = JOIN vlabel by S, hpage by S, country by S;
SJ2 = JOIN price by S, delDays by S, valTo by S,
           prod by S, vendor by S;
SJ3 = JOIN revFor by S, rev by S, rat1 by S;
J1  = JOIN SJ1 by $0, SJ2 by $2;
J2  = JOIN J1 by $20, SJ3 by $2;
STORE J2 into '/graphPatternResults';
```

Fig. 1. Example pattern matching query in (a) SPARQL (b) Pig Latin (VP approach)

2 Background and Motivation

2.1 RDF Graph Pattern Matching

The standard query construct for RDF data models is a graph pattern which is equivalent to the select-project-join (SPJ) construct in SQL. A graph pattern is a set of triple patterns which are RDF triples with variables in either of the s, p, or o positions. Consider an example query on the BSBM[2] data graph to retrieve "*the details of US-based vendors who deliver products within three days, along with the review details for these products*". Fig. 1 (a) shows the corresponding SPARQL query, whose graph pattern can be factorized into two main components (i) three star-join structures ($SJ1$, $SJ2$, $SJ3$) describing resources of type Vendor, Offer, and Review respectively, two chain-join patterns ($J1$, $J2$) combining these star patterns and, (ii) the filter processing.

There are two main ways of processing RDF graph patterns depending on the storage model used: (i) triple model, or (ii) vertically partitioned (VP) storage model in which the triple relation is partitioned based on properties. In the former approach, RDF pattern matching queries can be processed as series of relational style self-joins on a large triple relation. Some systems use multi-indexing schemes [8] to counter this bottleneck. The VP approach results in a series of join operations but on smaller property-based relations. Another observation [8] is that graph pattern matching queries on RDF data often consist of multiple star-structured graph sub patterns. For example, 50% of the benchmark queries in BSBM have at least two or more star patterns. Existing work [8][9] optimize pattern matching by exploiting these star-structures to generate bushy query plans.

[2] `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/`

2.2 Graph Pattern Matching in Apache Pig

Graph Pattern Matching in Pig. Map Reduce data processing platforms like Pig focus on ad hoc data processing in the cloud environment where the existence of preprocessed and suitably organized data cannot be presumed. Therefore, in the context of RDF graph pattern processing which is done directly from input documents, the VP approach with smaller relations is more suitable. To capture the VP storage model in Pig, an input triple relation needs to be "split" into property-based partitions using Pig Latin's `SPLIT` command. Then, the star-structured joins are achieved using an m-way `JOIN` operator, and chain joins are executed using th traditional binary `JOIN` operator. Fig. 1(b) shows how the graph pattern query in Fig. 1(a) can be expressed and processed in Pig Latin. Fig. 2 (a) shows the corresponding query plan for the VP approach. We refer to this sort of query plan as Pig's approach in the rest of the paper. Alternative plans may change the order of star-joins based on cost-based optimizations. However, that issue does not affect our discussion because the approaches compared in this paper all benefit similarly from such optimizations. Pig Latin queries are compiled into a sequence of Map-Reduce (MR) jobs that run over Hadoop. The Hadoop scheduling supports *partition parallelism* such that in every stage, one operator is running on different partitions of data at different nodes. This leads to a linear style physical execution plan. The above logical query plan will be compiled into a linear execution plan with a sequence of five MR cycles as shown in Fig. 2 (b). Each join step is executed as a separate MR job. However, Pig optimizes the multi-way join on the same column, and compiles it into a single MR cycle.

Issues. (i) Each MR cycle involves communication and I/O costs due to the data transfer between the Mappers and Reducers. Intermediate results are written to disk by Mappers after the *map* phase, which are read by Reducers and processed in the *reduce* phase after which the results are written to HDFS (Hadoop Distributed File System). These costs are summarized in Fig. 2 (b). Using this style of execution where join operations are executed in different MR cycles, join-intensive tasks like graph pattern matching will

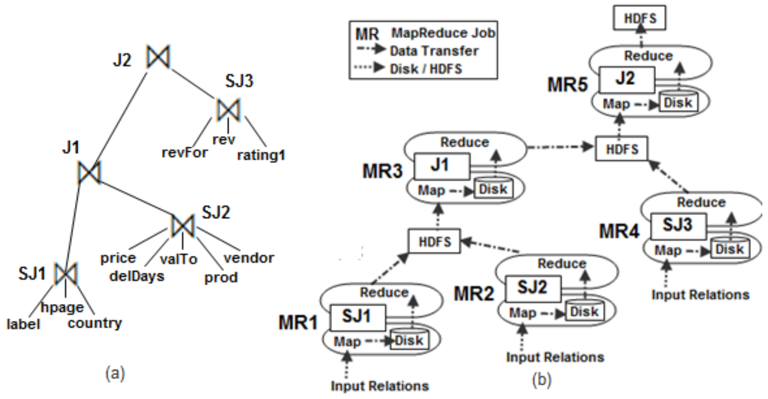


Fig. 2. Pattern Matching using VP approach (a) Query plan (b) Map-Reduce execution flow

result in significant I/O and communication overhead. There are other issues that contribute I/O costs e.g. the SPLIT operator for creating VP relations generates concurrent sub flows which compete for memory resources and is prone to disk spills. (ii) In imperative languages like Pig Latin, users need to explicitly manipulate the intermediate results. In *schema-last* data models like RDF, there is an increased burden due to the fact that users have to keep track of which columns of data are associated with which schema items (*properties*) as well as their corresponding values. For example, for the computation of join $J2$, the user needs to specify the join between intermediate relation $J1$ on the value of property type "product", and relation $SJ3$ on the value of property type "reviewFor". It is not straightforward for the user to determine that the value corresponding to property "product" is in column 20 of relation $J1$. In *schema-first* data models, users simply reference desired columns by attribute names.

TripleGroup-based Pattern Matching. In our previous work [10], we proposed an approach to exploit star sub patterns by re-interpreting star-joins using a *grouping-based join algorithm*. It can be observed that performing a group by Subject yields groups of tuples or *TripleGroups* that represent all the star sub graphs in the database. We can obtain all these star sub graphs using the relational style GROUP BY which executes in a *single* MR cycle, thus minimizing the overall I/O and communication overhead in RDF graph processing. Additionally, repeated data processing costs can be improved by *coalescing* operators in a manner analogous to "pushing select into cartesian product" in relational algebra to produce a more efficient operator. The empirical study in our previous work showed significant savings using this TripleGroup computational style, suggesting that it was worth further consideration.

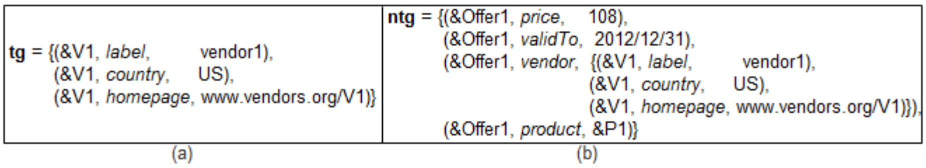
In this paper, we present a generalization of this strategy by proposing an intermediate algebra based on the notion of TripleGroups. This provides a formal foundation to develop first-class operators with more precise semantics, to enable tighter integration into existing systems to support automatic optimization opportunities. Additionally, we propose a more suitable data representation format that aids in efficient and user-friendly management of intermediate results of operators in this algebra. We also show how this representation scheme can be used to implement our proposed operators.

3 Foundations

3.1 Data Model and Algebra

Nested TripleGroup Algebra (NTGA) is based on the notion of the *TripleGroup* data model which is formalized as follows:

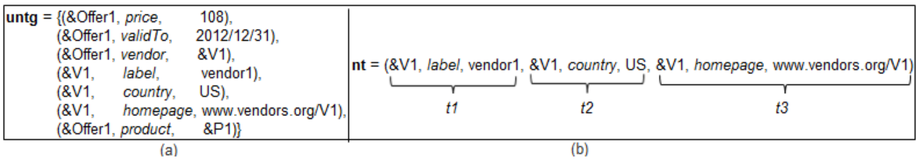
Definition 1. (*TripleGroup*) A TripleGroup tg is a relation of triples t_1,t_2,...t_k, whose schema is defined as (S, P, O). Further, any two triples t_i, $t_j \in tg$ have overlapping components i.e. t_i [col_i] = t_j[col_j] where col_i, col_j refer to subject or object component. When all triples agree on their subject (object) values, we call them $subject$ ($object$) *TripleGroups* respectively. Fig. 3 (a) is an example of a *subject TripleGroup* which corresponds to a star sub graph. Our data model allows TripleGroups to be nested at the object component.



```
tg = {(&V1, label,     vendor1),
      (&V1, country,   US),
      (&V1, homepage,  www.vendors.org/V1)}
(a)

ntg = {(&Offer1, price,   108),
       (&Offer1, validTo, 2012/12/31),
       (&Offer1, vendor,  {(&V1, label,    vendor1),
                           (&V1, country,  US),
                           (&V1, homepage, www.vendors.org/V1)}),
       (&Offer1, product, &P1)}
(b)
```

Fig. 3. (a) Subject TripleGroup tg (b) Nested TripleGroup ntg



```
untg = {(&Offer1, price,    108),
        (&Offer1, validTo,  2012/12/31),
        (&Offer1, vendor,   &V1),
        (&V1,     label,    vendor1),
        (&V1,     country,  US),
        (&V1,     homepage, www.vendors.org/V1),
        (&Offer1, product,  &P1)}
(a)
```

$$nt = (\underbrace{\&V1,\ label,\ vendor1}_{t1},\ \underbrace{\&V1,\ country,\ US}_{t2},\ \underbrace{\&V1,\ homepage,\ www.vendors.org/V1}_{t3})$$

(b)

Fig. 4. (a) ntg.unnest() (b) n-tuple after tg.flatten()

Definition 2. (*Nested TripleGroup*) A nested TripleGroup ntg consists of a root TripleGroup ntg.root and one or more child TripleGroups returned by the function ntg. child() such that:
For each child TripleGroup $ctg \in ntg$. child(),

– $\exists\, t_1 \in ntg$.root, $t_2 \in ctg$ such that t_1.Object = t_2.

Nested TripleGroups capture the graph structure in an RDF model in a more natural manner. An example of a nested TripleGroup is shown in Fig. 3 (b). A nested TripleGroup can be "unnested" into a flat TripleGroup using the unnest operator. The definition is shown in Fig. 5. Fig. 4 (a) shows the TripleGroup resulting from the unnest operation on the nested TripleGroup in Fig. 3 (b). In addition, we define the flatten operation to generate an "equivalent" n-tuple for a given TripleGroup. For example, if tg = t_1, t_2,..., then the n-tuple tu has triple t_1= ($s1$, $p1$, $o1$) stored in the first three columns of tu, triple t_2 = ($s2$, $p2$, $o2$) is stored in the fourth through sixth column, and so on. For convenience, we define the function triples() to extract the triples in a TripleGroup. For the TripleGroup in Fig. 3 (a), the flatten is computed as tg.triples($label$) $\bowtie$ tg.triples($country$) $\bowtie$ tg.triples($homepage$), resulting in an n-tuple as shown in Fig. 4 (b). It is easy to observe that the information content in both formats is equivalent. We refer to this kind of equivalence as *content equivalence* which we will denote as $\cong$. Consequently, computing query results in terms of TripleGroups is lossless in terms of information. This is specifically important in scenarios where TripleGroup-based processing is more efficient.

We define other TripleGroup functions as shown in Fig.5 (b). The *structure-labeling function* λ assigns each TripleGroup tg, with a label that is constructed as some function of tg.props(). Further, for two TripleGroups tg_1, tg_2 such that tg_1.props() $\subseteq$ tg_2.props(), λ assigns labels such that $tg_1.\lambda() \subseteq tg_2.\lambda()$. The labeling function λ induces a partition on a set of TripleGroups based on the structure represented by the property types present in that TripleGroup. Each equivalence class in the partition consists of TripleGroups that have the exact same set of property types.

Next, we discuss some of the TripleGroup operators proj, filter, groupfilter, and join, which are formally defined in Fig.5 (c).

Symbol	Description
tg	TripleGroup
TG	Set of TripleGroups
tp	Triple pattern
ntg.root	Root of the nested TripleGroup
ntg.child()	Children of the nested TripleGroup
$?v_{tp}$	A variable in the triple pattern tp

(a)

Function	Returns
tg.props()	Union of all property types in tg
tg.triples()	Union of all triples in tg
tg.triples(p_i)	Triples in tg with property type p_i
$tg.\lambda()$	Structure label for tg based on tg.props()
$\delta(tp)$	A triple matching the triple pattern tp
$\delta(?v_{tp})$	A variable substituion in the triple matching tp

(b)

Operator	Definition
`load`($\{t_i\}$)	$\{\ tg_i \mid tg_i = t_i$, and t_i is an input triple$\}$
`proj`$_{?v_{tp}}(TG)$	$\{\delta_i(?v_{tp}) \mid \delta_i(tp) \in tg_i, tg_i \in TG$ and $tp.\lambda() \subseteq tg_i.\lambda()\}$
`filter`$_{\Theta(?v_{tp})}(TG)$	$\{\ tg_i \mid tg_i \in TG$ and $\exists\ \delta_i(tp) \in tg_i$ such that $\delta_i(?v_{tp})$ satisfies the filter condition $\Theta(?v_{tp})\}$
`groupfilter`(TG, P)	$\{\ tg_i \mid tg_i \in TG$ and tg_i.`props`() = P $\}$
`join`($?v_{tp_x}$:TG_x, $?v_{tp_y}$:TG_y)	Assume $tg_x \in TG_x$, $tg_y \in TG_y$, $\exists\ \delta_1(tp_x) \in tg_x$, $\delta_2(tp_y) \in tg_y$, and $\delta_1(?v_{tp_x}) = \delta_2(?v_{tp_y})$ if O-S join, then $\{ntg_i \mid ntg_i$.`root` $= tg_x$, $\delta_1(tp_x)$.`Object` $= tg_y\}$ else $\{\ tg_x \cup tg_y\}$
tg.`flatten`()	$\{tg$.`triples`$(p_1) \bowtie tg$.`triples`$(p_2)...\bowtie tg$.`triples`(p_n) where $p_i \in tg$.`props`()$\}$
ntg.`unnest`()	$\{\ t_i \mid t_i$ is a non-nested triple in tg.`root` $\}$ $\cup \{\ (s, p, s') \mid t' = (s, p, (s', p', o'))$ is a nested triple in tg.`root`$\}$ $\cup \{\ ctg_i$.`unnest`() $\mid ctg_i \in tg$.`child()` $\}$

(c)

Fig. 5. NTGA Quick Reference (a) Symbols (b) Functions (c) Operators

(**`proj`**) The `proj` operator extracts from each TripleGroup, the required triple component from the triple matching the triple pattern. From our example data, `proj`$_{?hpage}(TG)$ =$\{www.vendors.org/V1\}$.

(**`filter`**) The `filter` operator is used for *value-based filtering* i.e. to check if the TripleGroups satisfy the given filter condition. For our example data, `filter`$_{price>500}$ (TG) would eliminate the TripleGroup ntg in Fig. 3 (b) since the triple ($\&Offer1$, *price*, 108) does not satisfy the filter condition.

(**`groupfilter`**) The `groupfilter` operation is used for *structure-based filtering* i.e. to retain only those TripleGroups that satisfy the required query sub structure. For example, the `groupfilter` operator can be used to eliminate TripleGroups like tg in Fig. 3 (a), that are structurally incomplete with respect to the equivalence class $TG_{\{label,country,homepage,mbox\}}$.

(**`join`**) The join expression `join`($?v_{tp_x}$:TG_x,$?v_{tp_y}$:TG_y) computes the join between a TripleGroup tg_x in equivalence class TG_x with a TripleGroup tg_y in equivalence class TG_y based on the given triple patterns. The triple patterns tp_x and tp_y share a common variable $?v$ at O or S component. The result of an object-subject (O-S) join is a nested TripleGroup in which tg_y is nested at the O component of the join triple in tg_x. For example, Fig. 6 shows the nested TripleGroup resulting from the `join` operation between equivalence classes $TG_{\{price,validTo,vendor,product\}}$ and $TG_{\{label,country,homepage\}}$ that join based on triple patterns $\{?o\ vendor\ ?v\}$ and $\{?v\ country\ ?vcountry\}$ respectively. For object-object (O-O) joins, the `join` operator computes a TripleGroup by union of triples in the individual TripleGroups.

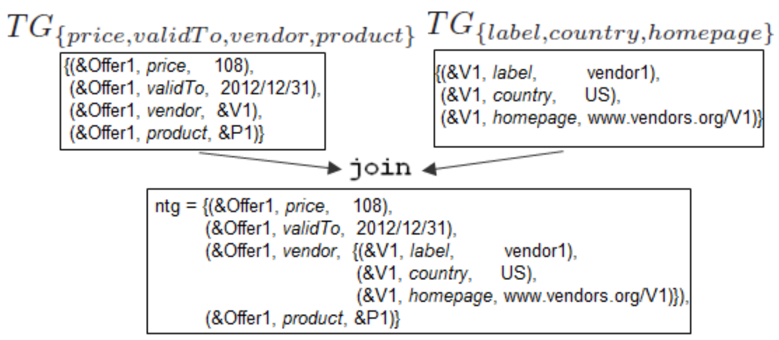

$TG_{\{price,validTo,vendor,product\}}$
{(&Offer1, price, 108),
(&Offer1, validTo, 2012/12/31),
(&Offer1, vendor, &V1),
(&Offer1, product, &P1)}
$TG_{\{label,country,homepage\}}$
{(&V1, label, vendor1),
(&V1, country, US),
(&V1, homepage, www.vendors.org/V1)}
join
ntg = {(&Offer1, price, 108),
(&Offer1, validTo, 2012/12/31),
(&Offer1, vendor, {(&V1, label, vendor1),
(&V1, country, US),
(&V1, homepage, www.vendors.org/V1)}),
(&Offer1, product, &P1)}

Fig. 6. Example join operation in NTGA

Execution plan using NTGA and its mapping to Relational Algebra. TripleGroup-based pattern matching for a query with n star sub patterns, compiles into a MapReduce flow with n MR cycles as shown in Fig. 7. The same query executes in double the number of MR cycles $(2n - 1)$ using Pig approach. Fig. 7 shows the equivalence between NTGA and relational algebra operators based on our notion of *content equivalence*. This mapping suggests rules for lossless transformation between queries written in relational algebra and NTGA. First, the input triples are loaded and the triples that are not part of the query pattern are filtered out. Pig `load` and `filter` operators are *coalesced* into a `loadFilter` operator to minimize costs of repeated data handling. The graph patterns are then evaluated using the NTGA operators, (i) star-joins using Pig's `GROUP BY` operator, which is *coalesced* with the NTGA `groupFilter` operator to enable *structure-based filtering* (represented as `StarGroupFilter`) and, (ii) chain joins on TripleGroups using the NTGA `join` operator (represented as `RDFJoin`). The final result can be converted back to n-tuples using the NTGA `flatten` operator. In general, TripleGroups resulting from any of the NTGA operations can be mapped to Pig's tupled results using the `flatten` operator. For example, the `StarGroupFilter` operation results in a set of TripleGroups. Each TripleGroup can be transformed to an equivalent n-tuple resulting from relational star-joins $SJ1$, $SJ2$, or $SJ3$.

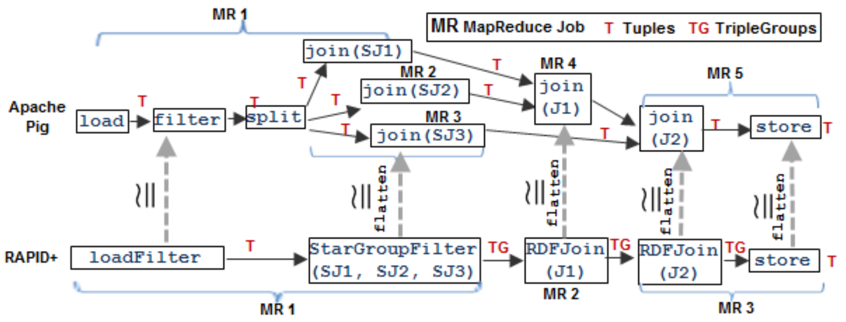
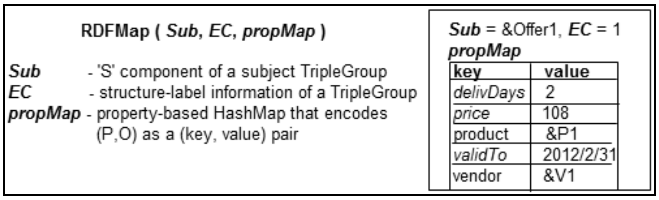


Fig. 7. NTGA execution plan and mapping to Relation Algebra

3.2 RAPID+: Integrating NTGA Operators into Pig

Data Structure for TripleGroups - RDFMap. Pig Latin data model supports a bag data structure that can be used to capture a TripleGroup. The Pig data bag is implemented as an array list of tuples and provides an iterator to process them. Consequently, implementing NTGA operators such as `filter`, `groupfilter`, `join` etc. using this data structure requires an iteration through the data bag which is expensive. For example, given a graph pattern with a set of triple patterns TP and a data graph represented as a set of TripleGroups TG, the `groupfilter` operator requires matching each triple pattern in TP with each tuple t in each TripleGroup $tg \in TG$. This results in the cost of the `groupfilter` operation being $O(|TP|*|tg|*|TG|)$. In addition, representing triples as 3-tuple (s, p, o) results in redundant $s(o)$ components for subject (object) TripleGroups. We propose a specialized data structure called *RDFMap* targeted at efficient implementation of NTGA operators. Specifically it enables, (i) efficient lookup of triples matching a given triple pattern, (ii) compact representation of intermediate results, and (iii) ability to represent structure-label information for TripleGroups. RDFMap is an extended HashMap that stores a mapping from property to object values. Since subject of triples in a TripleGroup are often repeated, RDFMap avoids this redundancy by using a single field Sub to represent the subject component. The field EC captures the structure-label (equivalence class mapped to numbers). Fig. 8. shows the RDFMap corresponding to the Subject TripleGroup in Fig. 3 (a). Using this representation model, a nested TripleGroup can be supported using a nested $propMap$ which contains another RDFMap as a value. The $propMap$ provides a property-based indexed structure that eliminates the need to iterate through the tuples in each bag. Since $propMap$ is hashed on the P component of the triples, matching a triple pattern inside a TripleGroup can now be computed in time $O(1)$. Hence, the cost of the `groupfilter` operation is reduced to $O(|P|*|TG|)$.


RDFMap (*Sub*, *EC*, *propMap*)

Sub	- 'S' component of a subject TripleGroup
EC	- structure-label information of a TripleGroup
propMap	- property-based HashMap that encodes (P,O) as a (key, value) pair

Sub = &Offer1, ***EC*** = 1
propMap

key	value
delivDays	2
price	108
product	&P1
validTo	2012/2/31
vendor	&V1

Fig. 8. RDFMap representing a subject TripleGroup

Implementing NTGA operators using RDFMap. In this section, we show how the property-based indexing scheme of an RDFMap can be exploited for efficient implementation of the NTGA operations. We then discuss the integration of NTGA operators into Pig.

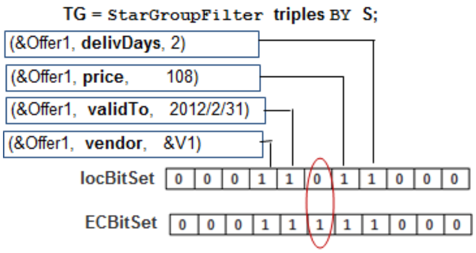
`StarGroupFilter`. A common theme in our implementation is to *coalesce* operators where possible in order to minimize the costs of parameter passing, and context switching between methods. The `starGroupFilter` is one such operator, which coalesces the NTGA `groupfilter` operator into Pig's relational `GROUP BY` operator. Creating subject TripleGroups using this operator can be expressed as:

$$TG = \texttt{StarGroupFilter}\text{ triples by } S$$

The corresponding *map* and *reduce* functions for the `StarGroupFilter` operator are shown in Algorithm 1. In the *map* phase, the tuples are annotated based on the S component analogous to the *map* of a `GROUP BY` operator. In the *reduce* function, the different tuples sharing the same S component are packaged into an *RDFMap* that corresponds to a subject TripleGroup. The `groupfilter` operator is integrated for *structure-based filtering* based on the query sub structures (equivalence classes). This is achieved using global bit patterns (stored as BitSet) that concisely represent the property types in each equivalence class. As the tuples are processed in the *reduce* function, the local BitSet keeps track of the property types processed (line 6). After processing all tuples in a group, if the local BitSet (`locBitSet`) does not match the global BitSet (`ECBitSet`), the structure is incomplete and the group of tuples is eliminated (lines 10-11). Fig. 9 shows the mismatch between the `locBitSet` and `ECBitSet` in the sixth position that represents the missing property "product" belonging to the equivalence class $TG_{\{price, validTo, delivDays, vendor, product\}}$. If the bit patterns match, a labeled *RDFMap* is generated (line 13) whose $propMap$ contains all the (p,o) pairs representing the edges of the star sub graph. The output of `StarGroupFilter` is a single relation containing a list of RDFMaps corresponding to the different star sub graphs in the input data.

Algorithm 1. StarGroupFilter

```
1  Map (Tuple tup(s,p,o))
2  return (s,tup)
3  Reduce (Key, List of tuples T)
4  foreach tup(s,p,o) in T do
5      Initialize: Sub ← s; EC ← findEC(p)
6      locBitset.set(p)
7      propMap.put(p,o)
8  end foreach
9  ECBitSet ← global BitSet for EC
10 if locBitset != ECBitSet then
11     return null
12 else
13     return new RDFMap(Sub, EC, propMap)
14 end if
```



TG = StarGroupFilter triples BY S;
(&Offer1, delivDays, 2)
(&Offer1, price, 108)
(&Offer1, validTo, 2012/2/31)
(&Offer1, vendor, &V1)
locBitSet 0 0 0 1 1 0 1 1 0 0 0
ECBitSet 0 0 0 1 1 1 1 1 0 0 0

Fig. 9. Structure-based filtering of TripleGroups

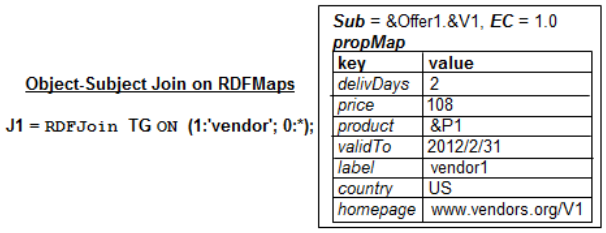
Algorithm 2. RDFJoin

```
1  Map (RDFMap rMap)
2  if joinKey == * then
3      return (rMap.Sub, rMap)
4  else
5      key ← rMap.propMap.get(joinKey)
6      return (key, rMap)
7  end if
8  Reduce (Key, List of RDFMaps R)
9  foreach rMap in R do
10     if rMap.EC == EC1 then
11         list1.add(rMap)
12     else if rMap.EC == EC2 then
13         list2.add(rMap)
14     end if
15 end foreach
16 foreach outer in list1 do
17     foreach inner in list2 do
18         propMapNew ←
           joinProp(outer.propMap,
           inner.propMap)
19         ECNew ← joinSub(outer.EC,
           inner.EC)
20         SubNew ← joinSub(outer.Sub,
           inner.Sub)
21         rMapNew ← new
           RDFMap(SubNew, ECNew,
           propMapNew)
22         resultList.add(rMapNew)
23     end foreach
24 end foreach
25 return resultList
```

`RDFJoin`: The `RDFJoin` operator takes as input a single relation containing RDFMaps, and computes the NTGA `join` between star patterns as described by Algorithm 2. The O-S join $J1$ between the star patterns in Fig. 1 can be expressed as follows:

$$J1 = \texttt{RDFJoin}\ TG \text{ on } (1\text{:'}vendor\text{'}, 0\text{:*});$$

where joins on O are specified using the property types, and joins on S are specified as a '*'. In the *map* phase, the RDFMaps are annotated based on the join key corresponding to their equivalence class (lines 2-8). In the *reduce* phase, the RDFMaps that join are packaged into a new RDFMap which corresponds to a nested TripleGroup. The EC of the new joined RDFMap is a function of the EC of the individual RDFMaps. For example, the `RDFJoin` between TripleGroups shown in Fig. 6, results in an RDFMap whose $propMap$ contains the union of triples from the individual TripleGroups as shown in Fig. 10. In our implementation, the Sub field is a concatenation of the Sub fields of the individual TripleGroups e.g. $\&Offer1.\&V1$. The join result is optimized by eliminating the (p, o) pair corresponding to the join triple if it is no longer required. This reduces the size of the intermediate RDFMaps after each MR cycle. Our join result corresponds to an unnested joined TripleGroup, as shown in Fig. 4 (a).



Object-Subject Join on RDFMaps

J1 = RDFJoin TG ON (1:'vendor'; 0:*);

Sub = &Offer1.&V1, **EC** = 1.0

propMap

key	**value**
delivDays	2
price	108
product	&P1
validTo	2012/2/31
label	vendor1
country	US
homepage	www.vendors.org/V1

Fig. 10. Example RDFMap after RDFJoin operation

4 Evaluation

Our goal was to empirically evaluate the performance of NTGA operators with respect to pattern matching queries involving combinations of star and chain joins. We compared the performance of RAPID+ with two implementations of Pig, (i) the naive Pig with the VP storage model, and (ii) an optimized implementation of Pig (Pig_{opt}), in which we introduced additional project operations to eliminate the redundant join columns. Our evaluation tasks included, (i) **Task1 -** Scalability of TripleGroup-based approach with size of RDF graphs, (ii) **Task2 -** Scalability of TripleGroup-based pattern matching with denser star patterns, and (iii) **Task3 -** Scalability of NTGA operators with increasing cluster sizes.

Table 1. Testbed queries and performance gain of RAPID+ over Pig (10-node cluster / 32GB)

Query	#Triple Pattern	#Edges in Stars	%gain	Query	#Triple Pattern	#Edges in Stars	%gain
Q1	3	1:2	56.8	Q6	8	4:4	58.4
Q2	4	2:2	46.7	Q7	9	5:4	58.6
Q3	5	2:3	47.8	Q8	10	6:4	57.3
Q4	6	3:3	51.6	2S1C	6	2:4	65.4
Q5	7	3:4	57.4	3S2C	10	2:4:4	61.5

4.1 Setup

Environment: The experiments were conducted on VCL[3], an on-demand computing and service-oriented technology that provides remote access to virtualized resources. Nodes in the clusters had minimum specifications of single or duo core Intel X86 machines with 2.33 GHz processor speed, 4G memory and running Red Hat Linux. The experiments were conducted on 5-node clusters with block size set to 256MB. Scalability testing was done on clusters with 10, 15, 20, and 25 nodes. Pig release 0.7.0 and Hadoop 0.20 were used. All results recorded were averaged over three trials.

Testbed - Dataset and Queries: Synthetic datasets (n-triple format) generated using the BSBM tool were used. A comparative evaluation was carried out based on size of data ranging from 8.6GB (approx. 35 million triples) at the lower end, to a data size of 40GB (approx. 175 million triples). 10 queries (shown in Table 1) adapted from the BSBM benchmark (Explore use case) with at least a star and chain join were used. The evaluation tested the effect of query structure on performance with, (i) $Q1$ to $Q8$ consisting of two star patterns with varying cardinality, (ii) $2S1C$ consisting of two star patterns, a chain join, and a filter component (6 triple patterns), and (ii) $3S2C$ consisting of three star patterns, two chain joins, and a filter component (10 triple patterns). Query details and additional experiment results are available on the project website[4].

4.2 Experiment Results

Task1: Fig. 11 (a) shows the execution times of the three approaches on a 5-node cluster for $2S1C$. For all the four data sizes, we see a good percentage improvement in the execution times for RAPID+. The two star patterns in $2S1C$ are computed in two separate MR cycles in both the Pig approaches, resulting in the query compiling into a total of three MR cycles. However, RAPID+ benefits by the grouping-based join algorithm (`StarGroupFilter` operator) that computes the star patterns in a single MR cycle, thus reducing one MR cycle in total. We also observe cost savings due to the integration of `loadFilter` operator in RAPID+ that coalesces the `LOAD` and `FILTER` phases. As expected, the Pig_{opt} performs better than the naive Pig approach due to the decrease in the size of the intermediate results.

[3] `https://vcl.ncsu.edu/`

[4] `http://research.csc.ncsu.edu/coul/RAPID/ESWC_exp.htm`

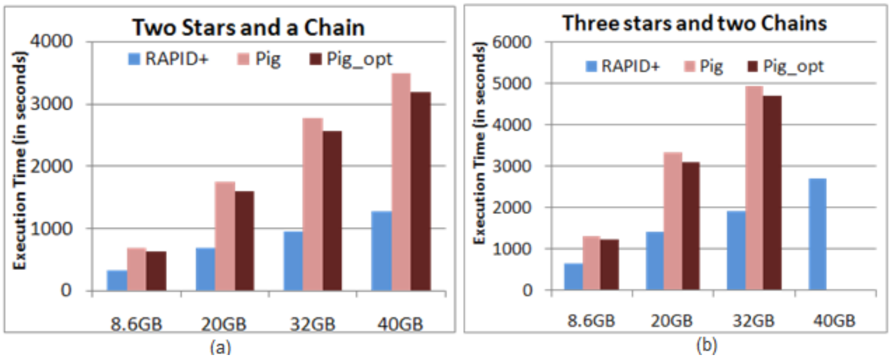


Fig. 11. Cost analysis on 5-node cluster for (a) 2SIC (b) 3S2C

Fig. 11 (b) shows the performance comparison of the three approaches on a 5-node cluster for $3S2C$. This query compiles into three MR cycles in RAPID+ and five MR cycles in Pig / Pig_{opt}. We see similar results with RAPID+ outperformed the Pig based approaches, achieving up to 60% performance gain with the 32GB dataset. The Pig based approaches did not complete execution for the input data size of 40GB. We suspect that this was due to the large sizes of intermediate results. In this situation, the compact representation format offered by the RDFMap proved advantageous to the RAPID+ approach. In the current implementation, RAPID+ has the overhead that the computation of the star patterns results in a single relation containing TripleGroups belonging to different equivalence classes. In our future work, we will investigate techniques for delineating different types of intermediate results.

Task2: Table 1 summarizes the performance of RAPID+ and Pig for star-join queries with varying edges in each star sub graph. NTGA operators achieve a performance gain of 47% with $Q2$ (2:2 cardinality) which increases with denser star patterns, reaching 59% with $Q8$ (6:4 cardinality). In addition to the savings in MR cycle in RAPID+, this demonstrates the cost savings due to smaller intermediate relations achieved by eliminating redundant subject values and join triples that are no longer required. Fig. 12 (b) shows a comparison on a 5-node cluster (20GB data size) with Pig_{opt} which eliminates join column redundancy in Pig, similar to RDFMap's concise representation of subjects within a TripleGroup. RAPID+ maintains a consistent performance gain of 50% across the varying density of the two star patterns.

Task3: Fig. 12(a) shows the scalability study of $3S2C$ on different sized clusters, for 32GB data. RAPID+ starts with a performance gain of about 56% with the 10-node cluster, but its advantage over Pig and Pig_{opt} reduces with increasing number of nodes. The increase in the number of nodes, decreases the size of data processed by each node, therefore reducing the probability of disk spills with the SPLIT operator in the Pig based approaches. However, RAPID+ still consistently outperforms the Pig based approaches with at least 45% performance gain in all experiments.

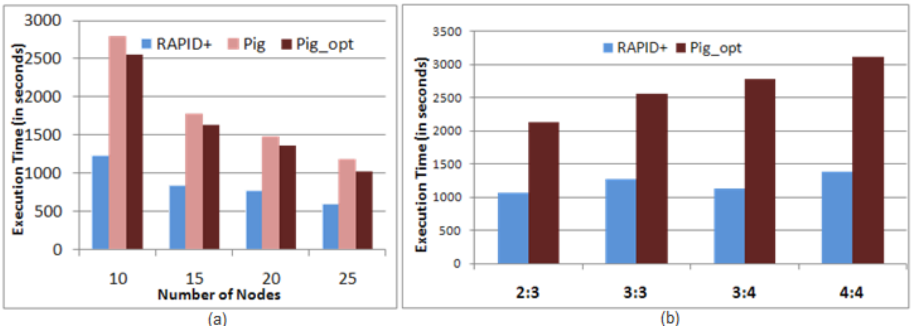


Fig. 12. Scalability study for (a) 3S2C varying cluster sizes (b) two stars with varying cardinality

5 Related Work

Data Models and High-Level Languages for cluster-based environment. There has been a recent proliferation of data flow language such as Sawzall [11], DryadLINQ [12], HiveQL [7], and Pig Latin [6] for processing structured data on parallel data processing systems such as Hadoop. Another such query language, JAQL[5] is designed for semi-structure data analytics, and uses the (key, value) JSON model. However, this model splits RDF sub graphs into different bags, and may not be efficient to execute bushy plans. Our previous work, RAPID [13] focused on optimizing analytical processing of RDF data on Pig. RAPID+ [10] extended Pig with UDFs to enable TripleGroup-based processing. In this work, we provide formal semantics to integrate TripleGroups as first-class citizens, and present operators for graph pattern matching.

RDF Data processing on MapReduce Platforms. MapReduce framework has been explored for scalable processing of Semantic Web data. For reasoning tasks, specialized map and reduce functions have been defined based on RDFS rules [3] and the OWL Horst rules [14], for materializing the closure of RDF graphs. Yet another work [15] extends Pig by integrating schema-aware RDF data loader and embedding reasoning support into the existing framework. For scalable pattern matching queries, there have been MapReduce-based storage and query systems [2],[1] that process RDFMolecules. Also, [16] uses HadoopDB [17] with a column-oriented database to support a scalable Semantic Web application. This framework enables parallel computation of star-joins if the data is partitioned based on the Subject component. However, graph pattern queries with multiple star patterns and chain join may not benefit much. Another recent framework [4] pre-processes RDF triples to enable efficient querying of billions of triples over HDFS. We focus on ad hoc processing of RDF graphs that cannot presume pre-processed or indexed data.

Optimizing Multi-way Joins. RDF graph pattern matching typically involves several join operations. There have been optimization techniques [9] to re-write SPARQL queries into small-sized star-shaped groups and generate bushy plans using two physical join operators called njoin and gjoin. It is similar in spirit to the work presented

[5] `http://code.google.com/p/jaql`

here since both exploit star-shaped sub patterns. However, our work focuses on parallel platforms and uses a grouping-based algorithm to evaluate star-joins. There has been work on optimizing m-way joins on structured relations like slice join [18]. However, we focus on joins involving RDF triples for semi-structured data. Another approach [19] efficiently partitions and replicates of tuples on reducer processes in a way that minimizes the communication cost. This is complementary to our approach and the partitioning schemes could further improve the performance of join operations. [20], investigates several join algorithms which leverage pre-processing techniques on Hadoop, but mainly focus on log processing. RDFBroker [21] is a RDF store that is based on the concept of a *signature* (set of properties of a resource), similar to NTGA's *structure-labeling* function λ. However, the focus of [21] is to provide a natural way to map RDF data to database tables, without presuming schema knowledge. Pregel [22] and Signal/Collect [23] provide graph-oriented primitives as opposed to relational algebra type operators, and also target parallel platforms. The latter is still in a preliminary stage and has not completely demonstrated its advantages across parallel platforms.

6 Conclusion

In this paper, we presented an intermediate algebra (NTGA) that enables more natural and efficient processing for graph pattern queries on RDF data. We proposed a new data representation format (RDFMap) that supports NTGA operations in a more efficient manner. We integrated these NTGA operators into Pig, and presented a comparative performance evaluation with the existing Pig implementation. For certain classes of queries, we saw a performance gain of up to 60%. However, there might be certain scenarios in which it may be preferable not to compute all star patterns. In such cases, we need a hybrid approach that utilizes cost-based optimization techniques to determine when the NTGA approach is the best. We will also investigate a more efficient method for dealing with heterogeneous TripleGroups resulting from join operations.

References

1. Newman, A., Li, Y.F., Hunter, J.: Scalable Semantics: The Silver Lining of Cloud Computing. In: IEEE International Conference on eScience (2008)
2. Newman, A., Hunter, J., Li, Y., Bouton, C., Davis, M.: A Scale-Out Rdf Molecule Store for Distributed Processing of Biomedical Data. In: Semantic Web for Health Care and Life Sciences Workshop (2008)
3. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
4. Husain, M., Khan, L., Kantarcioglu, M., Thuraisingham, B.: Data Intensive Query Processing for Large Rdf Graphs Using Cloud Computing Tools. In: IEEE International Conference on Cloud Computing, CLOUD (2010)
5. Dean, J., Ghemawat, S.: Simplified Data Processing on Large Clusters. ACM Commun. 51, 107–113 (2008)
6. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: Proc. International Conference on Management of data (2008)

7. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: A Warehousing Solution over a Map-Reduce Framework. Proc. VLDB Endow. 2, 1626–1629 (2009)
8. Neumann, T., Weikum, G.: The Rdf-3X Engine for Scalable Management of Rdf Data. The VLDB Journal 19, 91–113 (2010)
9. Vidal, M.-E., Ruckhaus, E., Lampo, T., Martínez, A., Sierra, J., Polleres, A.: Efficiently Joining Group Patterns in SPARQL Queries. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 228–242. Springer, Heidelberg (2010)
10. Ravindra, P., Deshpande, V.V., Anyanwu, K.: Towards Scalable Rdf Graph Analytics on Mapreduce. In: Proc. Workshop on Massive Data Analytics on the Cloud (2010)
11. Pike, R., Dorward, S., Griesemer, R., Quinlan, S.: Interpreting the Data: Parallel Analysis with Sawzall. Sci. Program. 13, 277–298 (2005)
12. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P.K., Currey, J.: Dryadlinq: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language. In: Proc. USENIX Conference on Operating Systems Design and Implementation (2008)
13. Sridhar, R., Ravindra, P., Anyanwu, K.: RAPID: Enabling Scalable Ad-Hoc Analytics on the Semantic Web. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 715–730. Springer, Heidelberg (2009)
14. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: OWL Reasoning with Webpie: Calculating the Closure of 100 Billion Triples. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 213–227. Springer, Heidelberg (2010)
15. Tanimura, Y., Matono, A., Lynden, S., Kojima, I.: Extensions to the Pig Data Processing Platform for Scalable Rdf Data Processing using Hadoop. In: IEEE International Conference on Data Engineering Workshops (2010)
16. Abouzied, A., Bajda-Pawlikowski, K., Huang, J., Abadi, D.J., Silberschatz, A.: Hadoopdb in Action: Building Real World Applications. In: Proc. International Conference on Management of data (2010)
17. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., Rasin, A.: Hadoopdb: an Architectural Hybrid of Mapreduce and Dbms Technologies for Analytical Workloads. Proc. VLDB Endow. 2, 922–933 (2009)
18. Lawrence, R.: Using Slice Join for Efficient Evaluation of Multi-Way Joins. Data Knowl. Eng. 67, 118–139 (2008)
19. Afrati, F.N., Ullman, J.D.: Optimizing Joins in a Map-Reduce Environment. In: Proc. International Conference on Extending Database Technology (2010)
20. Blanas, S., Patel, J.M., Ercegovac, V., Rao, J., Shekita, E.J., Tian, Y.: A Comparison of Join Algorithms for Log Processing in Mapreduce. In: Proc. International Conference on Management of data (2010)
21. Sintek, M., Kiesel, M.: RDFBroker: A Signature-Based High-Performance RDF Store. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 363–377. Springer, Heidelberg (2006)
22. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: A System for Large-Scale Graph Processing. In: Proc. International Conference on Management of data (2010)
23. Stutz, P., Bernstein, A., Cohen, W.: Signal/Collect: Graph Algorithms for the (Semantic) Web. 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. 764–780. Springer, Heidelberg (2010)

Query Relaxation for Entity-Relationship Search

Shady Elbassuoni, Maya Ramanath, and Gerhard Weikum

Max-Planck Institute for Informatics
{elbass,ramanath,weikum}@mpii.de

Abstract. Entity-relationship-structured data is becoming more important on the Web. For example, large knowledge bases have been automatically constructed by information extraction from Wikipedia and other Web sources. Entities and relationships can be represented by subject-property-object triples in the RDF model, and can then be precisely searched by structured query languages like SPARQL. Because of their Boolean-match semantics, such queries often return too few or even no results. To improve recall, it is thus desirable to support users by *automatically relaxing* or reformulating queries in such a way that the intention of the original user query is preserved while returning a sufficient number of ranked results.

In this paper we describe comprehensive methods to relax SPARQL-like triple-pattern queries in a fully automated manner. Our framework produces a set of relaxations by means of statistical language models for structured RDF data and queries. The query processing algorithms merge the results of different relaxations into a unified result list, with ranking based on any ranking function for structured queries over RDF-data. Our experimental evaluation, with two different datasets about movies and books, shows the effectiveness of the automatically generated relaxations and the improved quality of query results based on assessments collected on the Amazon Mechanical Turk platform.

1 Introduction

1.1 Motivation

There is a trend towards viewing Web or digital-library information in an entity-centric manner: what is the relevant information about a given sports club, a movie star, a politician, a company, a city, a poem, etc. Moreover, when querying the Web, news, or blogs, we like the search results to be organized on a per-entity basis. Prominent examples of this kind of search are entitycube.research.microsoft.com or google.com/squared/. Additionally, services that contribute towards more semantic search are large knowledge repositories, including both handcrafted ones such as freebase.com as well as automatically constructed ones such as trueknowledge.com or dbpedia.org. These have been enabled by knowledge-sharing communities such as Wikipedia and by advances in information extraction (e.g., [2, 6, 17, 23, 20]).

One way of representing entity-centric information, along with structured relationships between entities, is the Semantic-Web data model RDF. An RDF collection consists of a set of subject-property-object (SPO) triples. Each triple is a pair of entities with a named relationship. A small example about books is shown in Table 1.

Table 1. RDF triples

Subject (S)	**Property (P)**	**Object (O)**
Carl_Sagan	wrote	Contact
Carl_Sagan	type	American_Writer
Carl_Sagan	type	Astronomer
Carl_Sagan	bornIn	USA
Carl_Sagan	wonAward	Pulitzer_Prize
Contact	type	novel
Contact	hasGenre	Science_Fiction
Contact	hasTag	aliens
Contact	hasTag	philosopy
Jon_Krakauer	wrote	Into_the_Wild
Into_the_Wild	type	biography
Into_the_Wild	hasTag	adventure
Into_the_Wild	hasTag	wilderness
Jon _Krakauer	hasBestseller	Into_Thin_Air
Jon _Krakauer	citizenOf	USA

RDF data of this kind can be queried using a conjunction of *triple patterns* – the core of SPARQL – where a triple pattern is a triple with variables and the same variable in different patterns denotes a join condition. For example, searching for Pulitzer-prize winning science fiction authors from the USA could be phrased as:

```
?a wrote ?b ; ?b hasGenre Science_Fiction ;
?a wonAward Pulitzer_Prize ; ?a bornIn USA
```

This query contains a conjunction (denoted by ";") of four triple patterns where ?a and ?b denote variables that should match authors and their books respectively.

While the use of triple patterns enables users to formulate their queries in a precise manner, it is possible that the queries are overly constrained and lead to unsatisfactory recall. For example, this query would return very few results even on large book collections, and only one - Carl Sagan - for our example data. However, if the system were able to automatically reformulate one or more conditions in the query, say, replacing bornIn with citizenOf, the system would potentially return a larger number of results.

1.2 Query Relaxation Problem

This paper addresses the *query relaxation* problem: automatically broadening or reformulating triple-pattern queries to retrieve more results without unduly sacrificing precision. We can view this problem as the entity-relationship-oriented counterpart of query expansion in the traditional keyword-search setting. Automatically expanding queries in a robust way so that they would not suffer from topic drifts (e.g., overly broad generalizations) is a difficult problem [3].

The problem of query relaxation for triple-pattern queries has been considered in limited form in [22, 11, 7, 10] and our previous work [8]. Each of these prior approaches focused on very specific aspects of the problem, and only two of them [22, 8] conducted experimental studies on the effectiveness of their proposals. These techniques are discussed in more detail in Sect. 5.

1.3 Our Approach

This paper develops a comprehensive set of query relaxation techniques, where the relaxation candidates can be derived from both the RDF data itself as well as from external ontological and textual sources. Our framework is based on statistical language models (LMs) and provides a principled basis for generating relaxed query candidates. Moreover, we develop a model for holistically ranking results of both the original query and different relaxations into a unified result list.

Our query relaxation framework consists of the following three types of relaxations:

- Entities (subject, object) and relations (property) specified in a triple pattern are relaxed by substituting with related entities and relations. For example, bornIn could be substituted with citizenOf or livesIn and Pulitzer_Prize could be replaced by Hugo_Award or Booker_Prize.
- Entities and relations specified in a triple pattern could be substituted with variables. For example, Pulitzer_Prize could be replaced by ?p to cover arbitrary awards or wonAward could be replaced by ?r, allowing for matches such as nominatedFor and shortlistedFor.
- Triple patterns from the entire query could be either removed or made optional. For example, the triple pattern ?b hasGenre Science_Fiction could be removed entirely, thus increasing the number of authors returned.

The technical contributions of this paper are the following:

- We develop a novel, comprehensive framework for different kinds of query relaxation, in an RDF setting, based on language modeling techniques. Our framework can incorporate external sources such as ontologies and text documents to generate candidate relaxations. Our relaxation framework is described in Sect. 2.
- We develop a general ranking model that combines the results of the original and relaxed queries and utilizes relaxation weights for computing, in a principled manner, query-result rankings. The ranking model is described in Sect. 3.
- We evaluate our model and techniques with two datasets – movie data from imdb.com and book information from the online community librarything.com, and show that our methods provide very good results in terms of NDCG. The results of our user study are reported in Sect. 4.

2 Relaxation Framework

We start by describing the basic setting and some of the terminology used in the rest of this paper.

Knowledge Base. A knowledge base KB of entities and relations is a set of *triples*, where a triple is of the form $\langle e1, r, e2\rangle$ with entities $e1$, $e2$ and relation r (or $\langle s, p, o\rangle$

with subject s, object o, and property p in RDF terminology). An example of such a triple is: Carl_Sagan wrote Contact.

Queries. A query consists of *triple patterns* where a triple pattern is a triple with at least 1 variable. For example, the query "science-fiction books written by Carl Sagan" can be expressed as: Carl_Sagan wrote ?b; ?b hasGenre Science_Fiction consisting of 2 triple patterns.

Given a query with k triple patterns, the result of the query is the set of all k-tuples that are isomorphic to the query when binding the query variables with matching entities and relations in KB. For example, the results for the example query includes the 2-tuple Carl_Sagan wrote Contact;Contact hasGenre Science_Fiction.

2.1 Relaxation Strategy

As mentioned in the introduction, we are interested in three types of relaxations: i) replacing a constant (corresponding to an entity or a relation) in one or more triple patterns of the query with another constant which still reflects the user's intention, ii) replacing a constant in one or more triple patterns with a variable, and iii) removing a triple pattern altogether. In the rest of this section, we describe a framework which incorporates all three relaxations in a holistic manner. Specific details about how the knowledge-base is utilized in the framework are described in Sect. 2.2.

Finding Similar Entities[1]. For each entity $\mathsf{E_i}$ in the knowledge base KB, we construct a document $D(\mathsf{E_i})$ (the exact method of doing so will be described in Sect. 2.2). For each document $D(\mathsf{E_i})$, let $LM(\mathsf{E_i})$ be its language model. The similarity between two entities $\mathsf{E_i}$ and $\mathsf{E_j}$ is now computed as the distance between the LMs of the corresponding documents. Specifically, we use the square-root of the Jensen-Shannon divergence (JS-divergence) between two probability distributions (that is, $LM(\mathsf{E_i})$ and $LM(\mathsf{E_j})$, in this case), which is a metric. And so, for an entity of interest E, we can compute a ranked list of similar entities.

Replacing Entities with Variables. We interpret replacing an entity in a triple pattern with a variable as being equivalent to replacing that entity with *any* entity in the knowledge base.

We first construct a special document for the entire knowledge base, $D(\mathsf{KB})$. Let E be the entity of interest (i.e., the entity in the triple pattern to be replaced with a variable). Let $D(\mathsf{E})$ be its document. Now, we construct a document corresponding to "any" entity other than E as: $D(\mathsf{ANY}) = D(\mathsf{KB}) - D(\mathsf{E})$ (i.e., remove the contents of $D(\mathsf{E})$ from $D(\mathsf{KB})$). The similarity between the entity E and "any" entity ANY is computed as the distance between the LMs of their corresponding documents.

In the ranked list of potential replacements for entity E, a variable replacement is now simply another candidate. In other words, the candidates beyond a certain rank are so dissimilar from the given entity, that they may as well be ignored and represented by a single variable.

[1] Note that, even though we refer only to entities in the following, the same applies to relations as well.

Removing Triple Patterns. So far, we gave a high-level description of our relaxation technique for individual entities (or relations) in a triple pattern, without treating the triple pattern holistically. In a given query containing multiple triple patterns, a large number of relaxed queries can be generated by systematically substituting each constant with other constants or variables. We now consider the case when a triple pattern in the query contains only variables. In a post-processing step, we can now choose one of the following options. First, the triple pattern can be made optional. Second, the triple pattern can be removed from the query. To illustrate the two cases, consider the following examples.

Example 1: Consider the query asking for married couples who have acted in the same movie: ?a1 actedIn ?m; ?a2 actedIn ?m; ?a1 marriedTo ?a2 and a relaxation: ?a1 actedIn ?m; ?a2 ?r ?m; ?a1 marriedTo ?a2. Even though the second triple pattern contains only variables, retaining this pattern in the query still gives the user potentially valuable information – that ?a2 was related some how to the movie ?m. Hence, instead of removing this triple pattern from the query, it is only made optional – that is, a result may or may not have a triple which matches this triple pattern.

Example 2: Consider the query asking for movies which James Cameron produced, directed as well as acted in: James_Cameron produced ?m; James_Cameron directed ?m; James_Cameron actedIn ?m and a relaxation: ?x ?r ?m; James_Cameron directed ?m; James_Cameron actedIn ?m. In this case, the first triple pattern matches any random fact about the movie ?m. This does not give any valuable information to the user as in the previous case and can be removed.

2.2 Constructing Documents and LMs

We now describe how to construct documents for entities and relations and how to estimate their corresponding LMs.

Sources of Information. The document for an entity should "describe" the entity. There are at least three different sources of information which we can leverage in order to construct such a document. First, we have the knowledge base itself – this is also the primary source of information in our case since we are processing our queries on the knowledge base. Second, we could make use of external textual sources – for example, we could extract contextual snippets or keywords from text/web documents from which the triples were extracted. Third, we could also utilize external ontologies such as Wordnet, in order to find terms which are semantically close to the entity.

In this paper, our main focus is on utilizing the knowledge base as the information source and hence, we describe our techniques in this context and perform experiments using these techniques. But, our framework can be easily extended to incorporate other information sources and we briefly describe how this can be done at the end of this section.

Documents and LMs for entities. Let E be the entity of interest and $D(\mathsf{E})$ be its document, which is constructed as the set of all triples in which E occurs either as a subject or an object. That is,

$$D(\mathsf{E}) = \{\langle \mathsf{E\ r\ o}\rangle : \langle \mathsf{E\ r\ o}\rangle \in \mathsf{KB}\} \cup \{\langle \mathsf{s\ r\ E}\rangle : \langle \mathsf{s\ r\ E}\rangle \in \mathsf{KB}\}$$

We now need to define the set of terms over which the LM is estimated. We define two kinds of terms: i) “unigrams” U, corresponding to all entities in KB, and, ii) “bigrams” B, corresponding to all entity-relation pairs. That is,

$$U = \{\mathsf{e} : \langle \mathsf{e\ r\ o} \rangle \in \mathsf{KB} || \langle \mathsf{s\ r\ e} \rangle \in \mathsf{KB}\}$$

$$B = \{(\mathsf{er}) : \langle \mathsf{e\ r\ o} \rangle \in \mathsf{KB}\} \cup \{(\mathsf{re}) : \langle \mathsf{s\ r\ e} \rangle \in \mathsf{KB}\}$$

Example: The entity Woody_Allen would have a document consisting of triples Woody_Allen directed Manhattan, Woody_Allen directed Match_Point, Woody_Allen actedIn Scoop, Woody_Allen type Director, Federico_Fellini influences Woody_Allen, etc. The terms in the document would include Scoop, Match_Point, (type,Director), (Federico_Fellini,influences), etc.

Note that the bi-grams usually occur exactly once per entity, but it is still important to capture this information. When we compare the LMs of two entities, we would like identical relationships to be recognized. For example, if for a given entity, we have the bigram (hasWonAward, Academy_Award), we can then distinguish the case where a candidate entity has the term (hasWonAward, Academy_Award) and the term (nominatedFor, Academy_Award). This distinction cannot be made if only unigrams are considered.

Estimating the LM. The LM corresponding to document $D(\mathsf{E})$ is now a mixture model of two LMs: P_U, corresponding to the unigram LM and P_B, the bigram LM. That is,

$$P_{\mathsf{E}}(w) = \mu P_U(w) + (1 - \mu) P_B(w)$$

where μ controls the influence of each component. The unigram and bigram LMs are estimated in the standard way with linear interpolation smoothing from the corpus. That is,

$$P_U(w) = \alpha \frac{c(w; D(\mathsf{E}))}{\Sigma_{w' \in U} c(w'; D(\mathsf{E}))} + (1 - \alpha) \frac{c(w; D(\mathsf{KB}))}{\Sigma_{w' \in U} c(w'; D(\mathsf{KB}))}$$

where $w \in U$, $c(w; D(\mathsf{E}))$ and $c(w; D(\mathsf{KB}))$ are the frequencies of occurrences of w in $D(\mathsf{E})$ and $D(\mathsf{KB})$ respectively and α is the smoothing parameter. The bigram LM is estimated in an analogous manner.

Documents and LMs for relations. Let R be the relation of interest and let $D(\mathsf{R})$ be its document, which is constructed as the set of all triples in which R occurs. That is,

$$D(\mathsf{R}) = \{\langle \mathsf{s'\ R\ o'} \rangle : \langle \mathsf{s'\ R\ o'} \rangle \in \mathsf{KB}\}$$

As with the case of entities, we again define two kinds of terms – “unigrams” and “bigrams”. Unigrams correspond to the set of all entities in KB. But, we make a distinction here between entities that occur as subjects and those that occur as objects, since the relation is directional (note that there could be entities that occur as both). That is,

$$S = \{\mathsf{s} : \langle \mathsf{s\ r\ o} \rangle \in \mathsf{KB}\}$$

$$O = \{\mathsf{o} : \langle \mathsf{s\ r\ o} \rangle \in \mathsf{KB}\}$$

$$B = \{(\mathsf{so}) : \langle \mathsf{s\ r\ o} \rangle \in \mathsf{KB}\}$$

Example: Given the relation directed, $D(\mathsf{directed})$ would consist of all triples containing that relation, including, James_Cameron directed Aliens, Woody_Allen directed Manhattan, Woody_Allen directed Match_Point, Sam_Mendes directed American_Beauty, etc. The terms in the document would include James_Cameron, Manhattan, Woody_Allen, (James_Cameron, Aliens), (Sam_Mendes, American_Beauty), etc.

Estimating the LM. The LM of $D(\mathsf{R})$ is a mixture model of three LMs: P_S, corresponding to the unigram LM of terms in S, P_O, corresponding to the unigram LM of terms in O and P_B, corresponding to the bigram LM. That is,

$$P_R(w) = \mu_s P_S(w) + \mu_o P_O(w) + (1 - \mu_s - \mu_o) P_B(w)$$

where μ_s, μ_o control the influence of each component. The unigram and bigram LMs are estimated in the standard way with linear interpolation smoothing from the corpus. That is,

$$P_S(w) = \alpha \frac{c(w; D(\mathsf{R}))}{\Sigma_{w' \in S} c(w'; D(\mathsf{R}))} + (1 - \alpha) \frac{c(w; D(\mathsf{KB}))}{\Sigma_{w' \in S} c(w'; D(\mathsf{KB}))}$$

where $w \in S$, $c(w; D(\mathsf{R}))$ and $c(w; D(\mathsf{KB}))$ are the frequencies of occurrences of w in $D(\mathsf{R})$ and $D(\mathsf{KB})$ respectively, and α is a smoothing parameter. The other unigram LM and the bigram LM are estimated in an analogous manner.

Generating the Candidate List of Relaxations. As previously mentioned, we make use of the square root of the JS-divergence as the similarity score between two entities (or relations). Given probability distributions P and Q, the JS-divergence between them is defined as follows,

$$JS(P||Q) = KL(P||M) + KL(Q||M)$$

where, given two probability distributions R and S, the KL-divergence is defined as,

$$KL(R||S) = \Sigma_j R(j) \log \frac{R(j)}{S(j)}$$

and

$$M = \frac{1}{2}(P + Q)$$

2.3 Examples

Table 2 shows example entities and relations from the IMDB and LibraryThing datasets and their top-5 relaxations derived from these datasets, using the techniques described above. The entry *var* represents the variable candidate. As previously explained, a variable substitution indicates that there were no other *specific* candidates which had a high similarity to the given entity or relation. For example, the commentedOn relation has only one specific candidate relaxation above the variable relaxation – hasFriend. Note that the two relations are relations between people - a person X could comment on something a person Y wrote, or a person X could have a friend Y - whereas the remaining relations are not relations between people. When generating relaxed queries using these individual relaxations, we ignore all candidates which occur after the variable. The process of generating relaxed queries will be explained in Sect. 3.

Table 2. Example entities and relations and their top-5 relaxations

LibraryThing		IMDB	
Egypt	**Non-fiction**	**Academy_Award_for_Best_Actor**	**Thriller**
Ancient_Egypt	Politics	BAFTA_Award_for_Best_Actor	Crime
Mummies	American_History	Golden_Globe_Award_for_Best_Actor_Drama	Horror
Egyptian	Sociology	*var*	Action
Cairo	Essays	Golden_Globe_Award_for_Best_Actor_Musical_or_Comedy	Mystery
Egyptology	History	New_York_Film_Critics_Circle_Award_for_Best_Actor	*var*
wrote	**commentedOn**	**directed**	**bornIn**
hasBook	hasFriend	actedIn	livesIn
hasTagged	*var*	created	originatesFrom
var	hasBook	produced	*var*
hasTag	hasTag	*var*	diedIn
hasLibraryThingPage	hasTagged	type	isCitizenOf

2.4 Using Other Information Sources

The core of our technique lies in constructing the document for an entity E or relation R and estimating its LM. And so, given an information source, it is sufficient to describe: i) how the document is constructed, ii) what the terms are, and, iii) how the LM is estimated. In this paper, we have described these three steps when the information source is the knowledge base of RDF triples. It is easy to extend the same method for other sources. For example, for the case of entities, we could make use of a keyword context or the text documents from which an entity or a triple was extracted. Then a document for an entity will be the set of all keywords or a union of all text snippets associated with it. The terms can be any combination of n-grams and the LM is computed using well-known techniques from the IR literature (see for example, entity LM estimation in [16, 18, 15, 9, 21], in the context of entity ranking).

Once individual LMs have been estimated for an entity or a relation from each information source, a straight-forward method to combine them into a single LM is to use a mixture model of all LMs. The parameters of the mixture model can be set based on the importance of each source. Note that this method does not preclude having different subsets of sources for different entities or relations.

3 Relaxing Queries and Ranking Results

We have so far described techniques to construct candidate lists of relaxations for entities and relations. In this section we describe how we generate relaxed queries and how results for the original and relaxed queries are ranked.

3.1 Generating Relaxed Queries

Let $Q_0 = \{q_1, q_2, ..., q_n\}$ be the query, where q_i is a triple pattern. Let the set of relaxed queries be $R = \{Q_1, Q_2, ..., Q_r\}$, where Q_j is a query with one or more of its triple patterns relaxed. A triple pattern q_i is relaxed by relaxing at least one of its constants. Let s_j be the relaxation score of Q_j, computed by adding up the scores of all entity and relation relaxations in Q_j. Recall that an individual entity or relation score is the square root of the JS-divergence between the LMs of the relaxed entity/relation and the original entity/relation. Clearly, the score s_0 for the original query Q_0 is 0 and the queries can be ordered in ascending order of s_js.

Example: Consider the query asking for Academy award winning action movies and their directors. Table 3 shows the lists L_1, L_2 and L_3 containing the top-3 closest relaxations for each triple pattern along with their scores. Table 4 shows the top-5 relaxed queries and their scores.

Table 3. Top-3 relaxation lists for the triple patterns for an example query

Q: ?x directed ?m; ?m won Academy_Award; ?m hasGenre Action		
L_1	L_2	L_3
?x directed ?m : 0.0	?m won Academy_Award : 0.0	?m hasGenre Action : 0.0
?x actedIn ?m : 0.643	?m won Golden_Globe : 0.624	?m hasGenre Adventure : 0.602
?x created ?m : 0.647	?m won BAFTA_Award : 0.659	?m hasGenre Thriller : 0.612
?x produced ?m : 0.662	?m ?r Academy_Award : 0.778	?m hasGenre Crime : 0.653

Table 4. Top-5 relaxed queries for an example query. The relaxed entities/relations are underlined.

Q: ?x directed ?m; ?m won Academy_Award; ?m hasGenre Action	
Relaxed Queries	**score**
?x directed ?m;?m won Academy_Award;?m hasGenre <u>Adventure</u>	0.602
?x directed ?m;?m won Academy_Award;?m hasGenre <u>Thriller</u>	0.612
?x directed ?m;?m won <u>Golden_Globe</u>;?m hasGenre Action	0.624
?x <u>actedIn</u> ?m;?m won Academy_Award;?m hasGenre Action	0.643
?x <u>created</u> ?m;?m won Academy_Award;?m hasGenre Action	0.647

Now, we describe how results of the original and relaxed queries can be merged and ranked before representing them to the user.

3.2 Result Ranking

Let Q_0 be the original query and let the set of its relaxations be $R = \{Q_1, Q_2, ..., Q_r\}$. Moreover, let the results of the query Q_0 and all its relaxations be the set $\{T_1, T_2, ..., T_n\}$ where T_i is a result matching one (or more) of the queries Q_j. Note that a result T_i can be a match to more than one query. For example, consider the query Q_0 = ?m hasGenre Action and a relaxation Q_j = ?m hasGenre ?x. The result Batman_Begins hasGenre Action matches both Q_0 and Q_j.

To be able to rank the results of the original and relaxed queries, we assume that a result T matching query Q_j is scored using some score function f. The scoring function can be any ranking function for structured triple-pattern queries over RDF-data. In this paper, we make use of the language-model based ranking function described in [8]. Let the score of each result T with respect to query Q_j be $f(T, Q_j)$ and let the score of each relaxed query Q_j be s_j where the score of a relaxed query is computed as described in the previous subsection. In order to merge the results of the original and relaxed queires into a unified result set, we utilize the following scoring function for computing the score of a result T:

$$S(T) = \Sigma_{j=0}^{r} \lambda_j f(T, Q_j)$$

We next describe two techniques to set the values of the different λ's.

Adaptive Weighting. In this weighting scheme, we assume that the user is interested in seeing the results in a holistic manner. That is, a match to a lower ranked (relaxed) query can appear before a match to a higher ranked query. For example, consider the query Q_0 = ?m hasGenre Action and a relaxation Q_j = ?m hasGenre Thriller. The assumption now is that the user would rather see a "famous" movie of genre thriller, rather than an obscure movie of genre action. And so, a "mixing" of results is allowed. To this end, we set the λ_j's as a function of the scores of the relaxed queries s_j's as follows:

$$\lambda_j = \frac{1 - s_j}{\Sigma_{i=0}^{r}(1 - s_i)}$$

Recall that the smaller the s_j is, the closer Q_j is to the original query Q_0. Also recall that s_0 is equal to 0. This weighting scheme basically gives higher weights to the matches to relaxed queries which are closer to the original query. However, matches for a lower ranked query with sufficiently high scores can be ranked above matches for a higher ranked query.

Incremental Weighting. In this weighting scheme, we assume that the user is interested in seeing results *in order*. That is, all ranked matches of the original query first, followed by all ranked matches of the first relaxed query, then those of the second relaxed query, etc. That is, the results are presented "block-wise".

In order to do this, we need to set the λ_j's by examining the scores of the highest scoring and lowest scoring result to a given query. For example, consider our example query : ?m hasGenre Action. Suppose a relaxation to this query is ?x hasGenre Thriller. If we want to ensure that all matches of the original query are displayed before the first match of the relaxed query, we first examine the result with the lowest score for the original query and the highest score for the relaxed query. Let these results be T_0^{low} and T_1^{high}, respectively. We now need to ensure that $\lambda_0 * f(T_0^{low}, Q_0) > \lambda_1 * f(T_1^{high}, Q_1)$.

Note that both incremental as well as adaptive weighting are only two ways in which we can present results to the user. Additional schemes can include a mixture of both schemes for instance, or any other variations. Our ranking model is general enough and can support any number of such fine-grained result presentation schemes.

4 Experimental Evaluation

We evaluated the effectiveness of our relaxation techniques in 2 experiments. The first one evaluated the quality of individual entity and relation relaxations. It also evaluated the quality of the relaxed queries overall. The second experiment evaluated the quality of the final query results obtained from both original and relaxed queries. The complete set of evaluation queries used, relevance assessments collected and an online demo can be found at `http://www.mpii.de/~elbass/demo/demo.html`.

4.1 Setup

All experiments were conducted over two datasets using the Amazon Mechanical Turk service[2]. The first dataset was derived from the LibaryThing community, which is an online catalog and forum about books. The second dataset was derived from a subset of the Internet Movie Database (IMDB). The data from both sources was automatically parsed and converted into RDF triples. Overall, the number of unique entities was over *48,000* for LibraryThing and *59,000* for IMDB. The number of triples was over *700,000* and *600,000* for LibraryThing and IMDB, respectively.

Due to the lack of an RDF query benchmark, we constructed *40* evaluation queries for each dataset and converted them into structured triple-pattern queries. The number of triple patterns in the constructed queries ranged from 1 to 4. Some example queries include: “People who both acted as well as directed an Academy Award winning movie” (3 triple patterns), “Children’s book writers who have won the Booker prize” (3 triple patterns), etc.

4.2 Quality of Relaxations

To evaluate the quality of individual entity and relation relaxations, we extracted all unique entities and relations occurring in all evaluation queries. The total numbers of entities and relations are given in Table 5. For each entity, the top-5 relaxations were retrieved, excluding the variable relaxation. We presented the entity and each relaxation to 6 evaluators and asked them to assess how closely related the two are on a 3-point scale: 2 corresponding to ”closely related”, 1 corresponding to ”related” and 0 corresponding to ”unrelated”. The same was done for each relation.

To evaluate the quality of relaxed queries overall, we generated the top-5 relaxed queries for each evaluation query. The relaxed queries were ranked in ascending order of their scores, which were computed as described in Sect. 3.1. We asked 6 evaluators to assess how close a relaxed query is to the original one on a 4-point scale: 3 corresponding to ”very-close”, 2 to ”close”, 1 to ”not so close” and 0 corresponding to ”unrelated”.

Table 5 shows the results obtained for entity, relation and query relaxations. For a given entity, the average rating for each relaxation was first computed and then this rating was averaged over the top-5 relaxations for that entity. A similar computation was performed for relations and query relaxations. The second row shows the average rating over all relaxed entities, relations and queries. The third row shows the *Pearson*

[2] `http://aws.amazon.com/mturk/`

correlation between the average rating and the JS-divergence. We achieved a strong *negative* correlation for all relaxations which shows that the smaller the score of the relaxation (closer the relaxation is to the original), the higher the rating assigned by the evaluators. The fourth row shows the average rating for the top relaxation.

The fifth and sixth rows in Table 5 show the average rating for relaxations that ranked above and below the variable relaxation respectively. Recall that, for each entity or relation, a possible entry in the relaxation candidate-list is a variable as described in Section 2. For those relaxations that ranked above a variable (i.e., whose JS-divergence is less than that of a variable), the average rating was more than 1.29 for both entities and relations, indicating how close these relaxations are to the original entity or relation. For those relaxations that ranked below a variable, the average rating was less than 1.1 for entities and 0.8 for relations. This shows that the evaluators, in effect, agreed with the ranking of the variable relaxation.

Table 5. Results for entity, relation and query relaxations

Row	**Metric**	**Entities** (0-2)	**Relations** (0-2)	**Queries** (0-3)
1	**No. of items**	87	15	80
2	**Avg. rating**	1.228	0.863	1.89
3	**Correlation**	-0.251	-0.431	-0.119
4	**Avg. rating for top relaxation**	1.323	1.058	1.94
5	**Avg. rating above variable**	*1.295*	*1.292*	-
6	**Avg. rating below variable**	1.007	0.781	-

Table 6. Average NDCG and rating for all evaluation queries for both datasets

Librarything			
	Adaptive	**Incremental**	**Baseline**
NDCG	**0.868**	**0.920**	0.799
Avg. Rating	**2.062**	**2.192**	1.827
IMDB			
	Adaptive	**Incremental**	**Baseline**
NDCG	**0.880**	**0.900**	0.838
Avg. Rating	**1.874**	**1.928**	1.792

4.3 Quality of Query Results

We compared our relaxation framework, with its two weighting scheme variants, *Adaptive* and *Incremental* (see Sect. 3.2), against a baseline approach outlined in [8]. The latter simply replaces a constant in the original query with a variable to generate a relaxed query. The weight of the relaxed triple pattern is determined based on the number of constants replaced. For all 3 methods, we set the weight of an original triple pattern to the same value, to ensure that exact matches would rank on top, and thus make the differences rely solely on the quality and weights of the relaxations. We used the same ranking function f to rank the results with respect to a query Q_j for all 3 techniques. The ranking function was based on the LM-based ranking model in [8].

We pooled the top-10 results from all 3 approaches and presented them to 6 evaluators in no particular order. The evaluators were required to assess the results on a 4 point scale: 3 corresponding to ”highly relevant”, 2 corresponding to ”relevant”, 1 corresponding to ”somewhat relevant”, and 0 corresponding to ”irrelevant”. To measure the ranking quality of each technique, we used the Normalized Discounted Cumulative Gain (DCG) [12], a standard IR measure that takes into consideration the rank of relevant results and allows the incorporation of different relevance levels.

The results of our user evaluation are shown in Table 6. The reported NDCG values were averaged over all evaluation queries. Both variations of our framework (the first two columns) *significantly* outperformed the baseline approach, with a one-tailed paired t-test (p-value ≤ 0.01). The Adaptive approach had over 8% improvement in NDCG over the baseline for Librarything and over 5% for IMDB. The Incremental approach had improvements over 15% for Librarything and 7% for IMDB. Furthermore, we computed the average rating over all results for each technique as shown in the second and fourth rows (Avg. Rating) in Table 6.

Finally, in Table 7 we show an example evaluation query and the top-3 results returned by each relaxation approach. Next to each result, we show the average rating given by the evaluators. The relaxed constants are underlined.

The example query in Table 7 asks for science fiction books that have tag Film. There is only *one* one such result which is ranked as the top result by all 3 approaches. Since the Adaptive approach ranks the whole set of approximate results, it allows for more diversity in terms of relaxations. And so, the Adaptive approach returns the more famous and iconic movies, Blade and Star_Wars as the top results compared to The_Last_Unicorn and The_Mists_Of_Avalon returned by the Incremental scheme.

Table 7. Top-ranked results for the example query ”A science-fiction book that has tag Film”

Result	**Rating**
Q: ?b type Science_Fiction**;** ?b hasTag Film	
Adaptive	
Star_Trek_Insurrection type Science_Fiction; Star_Trek_Insurrection hasTag Film	2.50
Blade type Science_Fiction; Blade hasTag Movies	2.83
Star_Wars type Science_Fiction; Star_Wars hasTag Made_Into_Movie	2.00
Incremental	
Star_Trek_Insurrection type Science_Fiction; Star_Trek_Insurrection hasTag Film	2.50
The_Last_Unicorn type Science_Fiction; The_Last_Unicorn hasTag Movie/tv	2.50
The_Mists_of_Avalon type Science_Fiction; The_Mists_of_Avalon hasTag Movie/tv	2.17
Baseline	
Star_Trek_Insurrection type Science_Fiction; Star_Trek_Insurrection hasTag Film	2.50
Helter_Skelter type History; Helter_Skelter hasTag Film	0.83
Fear_&_Loathing_in_Vegas type History; Fear_&_Loathing_in_Vegas hasTag Film	1.83

5 Related Work

One of the problems addressed in this paper is that of relaxing entities and relations with similar ones. This is somewhat related to both record linkage [14], and ontology matching [19]. But a key difference is that we are merely trying to find candidates which are *close in spirit* to an entity or relation, and not trying to solve the entity disambiguation problem. Other kinds of reformulations such as spelling correction, etc. directly benefit from techniques for record linkage, but are beyond the scope of our work.

Query reformulation in general has been studied in other contexts such as keyword queries [5] (more generally called query expansion), XML [1, 13], SQL [4, 22] as well as RDF [11, 7, 10]. Our setting of RDF and triple patterns is different in being schema-less (as opposed to relational data) and graph-structured (as opposed to XML which is mainly tree-structured and supports navigational predicates).

For RDF triple-pattern queries, relaxation has been addressed to some extent in [22, 11, 7, 10, 8]. This prior work can be classified based on several criteria as described below. Note that except for [22] and our previous work in [8], none of the other papers report on experimental studies.

Scope of Relaxations. With the exception of [7, 11], the types of relaxations considered in previous papers are limited. For example, [22] considers relaxations of relations only, while [10, 8] consider both entity and relation relaxations. The work in [8], in particular, considers a very limited form of relaxation – replacing entities or relations specified in the triple patterns with variables. Our approach, on the other hand, considers a comprehensive set of relaxations and in contrast to most other previous approaches, weights the relaxed query in terms of the *quality* of the relaxation, rather than the number of relaxations that the query contains.

Relaxation Framework. While each of the proposals mentioned generates multiple relaxed query candidates, the method in which they do so differ. While [7, 11, 10] make use of rule-based rewriting, the work in [22] and our own work make use of the data itself to determine appropriate relaxation candidates. Note that rule-based rewriting requires human input, while our approach is completely automatic.

Result Ranking. Our approach towards result ranking is the only one that takes a holistic view of both the original and relaxed query results. This allows us to rank results based on both the *relevance* of the result itself, as well as the *closeness* of the relaxed query to the original query. The "block-wise" ranking adopted by previous work – that is, results for the original query are listed first, followed by results of the first relaxation and so on – is only one strategy for ranking, among others, that can be supported by our ranking model.

6 Conclusion

We proposed a comprehensive and extensible framework for query relaxation for entity-relationship search. Our framework makes use of language models as its foundation and can incorporate a variety of information sources on entities and relations. We showed how to use an RDF knowledge base to generate high quality relaxations. Furthermore, we showed how different weighting schemes can be used to rank results. Finally, we showed the effectiveness of our techniques through a comprehensive user evaluation. We believe that our contributions are of great importance for an extended-SPARQL API that could underlie the emerging "Web-of-Data" applications such as Linking-Open-Data across heterogeneous RDF sources.

References

[1] Amer-Yahia, S., Lakshmanan, L.V.S., Pandit, S.: Flexpath: Flexible structure and full-text querying for xml. In: SIGMOD (2004)

[2] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)

[3] Billerbeck, B., Zobel, J.: When query expansion fails. In: SIGIR (2003)

[4] Chaudhuri, S., Das, G., Hristidis, V., Weikum, G.: Probabilistic information retrieval approach for ranking of database query results. ACM Trans. on Database Syst. 31(3) (2006)

[5] Croft, B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice. Pearson Education, London (2009)

[6] Doan, A., Gravano, L., Ramakrishnan, R., Vaithyanathan, S. (eds.): Special issue on managing information extraction. ACM SIGMOD Record 37(4) (2008)

[7] Dolog, P., Stuckenschmidt, H., Wache, H., Diederich, J.: Relaxing rdf queries based on user and domain preferences. Journal of Intell. Inf. Sys. (2008)

[8] Elbassuoni, S., Ramanath, M., Schenkel, R., Sydow, M., Weikum, G.: Language-model-based ranking for queries on RDF-graphs. In: CIKM (2009)

[9] Fang, H., Zhai, C.: Probabilistic models for expert finding. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 418–430. Springer, Heidelberg (2007)

[10] Huang, H., Liu, C., Zhou, X.: Computing relaxed answers on RDF databases. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 163–175. Springer, Heidelberg (2008)

[11] Hurtado, C., Poulovassilis, A., Wood, P.: Query relaxation in rdf. Journal on Data Semantics (2008)

[12] Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: SIGIR (2000)

[13] Lee, D.: Query Relaxation for XML Model. Ph.D. thesis, UCLA (2002)

[14] Naumann, F., Herschel, M.: An Introduction to Duplicate Detection. Morgan & Claypool, San Francisco (2010)

[15] Nie, Z., Ma, Y., Shi, S., Wen, J.-R., Ma, W.Y.: Web object retrieval. In: WWW (2007)

[16] Petkova, D., Croft, W.: Hierarchical language models for expert finding in enterprise corpora. Int. J. on AI Tools 17(1) (2008)

[17] Sarawagi, S.: Information extraction. Foundations and Trends in Databases 2(1) (2008)

[18] Serdyukov, P., Hiemstra, D.: Modeling documents as mixtures of persons for expert finding. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 309–320. Springer, Heidelberg (2008)

[19] Staab, S., Studer, R.: Handbook on Ontologies (International Handbooks on Information Systems). Springer, Heidelberg (2004)

[20] Suchanek, F., Sozio, M., Weikum, G.: SOFIE: A self-organizing framework for information extraction. In: WWW (2009)

[21] Vallet, D., Zaragoza, H.: Inferring the most important types of a query: a semantic approach. In: SIGIR (2008)

[22] Zhou, X., Gaugaz, J., Balke, W.T., Nejdl, W.: Query relaxation using malleable schemas. In: SIGMOD (2007)

[23] Zhu, J., Nie, Z., Liu, X., Zhang, B., Wen, J.R.: Statsnowball: a statistical approach to extracting entity relationships. In: WWW (2009)

Optimizing Query Shortcuts in RDF Databases

Vicky Dritsou[1,3], Panos Constantopoulos[1,3],
Antonios Deligiannakis[2,3], and Yannis Kotidis[1,3]

[1] Athens University of Economics and Business, Athens, Greece
[2] Technical University of Crete, Crete, Greece
[3] Digital Curation Unit, IMIS, Athena Research Centre, Greece
{vdritsou,panosc,kotidis}@aueb.gr, adeli@softnet.tuc.gr

Abstract. The emergence of the Semantic Web has led to the creation of large semantic knowledge bases, often in the form of RDF databases. Improving the performance of RDF databases necessitates the development of specialized data management techniques, such as the use of shortcuts in the place of path queries. In this paper we deal with the problem of selecting the most beneficial shortcuts that reduce the execution cost of path queries in RDF databases given a space constraint. We first demonstrate that this problem is an instance of the quadratic knapsack problem. Given the computational complexity of solving such problems, we then develop an alternative formulation based on a bi-criterion linear relaxation, which essentially seeks to minimize a weighted sum of the query cost and of the required space consumption. As we demonstrate in this paper, this relaxation leads to very efficient classes of linear programming solutions. We utilize this bi-criterion linear relaxation in an algorithm that selects a subset of shortcuts to materialize. This shortcut selection algorithm is extensively evaluated and compared with a greedy algorithm that we developed in prior work. The reported experiments show that the linear relaxation algorithm manages to significantly reduce the query execution times, while also outperforming the greedy solution.

1 Introduction

The Semantic Web involves, among other things, the development of semantic repositories in which structured data is expressed in RDF(S) or OWL. The structure of this data - and of its underlying ontologies - is commonly seen as a directed graph, with nodes representing concepts and edges representing relationships between concepts. A basic issue that semantic repositories need to address is the formulation of ontology-based queries, often by repeatedly traversing particular paths [11] of large data graphs. Regardless of the specific model used to store these data graphs (RDF/OWL files, relational databases, etc.), path expressions require substantial processing; for instance, when using relational databases, multiple join expressions are often involved [3,20,22,28].

By analogy to materialized views in relational databases, a *shortcut* construct can be used in RDF repositories to achieve better performance in formulating and executing frequent path queries. In this paper we elaborate on the creation of *shortcuts* that correspond to frequently accessed paths by augmenting the schema and the data graph of an

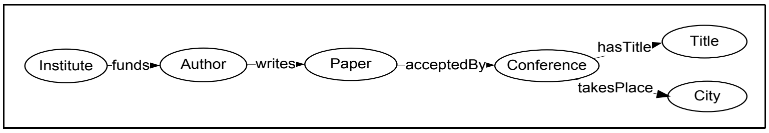


Fig. 1. Sample Schema Graph

RDF repository with additional triples, effectively substituting the execution of the corresponding path queries. Considering all possible shortcuts in a large RDF repository gives rise to an optimization problem in which we seek to select shortcuts that maximize the reduction of query processing cost subject to a given space allocation for storing the shortcuts. For this problem, which turns out to be a knapsack problem, known to be NP-hard, we have previously developed a greedy algorithm [13] which, however, leans on the side of wasting space whenever it comes to dealing with particular types of query workloads, especially correlated ones. We thus consider an alternative formulation in this paper, which leads to the development of a more efficient linear algorithm.

The contributions of this work are: (i) we elaborate on the notion of shortcuts and the decomposition of a set of user queries that describe popular user requests into a set of simple path expressions in the RDF schema graph of an RDF repository, which are then combined in order to form a set of candidate shortcuts that can help during the evaluation of the original queries; (ii) we formally define the shortcut selection problem as one of maximizing the expected benefit of reducing query processing cost by materializing shortcuts in the knowledge base, under a given space constraint; (iii) we provide an alternative formulation of the shortcut selection problem that trades off the benefit of a shortcut with the space required for storing its instances in the RDF database: the constraint matrix of the resulting bi-criterion optimization problem enjoys the property of total unimodularity, by virtue of which we obtain very efficient classes of linear programming solutions; and (iv) through extensive experimental evaluation we demonstrate that the linear algorithm outperforms our greedy solution in most cases.

2 Problem Formulation

In this section we provide the underlying concepts that are required in order to formulate our problem. Sections 2.1 and 2.2 describe these preliminary concepts, which have been introduced in [13].

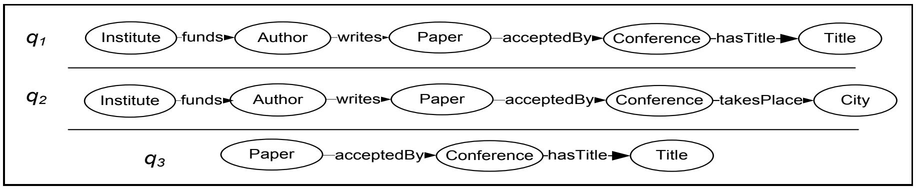


Fig. 2. Sample Queries

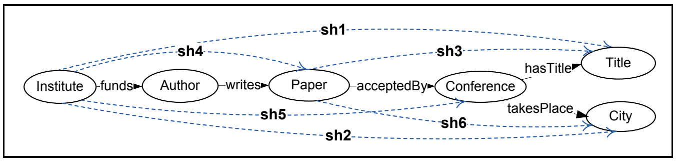


Fig. 3. Candidate Shortcuts

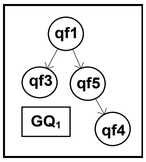


Fig. 4. Graph GQ_1

2.1 Preliminary Concepts

Assume an RDF database G containing two parts: (i) a schema graph, $G_S(V_S, E_S)$, where V_S is a set of nodes that represent entity classes (or, concepts) and a set of edges E_S, representing relationship classes (or, properties); and (ii) a data graph, $G_D(V_D, E_D)$, consisting of a set of nodes, V_D, that are instances of nodes in V_S and a set of edges, E_D, that are instances of edges in E_S. More generally, G could be any graph-structured semantic database. A sample schema graph of an RDF database is presented in Figure 1. Assume now a query workload $Q = \{q_1, q_2, q_3, \ldots, q_m\}$ of m queries. Each q_i is associated with a frequency f_i and comprises a data-level query (querying the schema graph is out of scope in this paper). Moreover, it is represented as a weakly connected subgraph G_q of the schema graph G_S. For example, consider the schema graph of Figure 1 and the three related queries: q_1: "For each institute, find the titles of the conferences in which the papers of the authors it funds have been accepted"; q_2: "For each institute, find the cities where the conferences in which the papers of the authors it funds have been accepted are taking place"; and q_3: "For each paper, find the title of the conference in which it has been accepted". The subgraphs of these queries are shown in Figure 2. Regardless of the query language used, to evaluate the queries and retrieve their matching instances we need to traverse the corresponding paths.

In this paper we deal only with path queries, like the three examples presented above. More complex queries can also be formulated, requiring more than one paths in order to be expressed, e.g. in our sample graph a query that relates $Institute$s with conference $Title$s together with the $City$ each one is located. Although this query cannot be part of our assumed workload, we can retrieve its matching instances by joining the results of the path queries it contains (namely q_1 and q_2).

Query fragments and candidate shortcuts. Each path query q_i in Q has a *length* l_i expressing the number of relationships its path contains. We call *query fragment* any subpath containing more than one edge. In our framework, a shortcut matches exactly one such query fragment. Given a query workload Q, containing $|Q|$ queries, and L the length of the query having the longest path among all queries in Q, there are $O(|Q| \times L^2)$ query fragments (and shortcuts). This large number is reduced when we define as *candidate shortcuts* only those that originate from or terminate to nodes that we consider to be "interesting": a node $u \in G_S$ is considered to be a *candidate shortcut node* iff (i) u is a starting node or an ending node of a query $q \in Q$; or (ii) u is a starting node or an ending node of two different edges e_i, e_j of G_S, each being traversed by at least one query $q \in Q$. Having defined the set of candidate nodes, we develop the set SH of *candidate shortcuts* by considering all valid combinations between candidate shortcut nodes, so that these are connected through an existing path of G_S and this path is traversed by at least one query $q \in Q$. For example, consider again the sample graph of Figure 1 and the queries of Figure 2 constituting the workload Q. Given Q, the candidate shortcut nodes are: (i) *Institute*, as starting node of q_1 and q_2; (ii) *Paper*, as starting node of q_3; (iii) *Conference*, since two edges belonging to two different queries originate from it; (iv) *Title*, as ending node of q_1 and q_3; and (v) *City*, as ending node of q_2. Starting from these, we generate the set of candidate shortcuts presented in Figure 3[1].

Each candidate shortcut sh_i maps to exactly one query fragment qf_i, which may be contained in more than one queries of Q. The set of queries in which qf_i is contained is called *related queries* of sh_i and is denoted as RQ_i. For instance, the set of related queries of sh_4 in Figure 3 is $RQ_4 = \{q_1, q_2\}$. Regarding the relationships between shortcuts, we distinguish the following three cases: (i) a shortcut sh_i is *disjoint* to sh_j if the query fragments they map to do not share any common edge; (ii) a shortcut sh_i is *fully contained* in sh_j iff the query fragment that sh_i maps to is a subpath of the corresponding query fragment of sh_j, denoted hereafter as $sh_i \prec sh_j$; and (iii) a shortcut sh_i *overlaps* sh_j if they are not disjoint, and none of them fully contains the other. Finally, for each shortcut sh_i we denote the set $SF_i = \{qf_j \mid qf_j \prec qf_i\}$.

2.2 Estimating Shortcut Benefit

We now turn to defining the benefit of introducing a shortcut and formulating shortcut selection as a benefit maximization problem. Assume a candidate shortcut sh_i with underlying query fragment qf_i and its set of related queries RQ_i. An estimate of the cost of retrieving the answer to qf_i may be given by (i) the number of edges that need to be traversed in the data graph G_D; (ii) the actual query time of qf_i; (iii) an estimate by a query optimizer. For the formulation of our problem we assume that this cost is given by the number of edges, tr_i, that need to be traversed in the data graph G_D (however in Section 4 we present a comparison of the results obtained when considering as cost the number of traversals on one hand and the actual query times on the other). Note that not all edges traversed necessarily lead to an answer, however they do contribute to the counting of tr_i. Now suppose that we augment the database by shortcut sh_i. This involves inserting one edge in the schema graph G_S, while in the data graph G_D one edge is inserted for each result of qf_i, a total of r_i edges between the corresponding nodes. Hence, by inserting one instance of sh_i for each result of qf_i using RDF Bags[2], thus allowing duplicate instances of shortcuts (triples) to be inserted, we retain the result cardinalities and obtain exactly the same query results. Then the new cost of answering qf_i is equal to the new number of traversals required, i.e. the number r_i of results of qf_i. Therefore, the benefit obtained by introducing a shortcut sh_i in order to answer qf_i is equal to the difference between the initial cost minus the new cost, $tr_i - r_i$. Since qf_i is used in answering each of its related queries, the benefit for query $q_k \in RQ_i$ can be estimated by multiplying the fragment benefit by the frequency of the query, i.e. $f_k(tr_i - r_i)$. The total benefit obtained by introducing shortcut sh_i is then equal to the sum of the benefits obtained for each related query. If the query fragments

[1] Due to readability issues, the number contained in the label of shortcuts is not presented in the figure as subscript.

[2] http://www.w3.org/TR/rdf-schema/

underlying the candidate shortcuts are disjoint then the aggregate benefit is the sum of the benefits of all candidate shortcuts. If, however, there are containment relationships between fragments, things get more complicated. For example, take the above mentioned queries q_1 and q_3 described in Figure 2. According to the definition of candidate shortcuts, there are four candidates beneficial for these queries, namely sh_1, sh_3, sh_4 and sh_5. Assume now that shortcut sh_5 has been implemented first and that adding sh_1 is being considered. The benefit of adding sh_1 with regard to query q_1 will be smaller than what it would have been without sh_5 in place. Indeed, sh_1 is not going to eliminate the traversals initially required for the fragment $qf_1 = Institute \overset{funds}{\rightarrow} Author \overset{writes}{\rightarrow} Paper \overset{acceptedBy}{\rightarrow} Conference \overset{hasTitle}{\rightarrow} Title$ but, rather, the traversals of edges of type sh_5. We denote as d_{ij} the difference in the number of traversals required to answer qf_i due to the existence of a shortcut induced by qf_j. This difference is positive for all $qf_j \prec qf_i$, i.e. for each $qf_j \in SF_i$. Finally, shortcuts induced by overlapping query fragments do not interfere in the above sense: since the starting node of one of them is internal to the other, they cannot have common related queries.

Let us now turn to the space consumption of shortcuts. Regardless of how many queries are related to a query fragment qf_i, the space consumption that results from introducing the corresponding shortcut sh_i is r_i as above. The total space consumption of all shortcuts actually introduced should not exceed some given space budget b.

2.3 Space-Constrained Benefit Maximization

We are now ready to formulate the problem of selecting beneficial shortcuts as a benefit maximization problem. Assume an RDF graph G, a finite set of m path queries $Q = \{q_i \mid i = 1, \dots, m\}$, $m \in \mathbb{N}$, each with a frequency f_i, a set $QF = \{qf_i \mid i = 1, \dots, n\}$ of n query fragments, $n \geqslant m, n \in \mathbb{N}$ deriving from the set Q and a space budget $b > 0$. Our problem is then defined as:

$$max \sum_{i=1}^{n} \sum_{q_k \in RQ_i} \{f_k(tr_i - r_i)x_i - f_k \sum_{qf_j \in SF_i} d_{ij} x_i x_j\}$$

subject to

$$\sum_{i=1}^{n} r_i x_i \leqslant b$$

$$x_i = \begin{cases} 1 & \text{if shortcut } sh_i \text{ is established} \\ 0 & \text{otherwise} \end{cases}$$

$$SF_i = \{qf_j \mid qf_j \prec qf_i, qf_j \in QF\}$$

$$RQ_i = \{q_j \mid qf_i \prec q_j, q_j \in Q\}.$$

This is a 0-1 quadratic knapsack problem, known to be NP-hard [15]. Several systematic and heuristic methods have been proposed in the literature [27] to obtain approximate solutions to this problem. All computationally reasonable methods assume non-negative terms in the objective function. This is obviously not the case of our problem, since the reductions d_{ij} of the traversals are non-negative integers thus resulting in non-positive quadratic terms.

To address this problem, we have previously developed a greedy algorithm [13] that seeks to maximize the overall benefit by selecting a subset of candidate shortcuts to materialize given a space budget. The algorithm takes as input the schema and data graph, together with the query workload, finds the candidate shortcuts and then computes the benefit of each one, considering as cost the number of edge traversals. It then incrementally selects the candidate with the maximum per-unit of space benefit that fits into the remaining budget, until it reaches the defined space consumption. This algorithm succeeds in identifying beneficial shortcuts, yet it wastes space, especially when the queries of the workload are correlated. This led us to consider an alternative formulation and develop a new algorithm, as described in the following Section. A detailed comparison of the two approaches is presented in Section 4.

3 Bi-criterion Optimization

One established class of solution methods for the 0-1 quadratic knapsack problem are Lagrangean methods, including the traditional Lagrangean relaxation of the capacity constraint and a more sophisticated variant, Lagrangean decomposition [8,10,25,27]. As explained in Section 2, such techniques are inapplicable to our problem, due to the negative quadratic terms in the objective function. However, the techniques cited above, provided the inspiration for an alternative modelling approach that would avoid the combined presence of binary decision variables and a capacity constraint.

Instead of stating a space constraint, we employ a cost term for space consumption in the objective function. This means that the utility of each calculated solution will, on one hand, increase based on the benefit of the selected shortcuts but, on the other hand, it will decrease proportionally to the solution's space consumption. The factor that determines the exact penalty incurred in our objective function per space unit significantly influences the final space required by the solution of our algorithm. A small penalty per space unit used typically leads to solutions that select many shortcuts, thus consuming a lot of space. On the contrary, a large penalty per space unit used typically leads to selecting fewer shortcuts of high benefit for materialization. As we describe at the end of this section, a binary search process will help determine the right value of this parameter, so as to select a solution that respects a space budget. Unlike Lagrangean methods that penalize consumption in excess of the budget, in our formulation space consumption is penalized from the first byte. As explained below, this formulation yields an efficiently solvable linear program.

We will now describe how the constraints are derived for three important cases:

- Specifying that a candidate shortcut may be useful (or not) for evaluating specific queries that contain its corresponding query fragment.
- Specifying how to prevent containment relations for the *same* query. For the same query, whenever we consider two materialized shortcuts such that one fully contains the other, then only one of them can be used for the evaluation of the query.[3] Expressing such constraints is crucial for the quality of the obtained solution. The

[3] In this case, it is not true that the larger of the two shortcuts will always be used, as the presence of additional materialized shortcuts for the same query may result in utilizing the smaller one.

greedy algorithm may select a smaller shortcut for materialization, and then select a shortcut that fully contains the first one.

- Specifying that two overlapping shortcuts (but none of them is fully contained in the other) cannot be useful at the same time for the same query.

In the model of Section 2, the resource-oriented treatment of space employs the space budget as a parameter for controlling system behavior. In the alternative, price-oriented model presented in this section, a price coefficient on space consumption is the control parameter. A small price on space consumption has an analogous effect to setting a high space budget. By iteratively solving the price-oriented model over a range of prices we effectively determine pairs of space consumption and corresponding solutions (sets of shortcuts). So, both the resource-oriented and the price-oriented approach yield shortcut selection policies the outcomes of which will be compared in terms of query evaluation cost and space consumption in Section 4.

Another important feature of the alternative model is the explicit representation of the usage of shortcuts in evaluating queries. In the model of Section 2, decision variables x_i express whether shortcut sh_i is inserted or not. The actual usage of selected shortcuts regarding each query is not captured by these variables. In fact, the set of shortcuts that serve a given query in an optimal solution as determined by the greedy algorithm has *no* containment relationships among shortcuts and *no* overlapping shortcuts. Of course, containment and overlapping can occur between shortcuts serving different queries. For example, consider again the queries q_1, q_2 and q_3 mentioned above. The greedy algorithm will not propose to use both shortcuts sh_1 and sh_3, which have a containment relationship, to serve q_1. Even if sh_3 is selected in one step and sh_1 in a subsequent step, only sh_1 will finally be used for the execution of q_1. Similarly, sh_5 and sh_3 will not be proposed by greedy for q_1, since their query fragments overlap. If we use sh_5 then the part of q_1 that corresponds to the subgraph $Paper \overset{acceptedBy}{\rightarrow} Conference$ will be "hidden" under the new edge sh_5 and therefore shortcut sh_3 cannot be applied.

Specifying that a shortcut is useful for a specific query. We introduce additional decision variables that express the usage of a given shortcut to serve a given query. In particular, x_{ik} denotes whether shortcut sh_i is useful for the query q_k, where $x_{ik} = \{0, 1\}$ and $q_k \in RQ_i$.

The x_{ik} variables depend on each other and on x_i in various ways. Note that if a shortcut sh_i is selected to serve any query q_k (specified by $x_{ik} = 1$), then this shortcut is also selected by our algorithm for materialization (specified by $x_i = 1$). Thus:

$$x_{ik} \leqslant x_i, \quad i = 1, ..., n; k \in RQ_i \tag{1}$$

Moreover, if a shortcut has been selected for materialization, then this shortcut is useful for at least one query. Thus:

$$x_i \leqslant \sum_{k \in RQ_i} x_{ik}, \quad i = 1, ..., n \tag{2}$$

Avoiding containment. For each query q_k we construct an auxiliary directed graph GQ_k as follows: The nodes of GQ_k represent the query fragments related to q_k. If (i) a query fragment qf_i is fully contained in another fragment qf_j, and (ii) there does not exist any qf_k ($k \neq i, j$) such that $qf_i \prec qf_k$ and $qf_k \prec qf_j$, then an edge is inserted in GQ_k starting from the bigger fragment qf_j and ending at the smaller fragment qf_i. Note that we connect each such node with its children only and not with all of its descendants. An example illustrating such a graph is shown in Figure 4, where the graph of query q_1 is presented. This graph is constructed with the help of Figure 3, and by recalling that each candidate shortcut sh_i matches the query fragment qf_i.

Using GQ_k we generate the containment constraints by the following procedure:

1. For each root node of the graph (i.e., each node with no incoming edges) of query q_k, find all possible paths p_t^k leading to leaf nodes of the graph (i.e., to nodes with no outgoing edges) and store them in a set of paths P^k.
2. For each path $p_t^k \in P^k$, create a containment constraint by restricting the sum of all variables x_{ik} whose fragment is present in the path to be less than or equal to 1.

The second step enforces that among all the candidate shortcuts participating in each path p_t^k, at most one can be selected. Continuing our previous example based on Figure 4, there exist two paths in P^1 for query q_1, namely $p_1^1 = \{qf_1, qf_3\}$ and $p_2^1 = \{qf_1, qf_5, qf_4\}$. These paths generate two containment constraints, namely $x_{11}+x_{31} \leqslant 1$ and $x_{11} + x_{51} + x_{41} \leqslant 1$.

Avoiding overlaps. Overlapping candidates are treated in a similar way. In graph GQ_k we compare each node with all other nodes on different paths of the graph. For each pair of compared nodes, if the corresponding query fragments of the nodes under comparison have edges in common, then we insert this pair of nodes in a set OF_k which stores the pairs of overlapping fragments with respect to query q_k. For each pair in OF_k we create a constraint specifying that the sum of its decision variables be less than or equal to 1, indicating that only one of the candidate shortcuts in the pair can be selected to serve the query q_k. In the previous example of query q_1, we would check the two pairs (qf_5, qf_3) and (qf_4, qf_3). Only the first of these pairs contains fragments that have edges in common, as qf_5 and qf_3 both contain the edge $Paper \overset{acceptedBy}{\rightarrow} Conference$. In order to specify that the shortcuts of these two query fragments cannot be useful for the same query q_1, we generate the constraint: $x_{51} + x_{31} \leqslant 1$.

In conclusion, given a parameter c (called *price coefficient*) that specifies the per space penalty of the chosen solution, we obtain the following bi-criterion 0-1 integer linear programming formulation for the shortcut selection problem:

$$max \sum_{i=1}^{n} \sum_{q_k \in RQ_i} f_k(tr_i - r_i)x_{ik} - c\sum_{i=1}^{n} r_i x_i \tag{3}$$

subject to

$$x_{ik} - x_i \leqslant 0, \;\; i = 1, ..., n; k \in RQ_i \tag{4}$$

$$-\sum_{k \in RQ_i} x_{ik} + x_i \leqslant 0, \;\; i = 1, ..., n \tag{5}$$

$$\sum_{i \in p_u^k} x_{ik} \leqslant 1, \;\; k = 1, ..., m; p_u^k \in P^k \tag{6}$$

$$x_{ik} + x_{jk} \leqslant 1, \quad k = 1, ..., m; (qf_i, qf_j) \in OF_k \tag{7}$$

$$x_{ik} = \begin{cases} 1 & \text{if shortcut } sh_i \text{ is selected for query } q_k \\ 0 & \text{otherwise} \end{cases}$$

$$x_i \in \{0, 1\}$$

3.1 Linear Relaxation

In the general case (i.e., unless particular conditions are satisfied), integer linear programs are NP-hard. In our problem though, a particular linear relaxation allows us to obtain fast solutions. Constraint (5) ensures that if *all* $x_{ik} = 0$, then also $x_i = 0$. If we remove this constraint, then it is possible to have $x_i = 1$ (i.e., to consume space) without using the shortcut in any query q_k. However, since this is a maximization problem and x_i has a negative coefficient in the objective function (3), a solution with $x_i = 1$ and all corresponding $x_{ik} = 0$ can be improved by setting $x_i = 0$. We can therefore drop the constraint from the problem, since it is never violated by optimal solutions.

For a reason that will shortly become clear, we also drop the overlap constraint(7). Contrary to constraint (5), this one can be violated by an optimal solution of the linear program. Then, a cutting plane method is used to restore feasibility with two alternative heuristics for cut selection, that will be presented in the sequel.

Consider now a relaxed linear program comprising the remaining constraints and with the 0-1 variables replaced by real variables in the interval $[0, 1]$:

$$max \sum_{i=1}^{n} \sum_{q_k \in RQ_i} f_k(tr_i - r_i)x_{ik} - c\sum_{i=1}^{n} r_i x_i \tag{8}$$

subject to

$$x_{ik} - x_i \leqslant 0, \quad i = 1, ..., n; k \in RQ_i \tag{9}$$

$$\sum_{i \in p_u^k} x_{ik} \leqslant 1, \quad k = 1, ..., m; p_u^k \in P^k \tag{10}$$

$$0 \leqslant x_{ik} \leqslant 1, \quad 0 \leqslant x_i \leqslant 1$$

The constraint matrix of the above linear program can be proven to fulfill the sufficient conditions of *total unimodularity* given in [31]. The proof is omitted due to space limitations. By virtue of this property, the linear program is guaranteed to have integer solutions (thus each x_{ik} and x_i will either be equal 0 or 1). Furthermore, the relaxation enables much more efficient solution since the empirical average performance of simplex-based linear program solvers is polynomial, in fact almost linear [4,7].

Satisfying the overlap constraints. The solution of the relaxed problem may violate constraint (9) that we have ignored in the relaxation. Here we propose a cutting plane method with two alternative heuristics for restoring constraint satisfaction. After solving the relaxed problem, we check whether the solution violates any of the omitted overlap constraints. Recall that these constraints contain two variables (representing two shortcuts used for the same query) that are not both allowed to be equal to 1. For each such violation we set the value of one of the variables equal to 0 (i.e., we exclude the respective shortcut) and insert this as an additional constraint into the initial problem. It can be proven that adding such constraints in the problem does not affect total unimodularity. The problem is solved again and new cuts are introduced as needed until we reach a solution that does not violate any overlap constraint.

But how do we choose which of the two variables to set equal to zero? We explore two alternative heuristics. The first heuristic (LRb) selects to prohibit the shortcut with the smallest benefit for the given query q_k. The second heuristic (LRl) chooses to prohibit the shortcut with the shortest corresponding query fragment. Both heuristics have been developed in evaluation tests and are discussed in Section 4.

Selecting the Value of the Price Coefficient c. In order to find a solution that is appropriate for a given space budget, we perform a binary search on the value of c. Both linear algorithms (i.e., heuristics) developed here take as input the space budget and then create linear models based on different values of c until they reach a solution the space consumption of which is close to (or, even better, equal to) the given budget.

4 Evaluation

In this section we present an experimental study of the two variations of the linear algorithm. We implemented LRb and LRl in C++ and compared their results with our previously developed greedy algorithm (GR) described in [13]. Our goal is to evaluate the reduction in query costs w.r.t. different RDF stores and data sets, to compare the performance of LRb/LRl with GR in general and more specifically to check whether LRb/LRl can capture the dependencies produced by strongly correlated workloads in a more effective way than GR. By strongly correlated we mean workloads containing path queries with containment relationships among the majority of them. In this spirit, we report on the reduction of the total execution time of the given query workloads after augmenting the database by the proposed shortcuts, vis-a-vis the allocated space budget. We used four different systems for our experiments, each of them following a different type of RDF store approach: (i) the SWKM[4] management system, which relies on a relational DBMS, (ii) the native Sesame[5] repository, (iii) the in-memory Sesame repository and (iv) the native RDF3X[6] system. In the following experiments, unless otherwise stated, we present in the graphs the percentage of reduction achieved in the query time of the entire workload after augmenting the system by the proposed shortcuts (y-axis) with regard to the space consumed for this augmentation, expressed as a fraction of the database size (x-axis). All reported experiments were executed on a Intel Core 2 Duo 2.33GHz PC with 4GB RAM running 32bit Windows Vista. In the reported experiments we used Lingo 9.0[7] as linear solver for LRb and LRl.

Before discussing the results obtained with strongly correlated queries, we examine the dependence of performance on the query cost metrics used. Recall from Section 2.2

[4] Available at http://139.91.183.30:9090/SWKM/
[5] Available at http://www.openrdf.org/
[6] Available at http://www.mpi-inf.mpg.de/ neumann/rdf3x/
[7] LINDO Systems, http://www.lindo.com

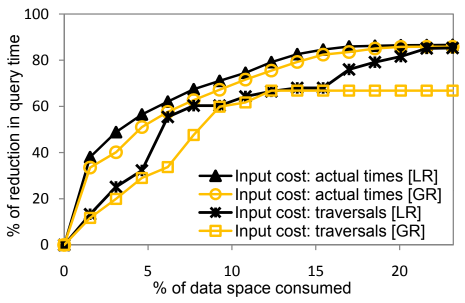


Fig. 5. Yago in Sesame Native

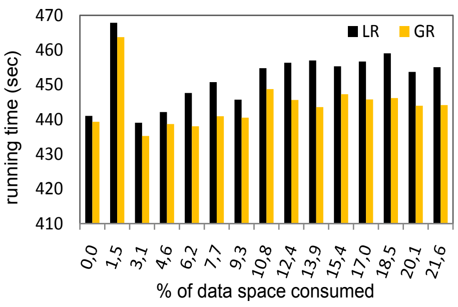


Fig. 6. Running Times of Algorithms

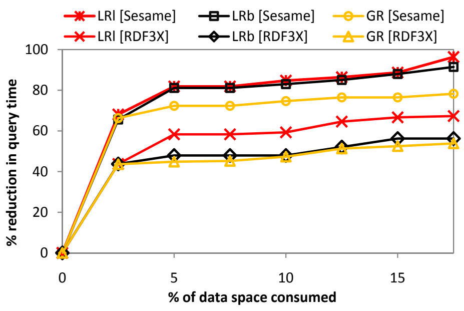


Fig. 7. Experiment with Correlated Queries

that computing the benefit of candidate shortcuts requires a cost metric of the query fragments. Our theoretical formulation employs the number of traversals required to answer the query, while the actual query times must also be considered. In [13] we have shown that both cost metrics give qualitatively similar results for the greedy algorithm. To confirm that this stands true for LRb/LRl too, we used the Yago data set [34] that contains 1.62 million real data triples. We considered as query workload the 233 distinct query paths of length 3 contained in the schema graph of Yago and used as RDF store the Sesame native repository. In Figure 5 we present the percentage of reduction in query time achieved for the entire workload (y-axis) after augmenting the system by the selected shortcuts w.r.t. the fraction of data space required to be consumed by the shortcut edges. In this experiment LRb and LRl give the same solutions, due to the small length of queries (presented as LR in Figure 5).

It is obvious that our approach significantly reduces the overall query execution time without consuming large amounts of space. Moreover, LR achieves a bigger reduction compared to greedy (GR): when using traversals as cost metric, LR reduces the query time by 55% by consuming only 6% of the data space, while GR reaches a reduction of 33% for the same consumption. LR reaches the maximum reduction of 86% of the initial query time with GR stopping at a maximum reduction of 66%. The reported results confirm that the two cost metrics used give qualitatively similar results. We

Table 1. Relative Improvement of Query Times in Sesame (over Greedy)

Space	$\frac{t_{GR}-t_{LRb}}{t_{GR}}$	$\frac{t_{GR}-t_{LRl}}{t_{GR}}$
2.5%	−2%	5%
5.0%	32%	34%
7.5%	32%	34%
10.0%	33%	40%
12.5%	36%	42%
15.0%	49%	52%
17.5%	61%	83%

will therefore continue in the following experiments by considering the traversals as input cost metric for our algorithms. Regarding the running times of the algorithms, we present in Figure 6 the time in seconds required by each algorithm to choose the appropriate shortcuts for the experiment described with the Yago data set. The graph shows the running time of the algorithms for each given budget. GR runs faster than LR in all cases, since LR must iterate through the binary search, with LR being from 2 to 13 seconds slower. On the other hand, running times are a secondary consideration, since shortcut selection is an offline procedure.

To evaluate the performance of the algorithms with strongly correlated queries, we generated a synthetic schema graph comprising 143 nodes and 117 edges and then used a Perl script to generate synthetic data. This script randomly picks a schema node at the beginning and generates one data node from it. It then produces all the data edges emanating from the latter according to the schema graph. This process is repeated on the newly created ending data nodes of the produced edges. A parameter p defining the probability of breaking a path (stop following the edges of the current node and continue with a new one) is also used, which in this experiment is set to 0.4. The data set generated for this experiment contains 1 million triples. For the correlated workload we used 31 path queries of length from 5 to 15 with 29 of them being fully contained in at least one other query of the workload; the smallest having length equal to 5 are fully contained in 10 longer queries. We used Sesame native repository and RDF3X for this experiment and the results obtained are presented in Figure 7. All algorithms show significant reduction in query time: by consuming just 2.5% of the data space they all reduce the query time by 65% in Sesame, while in RDF3X they all reach a reduction of 43% for the same space consumption. Moreover, LRb and LRl show much better results than GR in Sesame, while in RDF3X LRl outperforms the two remaining algorithms. We can thus confirm that the linear approach, and mostly LRl, captures the dependencies that exist in correlated workloads in a more effective way: LRl manages to reduce the query time by 96% when consuming only 17.5% of the data space in Sesame, while for the same consumption it achieves a reduction of 68% in RDF3X. Table 1 shows the relative improvement in query time reduction achieved in Sesame by the two variants of LR over GR for various space budgets. Both LRb and LRl show better results in most cases (LRb is by 2% worse than GR in one case of very small budget), while LRl yields 83% more efficient shortcuts than GR for the maximum budget.

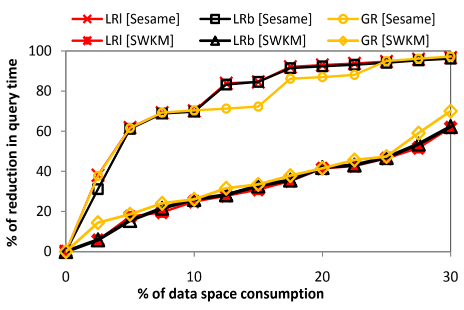


Fig. 8. Experiment with CIDOC Ontology

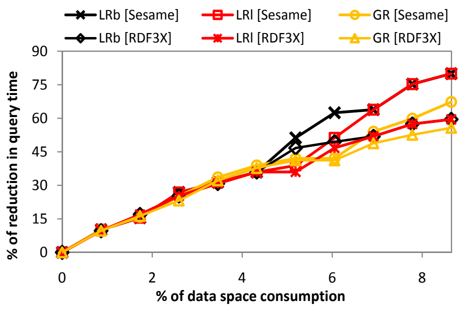


Fig. 9. Using Long Path Queries

In Figure 8 we present the results obtained by a different type of ontology, namely the CIDOC CRM[8], which is characterized by a schema graph forming a "constellation graph" structure with non-uniform density. This contains a small number of "star" nodes, each connected with a large number of "planet" nodes and sparsely connected with other star nodes. Our goal here is to evaluate the performance of LR algorithms with these schema graphs. We generated 1 million synthetic triples with the aforementioned Perl script and used as query workload 65 queries of length 2 to 4. We tested the algorithms using both the Sesame in-memory repository and the SWKM system. The reduction achieved is much bigger with Sesame than SWKM. Moreover, LRb and LRl perform better than GR when using Sesame, while this is not the case with SWKM: although in the majority of cases the three algorithms show similar results, in 3 cases GR achieves a bigger reduction by $5 - 7\%$.

To examine the performance of the algorithms with longer path queries, we generated a synthetic schema graph containing 189 nodes and 467 edges and populated it using the Perl script with 4 million of synthetic triples. The query workload here comprises 26 queries with length from 10 to 15. For this experiment we used the Sesame native repository and RDF3X, and obtained the results presented in Figure 9. The reduction achieved in both systems is similar in this case: all algorithms achieve a reduction of at least 40% by consuming only 5% of the data space. Moreover, LRb and LRl still give better results when compared to GR. E.g., for the maximum budget given in the experiment, the execution of the workload after augmenting the RDF3X system by the shortcuts proposed by LRl is by 39% faster than after inserting those proposed by GR.

5 Related Work

The emerging Semantic Web necessitates the development of methods to efficiently manage semantic data, often expressed in OWL and/or RDF. To this end, a variety of systems have been developed to handle RDF data including, among others, Jena[9], Sesame, RDF3X, SWKM. For the efficient querying of semantic data, several RDF

[8] Available at http://www.cidoc-crm.org/rdfs/cidoc_crm_v5.0.2_english_label.rdfs

[9] Available at http://jena.sourceforge.net/

languages have been proposed, with SPARQL [2], RQL [19] and RDQL [1] standing as main representatives.

In the relational world, views have long been explored in database management systems [30], either as pure programs [32,33], derived data that can be further queried [23], pure data (detached from the view query) or pure index [29]. The selection of shortcuts in RDF databases is similar to materialized views in relational systems, which has long been studied: selecting and materializing a proper set of views with respect to the query workload and the available system resources optimizes frequent computations [18,21]. Moreover, restricting the solution space into a set of views with containment properties has been proposed in the Data Cube [16] framework.

Indexing RDF data is similar in concept (due to the hierarchical nature of path queries) to indexing data in object oriented (OO) and XML Databases. An overview of indexing techniques for OO databases is presented in [5]. In [17] a uniform indexing scheme, termed U-index, for OO databases is presented, while in [6] the authors present techniques for selecting the optimal index configuration in OO databases when only a single path is considered and without taking into account overlaps of subpaths. The work of [12] targets building indices over XML data. Unlike XML data, RDF data is not rooted at a single node. Moreover, the above techniques target the creation of indices over paths, thus not tackling the problem of this paper, which is the selection of shortcuts to materialize given a workload of *overlapping* queries.

Following the experience from relational databases, indexing techniques for RDF databases have already been explored [14,24,35]. A technique for selecting which (out of all available) indices can help accelerate a given SPARQL query is proposed in [9]. In [3] the authors propose a vertically partitioned approach to storing RDF triples and consider materialized path expression joins for improving performance. While their approach is similar in spirit to the notion of shortcuts we introduce, the path expression joins they consider are hand-crafted (thus, there is no automated way to generate them in an arbitrary graph). Moreover the proposed solutions are tailored to relational backends (and more precisely column stores), while in our work we do not assume a particular storage model for the RDF database. Our technique can be applied in conjunction with these approaches to accelerate query processing, since it targets the problem of determining the proper set of shortcuts to materialize without assuming a particular RDF storage model. In [26] the authors propose a novel architecture for indexing and querying RDF data. While their system (RDF3X) efficiently handles large data sets using indices and optimizes the execution cost of small path queries, our evaluation shows that our technique further reduces the execution cost of long path queries in RDF3X.

6 Conclusions

Shortcuts are introduced as a device for facilitating the expression and accelerating the execution of frequent RDF path queries, much like materialized database views, but with distinctive traits and novel features. Shortcut selection is initially formulated as a benefit maximization problem under a space budget constraint. In order to overcome the shortcomings of a previous greedy algorithm solution to this problem, we developed an alternative bi-criterion optimization model and a linear solution method.

We have shown through an extensive experimental study that the augmentation of RDF databases with shortcuts reduces significantly the query time of the workload, while in the majority of cases the two variations of our algorithm outperform our previous greedy solution. This reduction was obtained using different types of RDF stores, uniform and non-uniform schema graphs, real as well as synthetic ontologies, different database sizes and different types of query workloads. In the case of strongly correlated query workloads, where the greedy solution tends to waste space, we have seen that our two variations (LRl and LRb) better capture the dependencies among queries and achieve a bigger reduction in query execution time.

References

1. RDQL - A Query language for RDF. W3C Member, http://www.w3.org/Submission/RDQL/
2. SPARQL Query Language for RDF. W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/
3. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB (2007)
4. Beasley, J.E.: Advances in Linear and Integer Programming. Oxford Science (1996)
5. Bertino, E.: A Survey of Indexing Techniques for Object-Oriented Database Management Systems. Query Processing for Advanced Database Systems (1994)
6. Bertino, E.: Index Configuration in Object-Oriented Databases. The VLDB Journal 3(3) (1994)
7. Borgwardt, K.H.: The average number of pivot steps required by the simplex-method is polynomial. Mathematical Methods of Operations Research 26(1), 157–177 (1982)
8. Caprara, A., Pisinger, D., Toth, P.: Exact Solution of the Quadratic Knapsack Problem. INFORMS J. on Computing 11(2), 125–137 (1999)
9. Castillo, R., Leser, U., Rothe, C.: RDFMatView: Indexing RDF Data for SPARQL Queries. Tech. rep., Humboldt University (2010)
10. Chaillou, P., Hansen, P., Mahieu, Y.: Best network flow bounds for the quadratic knapsack problem. Lecture Notes in Mathematics, vol. 1403, pp. 225–235 (2006)
11. Constantopoulos, P., Dritsou, V., Foustoucos, E.: Developing query patterns. In: Agosti, M., Borbinha, J., Kapidakis, S., Papatheodorou, C., Tsakonas, G. (eds.) ECDL 2009. LNCS, vol. 5714, pp. 119–124. Springer, Heidelberg (2009)
12. Cooper, B.F., Sample, N., Franklin, M.J., Hjaltason, G.R., Shadmon, M.: A Fast Index for Semistructured Data. In: VLDB (2001)
13. Dritsou, V., Constantopoulos, P., Deligiannakis, A., Kotidis, Y.: Shortcut selection in RDF databases. In: ICDE Workshops. IEEE Computer Society, Los Alamitos (2011)
14. Fletcher, G.H.L., Beck, P.W.: Indexing social semantic data. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, Springer, Heidelberg (2008)
15. Gallo, G., Hammer, P., Simeone, B.: Quadratic knapsack problems. Mathematical Programming 12, 132–149 (1980)
16. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In: ICDE (1996)
17. Gudes, E.: A Uniform Indexing Scheme for Object-Oriented Databases. Information Systems 22(4) (1997)
18. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing Data Cubes Efficiently. In: SIGMOD Conference (1996)

19. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: RQL: A declarative query language for RDF. In: WWW (2002)
20. Kotidis, Y.: Extending the Data Warehouse for Service Provisioning Data. Data Knowl. Eng. 59(3) (2006)
21. Kotidis, Y., Roussopoulos, N.: A Case for Dynamic View Management. ACM Trans. Database Syst. 26(4) (2001)
22. Larson, P., Deshpande, V.: A File Structure Supporting Traversal Recursion. In: SIGMOD Conference (1989)
23. Larson, P., Yang, H.Z.: Computing Queries from Derived Relations. In: VLDB (1985)
24. Liu, B., Hu, B.: Path Queries Based RDF Index. In: SKG, Washington, DC, USA (2005)
25. Michelon, P., Veilleux, L.: Lagrangean methods for the 0-1 Quadratic Knapsack Problem. European Journal of Operational Research 92(2), 326–341 (1996)
26. Neumann, T., Weikum, G.: The rdf-3x engine for scalable management of rdf data. VLDB J. 19(1) (2010)
27. Pisinger, D.: The quadratic knapsack problem - a survey. Discrete Applied Mathematics 155(5), 623–648 (2007)
28. Rosenthal, A., Heiler, S., Dayal, U., Manola, F.: Traversal Recursion: A Practical Approach to Supporting Recursive Applications. In: SIGMOD Conference (1986)
29. Roussopoulos, N., Chen, C.M., Kelley, S., Delis, A., Papakonstantinou, Y.: The ADMS Project: View R Us. IEEE Data Eng. Bull. 18(2) (1995)
30. Roussopoulos, N.: Materialized Views and Data Warehouses. SIGMOD Record 27 (1997)
31. Schrijver, A.: Theory of linear and integer programming. John Wiley, Chichester (1998)
32. Sellis, T.K.: Efficiently Supporting Procedures in Relational Database Systems. In: SIGMOD Conference (1987)
33. Stonebraker, M.: Implementation of Integrity Constraints and Views by Query Modification. In: SIGMOD Conference (1975)
34. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: WWW. ACM Press, New York (2007)
35. Udrea, O., Pugliese, A., Subrahmanian, V.S.: GRIN: A Graph Based RDF Index. In: AAAI (2007)

RDFS Update: From Theory to Practice

Claudio Gutierrez[1], Carlos Hurtado[2], and Alejandro Vaisman[3]

[1] Computer Science Department, Universidad de Chile
[2] Universidad Adolfo Ibañez, Chile
[3] Universidad de la República, Uruguay

Abstract. There is a comprehensive body of theory studying updates and schema evolution of knowledge bases, ontologies, and in particular of RDFS. In this paper we turn these ideas into practice by presenting a feasible and practical procedure for updating RDFS. Along the lines of ontology evolution, we treat schema and instance updates separately, showing that RDFS instance updates are not only feasible, but also deterministic. For RDFS schema update, known to be intractable in the general abstract case, we show that it becomes feasible in real world datasets. We present for both, instance and schema update, simple and feasible algorithms.

1 Introduction

RDF has become one of the prime languages for publishing data on the Web, thanks to initiatives like Linked Data, Open Data, Datagovs, etc. The next step is to work on the evolution of such data, thus, facing the issue of information update. If one analyzes the data that is being published, the vocabulary used includes the core fragment of RDFS plus some OWL features. This poses strong challenges to the goal of updating such information. It is well-known that the problem of updating and schema evolution in Knowledge Bases is both, intractable and non-deterministic in the general case. For example, erasing a statement φ (that is, updating the knowledge base so that the statement φ can not be deduced from it) not only could take exponential time, but, there could be many different and equally reasonable solutions. Thus, there is no global solution and the problem has to be attacked by parts.

In this paper we study the problem of updating data under the RDFS vocabulary, considering the rest of the vocabulary as constant. Many proposals on updates in RDFS and light knowledge bases (e.g. DL-lite ontologies) have been presented and we discuss them in detail in Section 5. Nevertheless, such proposals have addressed the problem from a strictly theoretical point of view, making them –due to the inherent complexity of the general problem– hard or impossible to be used in practice.

Using the Katsuno-Mendelzon theoretical approach (from now on, K-M approach) for update and erasure [8], which has been investigated and proved fruitful for RDFS (see [3,4,6]), we show that updates in RDFS can be made practical. We are able to get this result by (a) following the approach typical in ontology evolution, where schema and instance updates are treated separately; (b) focusing on the particular form of the deductive rules of RDFS; and (c) considering blank nodes as constants (which for current big data sets is a rather safe assumption) [1]. In this paper we concentrate in the erasure operation ('deleting' a statement), because update (adding information) in RDFS, due to the positive logic nature of it, turns out to be almost trivial [6]. Our two main results are, a deterministic and efficient algorithm for updating instances, and a reduction of the update problem for schema to a graph theoretical problem in very small graphs.

Regarding instance update, we show that due to the particular form of the rules involved in RDFS [13], and using a case by case analysis, instance erasure (i.e., erasing data without touching the schema) is a deterministic process for RDFS, that is, it can be uniquely defined, hence opening the door to automate it. Then, we show that this process can be done efficiently, and reduces essentially to compute reachability in small graphs. We present pseudo-code of the algorithms that implement this procedure.

As for schema erasure, the problem is intrinsically non-deterministic, and worst, intractable in general. A trivial example is a chain of subclases $(a_i, \texttt{sc}, a_{i+1})$ from where one would like to erase the triple $(a_1, \texttt{sc}, a_n)$. The minimal solutions consist in deleting one of the triples. In fact, we show that in general, each solution corresponds bi-univocally to the well-known problem of finding minimal cuts for certain graphs constructed from the original RDF graph to be updated. This problem is known to be intractable. The good news here is that the graphs where the cuts have to be performed are very small (for the data we have, it is almost of constant size: see Table 1). They correspond essentially to the subgraphs containing triples with predicates subClassOf and subPropertyOf. Even better, the cuts have to be performed over each connected component of these graphs (one can avoid cuts between different connected components), whose size is proportional to the length of subClassOf (respectively subPropertyOf) chains in the original graph. We also present pseudo-code for this procedure.

The remainder of the paper is organized as follows. Section 2 reviews RDF notions and notations and the basics of the K-M approach to erasure. Section 3 studies the theoretical basis of the erasure operations proposed, and Section 4 puts to practice the ideas presenting algorithms for efficiently computing erasure in practice. Section 5 discusses related work. We conclude in Section 6.

2 Preliminaries

To make this paper self-contained we present in this section a brief review of basic notions on RDF, and theory of the K-M approach to update in RDFS. Most of the material in this section can be found in [5,6,13] with more detail.

Definition 1 (RDF Graph). *Consider infinite sets U (URI references); $B = \{N_j : j \in \mathbb{N}\}$ (Blank nodes); and L (RDF literals). A triple $(v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an* RDF triple*. The union of U, B, L will be denoted by UBL.*

Table 1. Statistics of triples in schema, instances and $\texttt{sc}, \texttt{sp}$ chains of some RDF datasets. (The difference between # triples and #(schema + instances) is due the predicates sameAs, sameClass, which being schema, do not have semantics in RDFS.)

Dataset	# Triples	# Schema	#Instances	$\{\texttt{sc}, \texttt{sp}\}$-Chain-length	Most used voc.
bio2rdf (1)	2,024,177	685	1,963,738	3	type, label
data.gov.uk	22,504,895	16	22,503,962	1	type, value
bibsonomy	13,010,898	0	12,380,306	0	type, value
dbtune	58,920,361	418	58,248,647	7	type, label
geonames	9,415,253	0	9409247	0	type
uniprot	72,460,981	12295	72458497	4	type, reif.

An RDF graph *(just graph from now on) is a set of RDF triples. A* subgraph *is a subset of a graph. A graph is* ground *if it has no blank nodes.* □

A set of reserved words defined in RDF Schema (called the *rdfs-vocabulary*) can be used to describe properties like attributes of resources, and to represent relationships between resources. In this paper we restrict to a fragment of this vocabulary which represents the essential features of RDF and that contains the essential semantics (see [13]): [$\texttt{range}$], rdfs:domain [$\texttt{dom}$], rdf:type [$\texttt{type}$], rdfs: subClassOf [$\texttt{sc}$] and rdfs:subPropertyOf [$\texttt{sp}$]. The following set of rule schemas captures the semantics of this fragment [13]. In each rule schema, capital letters A, B, C, D, X, Y,... represent variables to be instantiated by elements of UBL.

GROUP A (Subproperty)

$$\frac{(A, \texttt{sp}, B) \ \ (B, \texttt{sp}, C)}{(A, \texttt{sp}, C)} \tag{1}$$

$$\frac{(A, \texttt{sp}, B) \ \ (X, A, Y)}{(X, B, Y)} \tag{2}$$

GROUP B (Subclass)

$$\frac{(A, \texttt{sc}, B) \ \ (B, \texttt{sc}, C)}{(A, \texttt{sc}, C)} \tag{3}$$

$$\frac{(A, \texttt{sc}, B) \ \ (X, \texttt{type}, A)}{(X, \texttt{type}, B)} \tag{4}$$

GROUP C (Typing)

$$\frac{(A, \texttt{dom}, C) \ \ (X, A, Y)}{(X, \texttt{type}, C)} \tag{5}$$

$$\frac{(A, \texttt{range}, D) \ \ (X, A, Y)}{(Y, \texttt{type}, D)} \tag{6}$$

Definition 2 (Proof Tree, Deduction). *Let G, H be RDF graphs, and t a triple. Then a* proof tree *of t from G is a tree constructed as follows: (1) The root is t; (2) The leaves are elements of G; (3) If t is a node, then t has children t_1, t_2 iff $\frac{t1 \ \ t_2}{t}$ is the instantiation of a rule (see rules above). If t has a proof tree from G we will write $G \vdash t$.*

A deduction of H from G is a set of proof trees from G, one for each $t \in H$. □

Definition 3 (Closure). *Let G be an RDF graph. The* closure *of G, denoted $cl(G)$, is the set of triples that can be deduced from G (under Definition 2), that is, $cl(G) = \{t : G \vdash t\}$.* □

The formalization of the K-M approach is based on the models of a theory. Thus we need the logical notion of a *model* of a formula (of an RDF graph). The model theory of RDF (given in [7]) follows standard classical treatment in logic with the notions of model, interpretation, and entailment, denoted $\models$ (see [5] for details). Also, throughout this paper we work with Herbrand models, which turn out to be special types of RDF graphs themselves. For a ground graph G, a Herbrand model of G is any RDF graph that contains $\mathrm{cl}(G)$ (in particular, $\mathrm{cl}(G)$ is a minimal model). $\mathrm{Mod}(G)$ will denote the set of such models of G. The deductive system presented is a faithful counterpart of these model-theoretic notions:

Proposition 1 (See [5,13]). *(1) $G \models H$ iff* $\mathrm{cl}(H) \subseteq \mathrm{cl}(G)$; *(2) The deductive system of Definition 2 is sound and complete for $\models$ (modulo reflexivity of* `sc` *and* `sp`*).*[1]

2.1 Semantics of Erase in RDF

From a model-theoretic point of view, the K-M approach can be characterized as follows: for each model M of the theory to be changed, find the set of models of the sentence to be inserted that are 'closest' to M. The set of all models obtained in this way is the result of the change operation. Choosing an update operator then reduces to choosing a notion of closeness of models.

Working with positive theories like RDFS, the problem of adding positive knowledge (e.g. a triple, a graph H) to a given graph G is fairly straightforward. In fact, for ground graphs it corresponds to the union of the graphs. (See [6]). Thus, in what follows we concentrate in the 'erase' operation, that is, 'deleting' a triple t (or a graph H) from a given graph G. A standard approach in KB is to ensure that, after deletion, the statement t should not be derivable from G, and that the deletion should be minimal. The result should be expressed by another formula, usually in a more expressive language. We next characterize the erase operation using the K-M approach, which essentially states that, erasing statements from G means adding models to $\mathrm{Mod}(G)$, the set of models of G.

Definition 4 (Erase Operator). *The operator $\bullet$, representing the erasure, is defined as follows: for graphs G and H, the semantics of $G \bullet H$ is given by:*

$$\mathrm{Mod}(G \bullet H) = \mathrm{Mod}(G) \ \cup \bigcup_{m \in \mathrm{Mod}(G)} \min(((\mathrm{Mod}(H))^c, \leq_m) \qquad (7)$$

[1] As in [13], we are avoiding triples of the form $(a, \mathtt{sc}, a)$ and $(b, \mathtt{sp}, b)$, because this causes no harm to the core of the deductive system (see [13]).

where $(\ )^c$ denotes complement. In words, the models of $(G \bullet H)$ are those of G plus the collection of models $m_H \not\models H$ such that there is a model $m \models G$ for which m_H is $\leq_m$-minimal among the elements of $\mathrm{Mod}(H)^c$. □

The following standard notion of distance between models gives an order which is the one we will use in this paper. Recall that the the symmetric difference between two sets S_1 and S_2, denoted as $S_1 \oplus S_2$, is $(S_1 \setminus S_2) \cup (S_2 \setminus S_1)$.

Definition 5 (Order $\leq_m$). *Let G, G_1, G_2 be models of RDF graphs, and let $\mathcal{G}$ be a set of models of RDF graphs. Then : (1) $G_1 \leq_G G_2$ (G_1 is 'closer' to G than G_2) if and only if $G_1 \oplus G \subseteq G_2 \oplus G$; (2) G_1 is $\leq_G$-minimal in $\mathcal{G}$ if $G_1 \in \mathcal{G}$, and for all $G_2 \in \mathcal{G}$, if $G_2 \leq_G G_1$ then $G_2 = G_1$.* □

Representing faithfully in RDF the notions of erase defined above is not possible in the general case, given its lack of negation and disjunction. The next example illustrates the problems.

Example 1. Let us consider the graphs $G = \{(a, \mathtt{sc}, b), (b, \mathtt{sc}, c)\}$, and $H = \{(a, \mathtt{sc}, c)\}$. Any graph G' representing the result of this update cannot contain both $(a, \mathtt{sc}, b)$, and $(b, \mathtt{sc}, c)$, since this would derive $(a, \mathtt{sc}, c)$. Then, the result of the update should be $\{(a, \mathtt{sc}, b)\} \vee \{(b, \mathtt{sc}, c)\}$. Elaborating on this a little further, in Equation 7, $\mathrm{Mod}(H)^c$ are the models that cannot derive $(a, \mathtt{sc}, c)$. From these models, $\min((\mathrm{Mod}(H))^c, \leq_m)$ contains the ones at distance '1' from $\mathrm{Mod}(G)$, namely $\{\{(a, \mathtt{sc}, b)\}, \{(b, \mathtt{sc}, c)\}\}$. Any model that does not include $(a, \mathtt{sc}, b)$ or $(b, \mathtt{sc}, c)$ is at distance ≥ 2 from $Mod(G)$. Moreover, any model including both triples would not be in $(\mathrm{Mod}(H))^c$ since it would derive $(b, \mathtt{sc}, c)$. □

2.2 Approximating Erase in RDF

Knowing that it is not possible in general to find RDF graphs representing the new state after erasure, we study the 'closest' RDF formulas that express it. In this direction, we introduce the notion of erase candidate, which gives a workable characterization of erase (expressed previously only in terms of sets of models).

Definition 6 (Erase Candidate). *Let G and H be RDF graphs. An erase candidate of $G \bullet H$ is a maximal subgraph G' of $\mathrm{cl}(G)$ such that $G' \not\models H$. We denote $\mathrm{ecand}(G, H)$ the set of erase candidates of $G \bullet H$.* □

Example 2. For the RDF graph G of Figure 1 (a), the set $\mathrm{ecand}(G, \{(a, \mathtt{sc}, d)\})$ is shown in Figure 2 (a). □

The importance of $\mathrm{ecand}(G, H)$ resides in that it defines a partition of the set of models of $G \bullet H$, and a set of formulas whose disjunction represents erase:

Theorem 1 (See [6]). *Let G, H be RDF graphs.*

1. *If $E \in \mathrm{ecand}(G, H)$, then $E \in \mathrm{Mod}(G \bullet H)$.*
2. *If $m \in \mathrm{Mod}(G \bullet H)$ and $m \notin \mathrm{Mod}(G)$, then there is a unique $E \in \mathrm{ecand}(G, H)$ such that $m \models E$.*

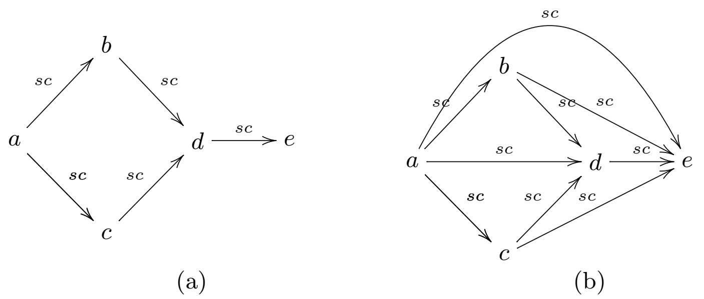


Fig. 1. (a) An RDF Graph G. (b) The closure of G.

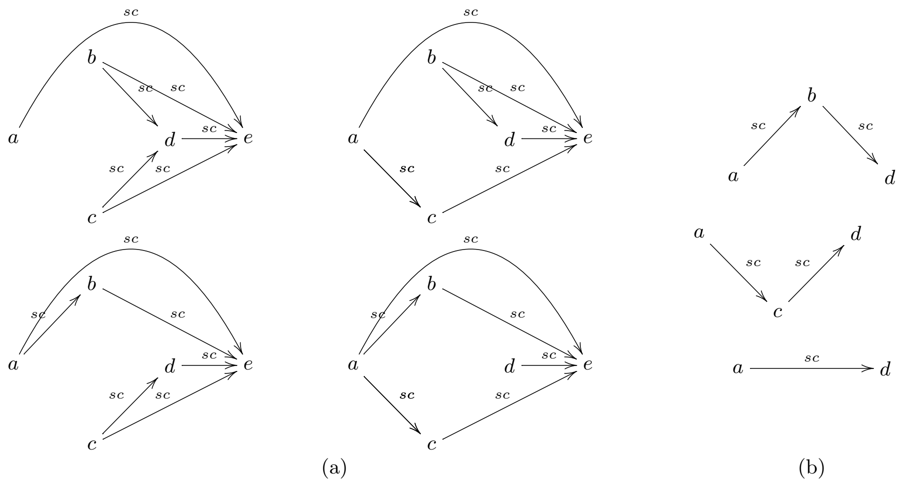


Fig. 2. (a) The set of erase candidates ecand$(G, \{(a, \texttt{sc}, d)\})$. (b) The set of minimal bases $\texttt{minbases}(\text{cl}(G), \{(a, \texttt{sc}, d)\})$.

3. *For all formulas F of RDF, $(\bigcap_{E \in \text{ecand}(G,H)} E) \models F$ if and only if* $\text{Mod}(G \bullet H) \subseteq \text{Mod}(F)$.

Items (1) and (2) in Theorem 1 state that if we had disjunction in RDF, erasure could be expressed by the following finite disjunction of RDF graphs:

$$G \bullet H \quad \text{“="} \quad G \vee E_1 \vee \cdots \vee E_n,$$

where E_j are the erase candidates of $G \bullet H$. Item (3) states the fact that all the statements entailed by $G \bullet H$ expressible in RDF are exactly represented by the RDF graph defined by the intersection of all the erase candidates graphs.

Note that the smaller the size of ecand(G, H), the better the approximation to $G \bullet H$, being the limit the case when it is a singleton:

Corollary 1. *If* ecand$(G, H) = \{E\}$, *then* $(G \bullet H) \equiv E$. □

3 Computing the Erase in RDF

From the discussion above, it follows that approximating $G \bullet H$ reduces to find the erase candidates of this operation. For working purposes, it is easier to work with the 'complement' of them in $\mathrm{cl}(G)$, that we will call *delta candidates*:

Definition 7 (Delta Candidates $\mathrm{dcand}(G,H)$**).** *The set of delta candidates, denoted* $\mathrm{dcand}(G,H)$*, is the set of minimal graphs* $D \subseteq \mathrm{cl}(G)$ *such that* $(\mathrm{cl}(G) \setminus D) \not\models H$*.* □

Thus, the relationship between delta and erase candidates is the following:

$$\mathrm{dcand}(G,H) = \{(\mathrm{cl}(G) \setminus E) : E \in \mathrm{ecand}(G,H)\}. \tag{8}$$

The remainder of this section provides a characterization of delta candidates, based in the notion of *proof tree* (Definition 2).

Definition 8 (Bases and Minimal Bases). *(1) The set of leaves of a proof tree (of H from G) is called the* base *of such proof.*

(2) A base *B of H from G, is a* minimal base *iff it is minimal under set-inclusion among all the bases of proofs of H from G (that is, for every base B' of H from G, it holds $B \subseteq B'$). We denote* $\texttt{minbases}(G,H)$ *the set of minimal bases of G,H.* □

Example 3. For the graph G given in Figure 1 (a), the set $\texttt{minbases}(\mathrm{cl}(G), \{(a,\texttt{sc},d)\})$ contains the graphs given in Figure 2 (b). □

We now need to define the notion of a *hitting set*.

Definition 9 (Hitting Set). *A hitting set for a collection of sets $C_1,\ldots,C_n$ is a set C such that $C \cap C_i$ is non-empty for every C_i. C is called* minimal *if it is a minimal under set-inclusion.* □

Theorem 2. *Let G,H,C be RDF graphs. Then, C is a hitting set for the collection of sets* $\texttt{minbases}(G,H)$ *iff* $(\mathrm{cl}(G) \setminus C) \not\models H$*. Moreover, C is a minimal hitting set iff* $\mathrm{cl}(G) \setminus C$ *is a maximal subgraph G' of cl(G) such that $G' \not\models H$.* □

Proof. (sketch) Note that if C is a hitting set, its minimality follows from the maximality of its complement, $G \setminus C$, and vice versa. Hence we only have to prove that C is a hitting set for $\texttt{minbases}(G,H)$ iff $(G \setminus C) \not\models H$.

Now we are ready to give an operational characterization of delta candidates in terms of hitting sets and minimal bases.

Corollary 2. *Let G,H,C be RDF graphs. $C \in \mathrm{dcand}(G,H)$ if and only if C is a minimal hitting set for* $\texttt{minbases}(\mathrm{cl}(G),H)$*.* □

Proof. Follows from the Definition 7, Theorem 2, and the observation that $C \subseteq \mathrm{cl}(G)$ is minimal iff $\mathrm{cl}(G) \setminus C$ is maximal. □

3.1 Erasing a Triple from a Graph

Now we are ready to present algorithms to compute the delta candidates. We reduce computing erase candidates to finding minimal multicuts in certain directed graphs. The essential case is the deletion of one triple.

Definition 10 (Minimal Cut). *Let (V, E) be a directed graph. A set of edges $C \subseteq E$ disconnects two vertices $u, v \subseteq V$ iff each path from u to v in the graph passes through a vertex in C. In this case C is called a* cut. *This cut is* minimal *if the removal of any node from C does not yield another cut.*

Cuts can be generalized to sets of pairs of vertices yielding multicuts. *A* minimal multicut *for a set of pairs of nodes $(u_1, v_1), (u_2, v_2 g), \ldots, (u_n, v_n)$ is a minimal set of edges that disconnects u_i and v_i, for all i.* □

For a triple t in a graph G, we will show that the graphs in dcand(G, t) correspond to certain cuts defined in two directed graphs derived from G, that we denote $G[\mathtt{sc}]$ and $G[\mathtt{sp}]$, defined as follows:

Definition 11 (Graphs $G[\mathtt{sc}]$ and $G[\mathtt{sp}]$). *Given an RDF graph G, we denote $G[\mathtt{sc}] = (N, E)$ the directed graph defined in Table 2. For each triple of the form specified in the first column of the table, we have the corresponding edges in E. The set of nodes N is composed of all the nodes mentioned in the edges given in the table. The directed graph $G[\mathtt{sp}]$ is defined similarly in Table 2. We use the letters n and m to refer to nodes in $G[\mathtt{sc}]$ and $G[\mathtt{sp}]$, respectively.* □

Table 2. Description of the construction of the graphs $G[\mathtt{sc}]$ (above) and $G[\mathtt{sp}]$ (below)

Triple in G	Edge in $G[\mathtt{sc}]$
$(a, \mathtt{sc}, b)$	(n_a, n_b)
$(a, \mathtt{type}, b)$	$(n_{t,a}, n_b)$
Triple in G	**Edges in $G[\mathtt{sp}]$**
$(p, \mathtt{sp}, q)$	(m_p, m_q)
(a, p, b)	$(m_{a,b}, m_p)$
$(p, \mathtt{dom}, c)$	$(m_p, m_{v,\mathtt{dom}})$ for each $n_c \to^* n_v$ in $G[sc]$
$(p, \mathtt{range}, c)$	$(m_p, m_{v,\mathtt{range}})$ for each $n_c \to^* n_v$ in $G[sc]$

For an RDF triple t, the set of multicuts (set of pairs of nodes) associated to the erasure of t from an RDF graph G, is defined as follows:

Definition 12 (Set of edges $t[\mathtt{sc}, G]$ and $t[\mathtt{sp}, G]$). *The set $t[\mathtt{sc}, G]$ contains the pairs of nodes (u, v) as described in Table 3 (second column) with u, v nodes in $G[\mathtt{sc}]$. Analogously, we define $t[\mathtt{sp}, G]$ using Table 3 (third column).* □

Example 4. Let us consider graph G on the left hand side of Figure 3. The center part and the right hand side show, respectively, the graphs $G[\mathtt{sp}]$ and $G[\mathtt{sc}]$, built according to Table 2. For example, for the triple $t = (d, \mathtt{sc}, c)$, the sets of edges are $(d, \mathtt{sc}, c)[\mathtt{sc}, G] = \{(n_d, n_c)\}$ and $(d, \mathtt{sc}, c)[\mathtt{sp}, G] = \emptyset$. There are triples which

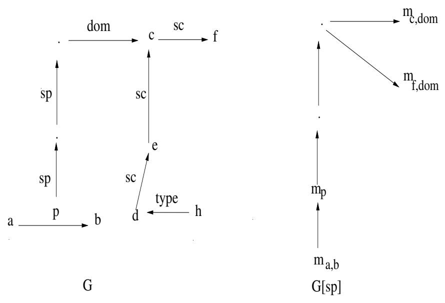


Fig. 3. An RDF graph G and its associated $G[\texttt{sc}]$ and $G[\texttt{sp}]$ graphs

give rise to multiple pairs of nodes. For example, for the triple $t = (a, \texttt{type}, c)$ and the graph in Figure 3, the sets contain the pairs $(a, \texttt{type}, c)[\texttt{sc}, G] = \{(n_{t,a}, n_c)\} \cap G[\texttt{sc}] = \emptyset$, and $(a, \texttt{type}, c)[\texttt{sp}, G] = \{(m_{ab}, m_{c,dom}), (m_{ba}, m_{c,dom})\}$. □

Table 3. Construction of the pairs of nodes $t[\texttt{sc}, G]$ and $t[\texttt{sp}, G]$ associated to a triple t. The minimal multicuts of them in $G[\texttt{sc}]$ and $G[\texttt{sp}]$ respectively, will give the elements of dcand(G, t) (Theorem 3).

$t \in G$	$t[\texttt{sc}, G]$	$t[\texttt{sp}, G]$
$(a, \texttt{sc}, b)$	(n_a, n_b)	–
$(a, \texttt{sp}, b)$	–	(m_a, m_b)
(a, p, b)	–	(m_{ab}, m_p)
$(a, \texttt{type}, c)$	$(n_{t,a}, n_c)$	pairs $(m_{a,x}, m_{c,\texttt{dom}})$ for all x
		pairs $(m_{x,a}, m_{c,\texttt{range}})$ for all x

The next theorem shows that computing the delta candidates can be reduced to compute minimal multicuts, in particular the set of cuts defined in Table 3 in the graphs defined in Table 2.

Theorem 3. *The elements of* dcand(G, t) *are precisely the triples of* G *that correspond (according to the mapping in Table 2) to the minimal multicuts of* $t[\texttt{sc}, G]$ *in* $G[\texttt{sc}]$ *plus the minimal multicuts of* $t[\texttt{sp}, G]$ *in* $G[\texttt{sp}]$.

Proof. The proof is a case-by-case analysis of each form of t. For $t = (a, \texttt{dom}, c)$ or $t = (a, \texttt{range}, c)$, the set dcand$(G, t) = \{t\}$, because t cannot be derived by any rule, thus, $G \not\models t$ if and only if $t \notin G$.

Case $t = (a, sc, b)$. From the deduction rules in Section 2, t can be deduced from G if and only if there is a path in $G[\texttt{sc}]$ from n_a to n_b (note that the only rule that can derive t is (3)). Hence dcand(G, t) is in correspondence with the set of the minimal cuts from n_a to n_b in $G[\texttt{sc}]$.

Case $t = (a, \texttt{sp}, b)$. This is similar to the previous one. ¿From the deduction rules, it follows that t can be only be deduced from G if there is a path in $G[\texttt{sp}]$

from m_a to m_b (by rule (1)). Thus $\text{dcand}(G, t)$ is the set of the minimal cuts from m_a to m_b in $G[\texttt{sp}]$.

Case $t = (a, \texttt{type}, c)$. To deduce t we can use rules (4), (5) and (6). Rule (4) recursively needs a triple of the same form $(a, \texttt{type}, d)$ and additionally the fact that $(d, \texttt{sc}, c)$. Thus, t can be derived from G if there is path in $G[\texttt{sc}]$ from $n_{t,a}$ to n_c. Triple t can also be derived from (5) and (6). Let us analyze (5) (the other case is symmetric). We need the existence of triples (a, P, x) and $(P, \texttt{dom}, u)$ and $u \to^* c$ in $G[\texttt{sc}]$, i.e., $(u, \texttt{sc}, c)$. Then (a, P, x) can be recursively derived by rule(2) (and by *no other* rule); $(P, \texttt{dom}, u)$ should be present; and the last condition needs $(u, \texttt{sc}, c)$. Hence t can be derived if for some x there is a path from $m_{a,x}$ to $m_{c,dom}$ in $G[\texttt{sp}]$ (this explains the two last lines of Table 1).

Analyzing the rules, we can conclude that t is derivable from G if and only if we can avoid the previous forms of deducing it. That is, producing a minimal cut between $n_{t,a}$ and n_c in $G[\texttt{sc}]$ and a minimal multicut between the set of pairs $(m_{ax}, m_{c,dom})$ for all x, and the set of pairs $(m_{y,a}, m_{range,c})$ for all y, in the graph $G[\texttt{sp}]$.

Case $t = (a, p, b)$. Here, t can only be derived using rule (2). This needs the triples (a, q, b) and $(q, \texttt{sp}, p)$. With similar arguments as above, it can be shown that t can be derived from G iff there is path in $G[\texttt{sp}]$ from $m_{a,b}$ to m_p. Hence $\text{dcand}(G, t)$ is the set of minimal cuts from $m_{a,b}$ to m_p in $G[\texttt{sp}]$. $\square$

The complexity of the above process is given essentially by the complexity of finding minimal multicuts:

Theorem 4. *Let G, H be ground RDF graphs, and t be a ground triple. The problem of deciding whether $E \in \text{ecand}(G, t)$ is in PTIME.*

Proof. From Definition 6, the problem reduces to determine if $D = \text{cl}(G) \setminus E$ is a delta candidate in $\text{dcand}(G, t)$. Let $G' = \text{cl}(G)$, G' can be computed in polytime. Theorem 3 shows that we have to test (i) whether $t[\texttt{sc}, G']$ is a minimal cut in $G'[\texttt{sc}]$ and (ii) whether $t[\texttt{sp}, G']$ is a minimal (multi)cut in $G'[\texttt{sp}]$. In both cases the test can be done in PTIME by simple reachability analysis in the graphs $G'[\texttt{sc}]$ and $G'[\texttt{sp}]$, respectively. Testing whether a set of edges S is a minimal cut for (v_1, u_1) in a graph (V, E) can be done performing polytime reachability analysis in the graph as follows. To test whether S is a cut, delete from E the edges in S, and test whether v_1 reaches u_1 in this new graph. To test minimality, do the same test for each set of edges $S' \subset S$ resulting from removing a single edge from S. S is minimal iff all of the S's are not cuts. We proceed similarly for testing if a set of edges is a minimal multicut. $\square$

3.2 Erasing a Graph from a Graph

The problem of computing erase candidates $\text{ecand}(G, H)$ for the case where H has several triples can be easily reduced to the previous one when $H = \{t\}$.

Lemma 1. *Let G, H be ground RDF graphs in normal form (i.e. without redundancies, see [5]). (i) If $E \in \text{ecand}(G, H)$, then there exists a triple $t_i \in H$*

such that $E \in \mathrm{ecand}(G, \{t_i\})$*; (ii) If* $D \in \mathrm{dcand}(G, H)$*, then there exists a triple* $t_i \in H$ *such that* $D \in \mathrm{dcand}(G, \{t_i\})$*.*

Proof. (i) Suppose $G \not\models H$, then there is a triple $t_i \in H$ such that $G \not\models t_i$, which yields $\mathrm{ecand}(G, H) = \{G\} = \mathrm{ecand}(G, \{t_i\})$. Now we assume that $G \models H$. That is $H \subseteq \mathrm{nf}(G)$. Let $T = (H \setminus I)$. T is non-empty because $I \not\models H$ and $\mathrm{nf}(E) = E$. Now if T has more than one triple, then we add one triple of T to I and obtain $I' \in \mathrm{ecand}(G, H)$ which is greater than I contradicting that E is maximal. Therefore T must have exactly one triple, say t_j. In this case can be easily verified that $E = \mathrm{ecand}(G, \{t_j\})$. (ii) Follows directly from (ii).

The intuition of Lemma 1 is that each delete candidate in $\mathrm{dcand}(G, H)$ is also a delete candidate of $\mathrm{dcand}(G, \{t_i\})$ for some triple t_i in H. Therefore, the problem of computing delete candidates reduces to finding the minimal sets among the delete candidates associated to each triple in H.

The following result, consequence of Lemma 1(ii), yields a basic procedure for computing delete candidates: find the minimal cuts for $\mathrm{ecand}(G, \{t\})$ for each triple $t \in H$, and then find the minimum among them.

Proposition 2. *Let* G *and* H *be ground RDF graphs. Then*
$\mathrm{dcand}(G, H) = \min\{D : D \in \bigcup_{t \in H} \mathrm{dcand}(G, t)\}$.

4 Computing the Delta Candidates in Practice

We have seen that if we had disjunction, erase could be expressed as $G \bullet H = G \vee E_1 \vee \cdots \vee E_n$, where the E_i's are the erase candidates. From each E_i we get a delta candidate $D_i = \mathrm{cl}(G) \setminus E_i$. In Section 3 we studied how to compute the D_i's borrowing standard machinery from graph theory. This computation is hard in the general case. In practice, however, there are two factors which turn this computation feasible. First, in RDFS data, as in most data models, schemas are small and stable, and data are large and dynamic. Second, what really matters when computing RDFS updates of schemas are small parts of the schema, essentially the length of the subclass and subproperty chains. Table 1 shows some examples of well-known RDF datasets.

Taking into account these observations, we present practical and feasible algorithms for updating RDF data. We concentrate in the case of a single triple which is the kernel of the operation (as can be deduced from Lemma 1).

4.1 Computing RDF Schema Erasure

We have already reduced the computation of erasure to that of computing the set $\mathrm{ecand}(G, t)$. Algorithm 1 indicates the steps to be done. We have so far studied the decision problem related to computing the set of erase candidates. Generating $\mathrm{ecand}(G, H)$ (respectively $\mathrm{dcand}(G, H)$) requires, in the worst case, time exponential in the size of the graphs $G[sp]$ and $G[sc]$. Indeed, the number of

Algorithm 1. Compute dcand(G,t) (General Case)

```
Input:  Graph $G$, triple $t$
Output: dcand$(G,t)$
1: Compute $G := \mathrm{cl}(G)$
2: Compute $G[sc]$
3: Compute $G[sp]$
4: Compute $t[sc,G]$
5: Compute $t[sp,G]$
6: Compute minimal multicults for $t[sc,G]$ in $G[sc,G]$
7: Compute minimal multicults for $t[sp,G]$ in $G[sp,G]$
```

cuts could be exponential. Standard algorithms on cut enumeration for directed graphs can be adapted to our setting [10].

The good news is that what really matters is the size of the maximal connected component of the graphs (one can avoid cuts between disconnected components). In our case, the size of the connected components of $G[sc]$ and $G[sp]$ are small, and a good estimation of it is the length of the maximal chain of `sc` and `sp` respectively (very small in most real-world datasets). Based on the above, Algorithm 2 would be a feasible one for updating schemas in most practical cases.

Algorithm 2. Update schema of G by erasing t

```
Input:  Graph $G$, triple $t$
Output: $G \bullet t$
1: Choose an ordering on predicates (see e.g. [9], 2.3)
2: Compute dcand$(G,t)$
3: Order the elements $D \in$ dcand$(G,t)$ under this ranking
4: Delete the minimal $D$ from $G$
```

4.2 Computing RDF Instance Erasure

For instance erasure, the situation is optimal, since it assumes that the schema of the graph remains untouched. In this setting, we will show that (i) the procedure is deterministic, that is, there is a unique choice of deletion (i.e., dcand(G,t) has a single element); (ii) this process can be done efficiently.

Algorithm 3 computes dcand(G,t) for instances. The key fact to note is that for instances t, that is, triples of the form $(a, \texttt{type}, b)$ or (a,p,b), where p does not belong to RDFS vocabulary, the minimal multicut is unique. For triples of the form (a,p,b), it follows from Table 3 that one has to cut paths from m_{ab} to m_p in $G[sp]$. Note that nodes of the form m_p are non-leaf ones, hence all edges in such paths in $G[sp]$ come from schema triples $(u, \texttt{sp}, v)$ (see Table 2). Because we cannot touch the schema, if follows that if there is such path, the unique option is to eliminate the edge m_{ab}, which corresponds to triples (a,w,b) in G. For triples of the form $(a, \texttt{type}, b)$ the analysis (omitted here for the sake

Algorithm 3. Compute dcand(G, t) (Optimized version for Instances)

```
Input:  Graph $G$, triple $t$
Output: dcand$(G, t)$
 1:  Compute $G' := \mathrm{cl}(G)$
 2:  Compute $G'[sc]$
 3:  Compute $G'[sp]$
 4:  Compute $t[sc, G']$
 5:  Compute $t[sp, G']$
 6: if $t = (a, \mathtt{type}, b)$ then
 7:     $D \leftarrow \{(a, \mathtt{type}, z) \in G :\ n_z$ reaches $n_b$ in $G'[sc]\ \}$
 8:     $D \leftarrow D \cup \{(a, p, x) \in G :\ m_{ax}$ reaches $m_{b,dom}$ in $G'[sp]\ \}$
 9:     $D \leftarrow D \cup \{(y, p, a) \in G :\ m_{ya}$ reaches $m_{b,range}$ in $G'[sp]\ \}$
10: else
11:     if $t = (a, p, b)$ then
12:         $D \leftarrow \{(a, w, b) \in G :\ m_{ab}$ reaches $m_p$ in $G'[sp]\ \}$
13:     end if
14: end if
15: dcand$(G, t) \leftarrow D$
```

of space) is similar, though slightly more involved. The cost of Algorithm 3 is essentially the computation of the graphs $G[sc]$ and $G[sp]$ and then reachability tests. Once the triples are ordered by subject and object (time $O(n \lg n)$), the rest can be done in linear time.

5 Related Work

Although updates have attracted the attention of the RDF community, only updates to the instance part of an RDF graph have been addressed so far. Sarkar [15] identifies five update operators: and presented algorithms for two of them. Zhan [18] proposes an extension to RQL, and defines a set of update operators. Both works define updates in an operational way. Ognyanov and Kiryakov [14] describe a graph updating procedure based on the addition and the removal of a statement (triple), and Magiridou *et al* [12] introduce RUL, a declarative update language for RDF instances (schema updates are not studied). SPARQL/Update [17] is an extension to SPARQL to support updates over a collection of RDF graphs. The treatment is purely syntactic, not considering the semantics of RDFS vocabulary. In this sense, our work can be considered as input for future enrichments of this language to include RDF semantics.

Konstantinidis *et al.* [9] introduce a framework for RDF/S ontology evolution, based on the belief revision principles of Success and Validity. The authors map RDF to First-Order Logic (FOL), and combine FOL rules (representing the RDF ontology), with a set of validity rules (which capture the semantics of the language), showing that this combination captures an interesting fragment of the RDFS data model. Finally, an ontology is represented as a set of positive *ground* facts, and an update is a set of negative *ground* facts. If the update causes side-effects on the ontology defined above, they chose the best option approach based on the principle of minimal change, for which they define an order between the FOL predicates. The paper overrides the lack of disjunction and negation in RDF by means of working with FOL. Opposite to this approach, in the present paper we remain within RDF.

Chirkova and Fletcher [3], building on [6] and in [9], present a preliminary study of what they call well-behaved RDF schema evolution (namely, updates that are unique and well-defined). They focus in the deletion of a triple from an RDF graph, and define a notion of determinism, showing that when deleting a triple t from a graph G, an update graph for the deletion exists (and is unique), if and only if t is not in the closure of G, or t is deterministic in G. Although closely related to our work, the proposal does not study instance and schema updates separately, and practical issues are not discussed.

Description Logics ontologies can be seen as knowledge bases (KB) composed of two parts, denoted TBox and ABox, expressing intensional and extensional knowledge, respectively. So far, only updates to the extensional part (i.e., *instance* updates) have been addressed. De Giacomo *et al.* [4] study the non-expressibility problem for *erasure.* This states that given a fixed Description Logic $\mathcal{L}$, the result of an *instance level* update/erasure is not expressible in $\mathcal{L}$ (for update, this has already been proved by Liu *et al.* [11]). Here, it is also assumed that the schema remains unchanged (i.e., only the ABox is updated). For update they use the possible models approach [16], and for erasure, the K-M approach. Building also in the ideas expressed in [6], the authors show that for a fragment of Description Logic, updates can be performed in PTIME with respect to the sizes of the original KB and the update formula. Calvanese *et al.* also study updates to ABoxes in DL-Lite ontologies. They present a classification of the existing approaches to evolution, and show that ABox evolution under what they define as *bold semantics* is uniquely defined [2].

6 Conclusions

Following the approach typical in ontology evolution, where schema and instance updates are treated separately, we proposed practical procedures for computing schema and instance RDF erasure, basing ourselves on the K-M approach. We focused in bringing to practice the theory developed on this topic. As one of our main results, we have shown that instance erasure is deterministic and feasible for RDFS. Further, we presented an algorithm to perform it. For schema erasure, the problem is non-deterministic and intractable in the general case. However, we show that since schemas are very small in practice, it can become tractable. Thus, we proposed an algorithm to compute schema updates, based on computing multicuts in graphs. Future work includes developing an update language for RDF based on the principles studied here, and implementing the proposal at big scale.

Acknowledgements. This work was partly done during a visit of C. Gutierrez to INCO, Univ. Republica, Uruguay. C. Gutierrez thanks Project Fondecyt 1110287.

References

1. Arenas, M., Consens, M., Mallea, A.: Revisiting blank nodes in rdf to avoid the semantic mismatch with sparql. In: W3C Workshop: RDF Next Steps, Palo Alto, CA (2010)
2. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of $DL - lite$ 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. 112–128. Springer, Heidelberg (2010)
3. Chirkova, R., Fletcher, G.H.L.: Towards well-behaved schema evolution. In: WebDB (2009)
4. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On instance-level update and erasure in description logic ontologies. J. Log. Comput. 19(5), 745–770 (2009)
5. Gutierrez, C., Hurtado, C.A., Mendelzon, A.O., Pérez, J.: Foundations of semantic web databases. Journal of Computer and System Sciences (JCSS) 77, 520–541 (2010); This is the Journal version of the paper with same title Presented at the PODS Conference Proc. PODS, pp. 95–106 (2004)
6. Gutiérrez, C., Hurtado, C.A., Vaisman, A.A.: The meaning of erasing in RDF under the katsuno-mendelzon approach. In: WebDB (2006)
7. Hayes, P. (ed.): RDF semantics. W3C Working Draft (October 1, 2003)
8. Katsuno, H., Mendelzon, A.O.: On the difference between updating knowledge base and revising it. In: International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, pp. 387–394 (1991)
9. Konstantinidis, G., Flouris, G., Antoniou, G., Christophides, V.: A formal approach for RDF/S ontology evolution. In: ECAI, pp. 70–74 (2008)
10. Lin, H.-Y., Kuo, S.-Y., Yeh, F.-M.: Minimal cutset enumeration and network reliability evaluation by recursive merge and BDD. In: IEEE Symposium on Computers and Communications (ISCC 2003), Kiris-Kemer, Turkey, June 30 - July 3 (2003)
11. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic aboxes. In: KR, pp. 46–56 (2006)
12. Magiridou, M., Sahtouris, S., Christophides, V., Koubarakis, M.: RUL: A declarative update language for RDF. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 506–521. Springer, Heidelberg (2005)
13. Muñoz, S., Pérez, J., Gutierrez, C.: Minimal deductive systems for RDF. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 53–67. Springer, Heidelberg (2007)
14. Ognyanov, D., Kiryakov, A.: Tracking changes in RDF(S) repositories. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 373–378. Springer, Heidelberg (2002)
15. Sarkar, S., Ellis, H.C.: Five update operations for RDF. Rensselaer at Hartford Technical Report, RH-DOES-TR 03-04 (2003)
16. Winslett, M.: Reasoning about action using a possible models approach. In: AAAI, pp. 89–93 (1988)
17. WWW Consortium. SPARQL/Update: A language for updating RDF graphs (2008), http://www.w3.org/Submission/SPARQL-Update/
18. Zhan, Y.: Updating RDF. In: 21st Computer Science Conference, Rensselaer at Hartford (2005)

Benchmarking Matching Applications on the Semantic Web

Alfio Ferrara[1], Stefano Montanelli[1],
Jan Noessner[2], and Heiner Stuckenschmidt[2]

[1] Università degli Studi di Milano,
DICo - via Comelico 39, 20135 Milano, Italy
{ferrara,montanelli}@dico.unimi.it
[2] KR & KM Research Group
University of Mannheim, B6 26, 68159 Mannheim, Germany
{jan,heiner}@informatik.uni-mannheim.de

Abstract. The evaluation of matching applications is becoming a major issue in the semantic web and it requires a suitable methodological approach as well as appropriate benchmarks. In particular, in order to evaluate a matching application under different experimental conditions, it is crucial to provide a test dataset characterized by a controlled variety of different heterogeneities among data that rarely occurs in real data repositories. In this paper, we propose SWING (Semantic Web INstance Generation), a disciplined approach to the semi-automatic generation of benchmarks to be used for the evaluation of matching applications.

1 Introduction

In the recent years, the increasing availability of structured linked data over the semantic web has stimulated the development of a new generation of semantic web applications capable of recognizing identity and similarity relations among data descriptions provided by different web sources. This kind of applications are generally known as *matching applications* and are more and more focused on the specific peculiarities of instance and linked data matching [7]. Due to this situation, the evaluation of matching applications is becoming an emerging problem which requires the capability to measure the effectiveness of these applications in discovering the right correspondences between semantically-related data. One of the most popular approaches to the evaluation of a matching application consists in extracting a test dataset of linked data from an existing repository, such as those available in the linked data project, in deleting the existing links, and in measuring the capability of the application to automatically restore the deleted links. However, the datasets extracted from a linked data repository, suffer of three main limitations: i) the majority of them are created by acquiring data from web sites using automatic extraction techniques, thus both data and links are not validated nor checked; ii) the logical structure of these datasets is usually quite simple and the level of semantic complexity quite low; iii) the number and kind of dissimilarities among data is not controlled, so that it is difficult to tailor the evaluation on the specific problems that affect the currently considered matching application.

In this context, a reliable approach for the evaluation of matching applications has to address the following requirements: i) the test dataset used for the evaluation must be coherent to a given domain and must be represented according to the desired level of structural and semantic complexity; ii) the evaluation must be executed by taking into account a controlled variety of dissimilarities among data, including value, structural, and logical heterogeneity; iii) a ground-truth must be available, defined as a set of links among data that the matching application under evaluation is expected to discover. In order to address these requirements, we propose SWING (Semantic Web INstance Generation) a disciplined approach to the semi-automatic generation of benchmarks to be used for the evaluation of matching applications. A SWING benchmark is composed of a set of test cases, each one represented by a set of instances and their corresponding assertions (i.e., an OWL ABox) built from an initial dataset of real linked data extracted from the web. SWING aims at supporting the work of an *evaluation designer*, who has the need to generate a tailored benchmark for assessing the effectiveness of a certain matching application. The SWING approach has been implemented as a Java application and it is available at `http://code.google.com/p/swing`.

The paper is organized as follows. In Section 2, we discuss some related work on the subject of matching evaluation. In Section 3, we summarize our approach. In Section 4 and Section 5 we present the SWING acquisition and transformation techniques, respectively. In Section 6, we present the experimental results obtained by executing six different matching algorithms over the benchmark. In Section 7, we give our concluding remarks.

2 Related Work

In the last years, significant research effort has been devoted to ontology matching with special attention on techniques for instance matching [7]. In the literature, most of the existing approaches/techniques use their individually created benchmarks for evaluation, which makes a comparison difficult or even impossible [10]. In the fields of object reconciliation, duplicate detection, and entity resolution, which are closely related to instance matching, a widely used set of benchmarks are proposed by the Transaction Processing Performance Council (TPC)[1] that focuses on evaluating transaction processing and databases. A number of benchmarks are also available for XML data management. Popular examples are presented in [3,5]. Since these datasets are not defined in a Semantic Web language (e.g., OWL) their terminological complexity is usually very shallow. In the area of ontology matching, the Ontology Alignment Evaluation Initiative (OAEI) [6] organizes since 2005 an annual campaigns aiming at evaluating ontology matching technologies through the use of common benchmarks.

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

However, the main focus of the past OAEI benchmarks was to compare and evaluate schema-level ontology matching tools. From 2009, a new track specifically focused on instance matching applications has been introduced in OAEI and a benchmark has been developed to this end [8]. The weakness of this 2009 OAEI benchmark is the basic level of flexibility enforced during the dataset creation and the limited size of the generated test cases. The benchmark provided by Yatskevich et al. [13] is based on real-world data, using the taxonomy of Google and Yahoo as input. In [13], the limit of the proposed approach is the problem to create an error-free gold standard, since the huge size of the datasets prevents a manual alignment. Intelligent semi-automatic approximations are used to overcome such a weakness, however it is not possible to guarantee that all the correct correspondences are found and that none of the found correspondences is incorrect. The same problem raises with the linked data benchmarks DI[2] and VLCR[3]. Alexe et al. [1] provide a benchmark for mapping systems, which generate schema files out of a number of given parameters. Their automatic generation process ensures that a correct gold standard accrues. However, real-world data are not employed and artificial instances with meaningless content are mainly considered. Other benchmarks in the area of ontology and instance matching are presented in [13] and [9]. In these cases, the weak point is still the limited degree of flexibility in generating the datasets of the benchmark. We stress that the proposed SWING approach provides a general framework for creating benchmarks for instance matching applications starting with a linked data source and ending with various transformed ABox ontologies. In particular, the SWING approach combines the strength of both benchmarks [1] and [12] by taking real-world data from the linked data cloud as input and by performing transformations on them which ensure that the gold standard must be correct in all cases. Moreover, a further contribution of the SWING approach is the high level of flexibility enforced in generating the datasets through data transformations that is a widely recognized weak point of the other existing benchmarks.

3 The SWING Approach

The SWING approach is articulated in three phases as shown in Figure 1.

Data acquisition techniques. SWING provides a set of techniques for the acquisition of data from the repositories of linked data and their representation as a reference OWL ABox. In SWING, we work on open repositories by addressing two main problems featuring this kind of datasources. First, we support the evaluation designer in defining a subset of data by choosing both the data categories of interest and the desired size of the benchmark. Second, in the data enrichment activity, we add semantics to the data acquired. In particular, we adopt specific ontology design patterns that drive the evaluation designer in defining a data description scheme capable of supporting the simulation of a wide spectrum of data heterogeneities.

[2] `http://www.instancematching.org/oaei/imei2010.html`
[3] `http://www.cs.vu.nl/~laurah/oaei/2010/`

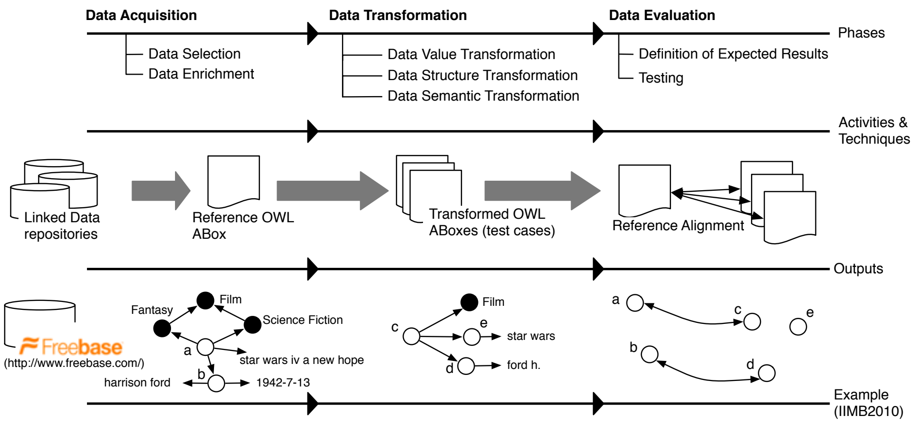


Fig. 1. The SWING approach

Data transformation techniques. In the subsequent *data transformation* activity the TBox is unchanged, while the ABox is modified in several ways by generating a set of new ABoxes, called *test cases*. Each test case, is produced by transforming the individual descriptions in the reference ABox in new individual descriptions that are inserted in the test case at hand. The goal of transforming the original individuals is twofold: on one side, we provide a simulated situation where data referred to the same objects are provided in different datasources; on the other side, we generate a number of datasets with a variable level of data quality and complexity.

Data evaluation techniques. Finally, in the *data evaluation* activity, we automatically create a ground-truth as a reference alignment for each test case. A reference alignment contains the mappings (in some contexts called "links") between the reference ABox individuals and the corresponding transformed individuals in the test case. These mappings are what an instance matching application is expected to find between the original ABox and the test case.

As a running example illustrating our approach, in this paper, we present the IIMB 2010 benchmark[4], which has been created by applying our SWING approach. IIMB 2010 has been used in the instance matching track of OAEI 2010. IIMB 2010 is a collection of OWL ontologies consisting of 29 concepts, 20 object properties, 12 data properties and thousands of individuals divided into 80 test cases. In fact in IIMB 2010, we have defined 80 test cases, divided into 4 sets of 20 test cases each. The first three sets are different implementations of data value, data structure, and data semantic transformations, respectively, while the fourth set is obtained by combining together the three kinds of transformations. IIMB 2010 is created by extracting data from Freebase [2], an open knowledge

[4] `http://www.instancematching.org/oaei/imei2010.html`

base that contains information about 11 Million real objects including movies, books, TV shows, celebrities, locations, companies and more. Data extraction has been performed using the query language JSON together with the Freebase JAVA API[5].

4 Data Acquisition

The SWING data acquisition phase is articulated in two tasks, called data selection and data enrichment. The task of *data selection* has the aim to find the right balance between the creation of a realistic benchmark and the manageability of the dataset therein contained. In SWING, the data selection task is performed according to an initial query that is executed against a linked data repository with the supervision of the evaluation designer. In particular, the size of the linked data source is narrowed down by i) selecting a specific subset of all available linked data classes and ii) limit the individuals belonging to these selected classes. With the latter selection technique, we can easily scale the number of individuals from hundreds to millions only by adjusting one single parameter. The goal of the *data enrichment* task is to provide a number of data enrichment techniques which can be applied to any linked data source for extending its structural and semantic complexity from the description logic $\mathcal{ALE}(\mathcal{D})$ up to $\mathcal{ALCHI}(\mathcal{D})$. This data enrichment has to be realized, because in the open linked data cloud the concept hierarchies are usually very low and disjointness axioms or domain and range restrictions are rarely defined. The limited level of semantic complexity is a distinguishing feature of linked data. Nevertheless, many matching applications are capable of dealing with data at different levels of OWL expressiveness.

To illustrate the SWING enrichment techniques, we will refer to a small snippet of the IIMB 2010 benchmark displayed in Figure 4. The black colored nodes, arrows, and names represent information that has been extracted from Freebase, while the gray colored information has been added according to the following enrichment techniques of our SWING approach.

Add Super Classes and Super Properties. The designer can adopt two different approaches for determining new super classes. The first one is a bottom-up approach where new super classes are created by aggregating existing classes. Thereby, the evaluation designer has to define a super class name which encompasses all the classes to include. The second approach is top-down and it requires to define how to split a class into more subclasses. The same approaches can be applied for determining super object properties, respectively. This task is mainly performed manually by the designer, with the support of the system to avoid the insertion of inconsistency errors.

In IIMB we added for instance following statements for classes and object properties:

[5] http://code.google.com/p/freebase-java/. However, we stress that any kind of linked data-compatible source can be used to implement the SWING approach.

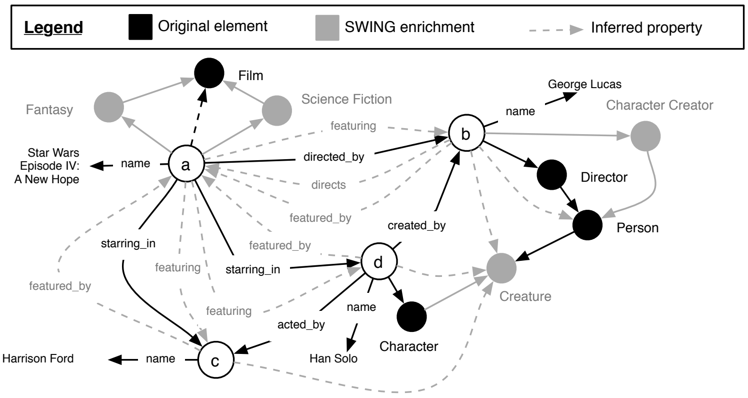


Fig. 2. A portion of IIMB 2010

$$(Person \sqcup Character) \sqsubseteq Creature$$

$$(directed\_by \sqcup acted\_by \sqcup starring\_in) \sqsubseteq featuring$$

Convert Attributes to Class Assertions. Sometimes, linked data sources contain string attributes which can be "upgraded" to classes. A good indicator for such attributes is if the data values are restricted to a number of predefined expressions like for example *male* and *female* or a restricted number of terms denoting concepts, like for example *red*, *green*, and *yellow*. In this case, an external lexical system such as for example WordNet can be used to support the designer in finding those terms that can be considered as candidates for the class assertion conversion. For example, in Freebase, every film has an attribute *genre* with values like *Fantasy*, *Science Fiction*, *Horror*, and many more. This attribute was used in IIMB 2010 to derive additional class assertions as subclasses of *Film* as shown in Figure 4.

$$(Fantasy \sqcup Science\ Fiction \sqcup Horror \sqcup ...) \sqsubseteq Film$$

Determine Disjointness Restrictions. The challenge of this task is to add as many disjointness restrictions as possible while ensuring the ontology consistency. This could be realized by trying to add all possible disjointness combinations and by checking the consistency of the ontology after each combination. If the ontology does not turn to inconsistency we integrate the axiom, otherwise, we discard it. In general, disjointness axioms can not only be added for classes, but also for object properties. For the sake of readability, disjointness axioms are not shown in Figure 4. However, in IIMB 2010, the classes *Film*, *Location*, *Language*, *Budget*, and *Creature* were set to be pairwise disjoint.

Enrich with Inverse Properties. We perform the insertion of inverse properties by creating an object property with both the active and the passive verb forms used as property names. To automate this procedure, we rely on a lexical systems (e.g., WordNet) to determine the active/passive verb forms to insert. Moreover, the evaluation designed can manually insert the inverse property for those property names that are not retrieved in WordNet. For example, in IIMB 2010, we added the property *directs* as the inverse of the existing property *directed_by*.

Specify Domain and Range Restrictions. For all existing properties, the challenge is to find the best - that means the narrowest - domain and range without turning the ontology to inconsistency. For an object property, all the possible domain and range restrictions can be determined by attempting to assign all the existing classes to be the potential domain/range. If the ontology is still consistent after the assignment of this new domain/range restriction, the change is saved, otherwise it is discarded. In the example of IIMB 2010, among the others, the following domain and range restrictions have been added to the object property *created_by*:

$$\exists created\_by \sqsubseteq Character \quad \wedge \quad \exists created\_by^{-} \sqsubseteq Character\ Creator$$

5 Data Transformation

Once the data acquisition phase is executed and a reference ontology $\mathcal{O}$ is produced, we start the SWING data transformation phase that has the goal of producing a set $\mathcal{T} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_{n-1}, \mathcal{O}_n\}$ of new ontologies, called test cases. Each test case $\mathcal{O}_i \in \mathcal{T}$ has the same schema (i.e., TBox) of $\mathcal{O}$ but a different set of instances (ABox) generated by transforming the ABox $\mathcal{A}_{\mathcal{O}}$ of $\mathcal{O}$. In detail, the input of each transformation is a reference ontology $\mathcal{O}$ and a configuration scheme $\mathcal{C}$, which contains the specification of properties involved in the transformations process, the kind of transformations enforced, and the parameters required by the transformation functions. The output is a new ontology $\mathcal{O}_i$. The implementation of each ontology transformation can be described in terms of a transformation function $\theta : \mathcal{A}_{\mathcal{O}} \rightarrow \mathcal{A}^i_{\mathcal{O}}$, where $\mathcal{A}_{\mathcal{O}}$ and $\mathcal{A}^i_{\mathcal{O}}$ denote two ABoxes consistent with the TBox $\mathcal{T}_{\mathcal{O}}$ of the ontology $\mathcal{O}$. The transformation function θ maps each assertion $\alpha_k \in \mathcal{A}_{\mathcal{O}}$ into a new set of assertions $\theta(\alpha_k)_{\mathcal{C}}$ according to the configuration scheme $\mathcal{C}$. Thus, given a configuration scheme $\mathcal{C}$ and an ontology $\mathcal{O}$, a test case $\mathcal{O}_i$ is produced as follows:

$$\mathcal{O}_i = \mathcal{T}_{\mathcal{O}} \cup \mathcal{A}^i_{\mathcal{O}}\ with\ \mathcal{A}^i_{\mathcal{O}} = \bigcup_{k=1}^{N} \theta(\alpha_k)_{\mathcal{C}}$$

where N is the number of assertions in $\mathcal{A}_{\mathcal{O}}$. In SWING, we take into account two kinds of assertions for the transformation purposes, namely class assertions and property assertions. A class assertion has the form $C(x)$ and denotes the fact that and individual x is an instance of class C (i.e., the type of x is C). A property assertion has the form $P(x, y)$ and denotes the fact that an individual x is featured by a property P which has value y. P may be either an object property or a data property. In the first case, the value y is another individual, while in the second case y is a concrete value. As an example, in the reference ontology used for IIMB 2010, we have the following assertions:

$$\alpha_1 : Director(b),\ \alpha_2 : name(b, \text{“George Lucas”}),\ \alpha_3 : created\_by(d, b)$$

denoting the fact that b is a *Director* whose name is represented by the string “George Lucas”. Moreover the object denoted by the individual d is created by the individual b (d denotes the character “Han Solo” as shown in Figure 2). Both the kinds of assertions taken into account in SWING can be described in terms of an assertion subject, denoted α^s, that is the individual which the assertion α is referred to, an assertion predicate, denoted α^p, that is the RDF property $rdf : type$ in case of class assertions or the property involved in the assertion in case of property assertions, and an assertion object, denoted α^o, that is a class in case of class assertions and a concrete or abstract value in case of property assertions. For example, referring to the IIMB 2010 example above, we have $\alpha_1^s = b$, $\alpha_1^p = rdf : type$, $\alpha_1^o = Director$ and $\alpha_2^s = b$, $\alpha_2^p = name$, $\alpha_2^o =$ “George Lucas”. According to this notation, we can define the individual description D_j of and individual j into an ABox $\mathcal{A}_\mathcal{O}$ as follows:

$$D_j = \{\alpha_k \in \mathcal{A}_\mathcal{O} \mid \alpha_k^s = j\}$$

that is the set of all assertions in $\mathcal{A}_\mathcal{O}$ having j as subject. According to this notion of individual description, we define also the notion of individual transformation $\theta(j)$ as the result of the transformation $\theta(\alpha_k)$ of each assertion α_k in the definition D_j of j.

5.1 Data Transformation Procedure

The data transformation of an ontology $\mathcal{O}$ into an ontology $\mathcal{O}_i$ is run as a procedure articulated in three steps:

Preprocessing of the Initial Ontology. The preprocessing step has the goal of adding some axioms to the ontology TBox $\mathcal{O}$, that will be the reference TBox for the rest of the transformation and will be identical for all the test cases. These additions are required in order to implement some of the subsequent data transformations without altering the reference TBox. In particular, we add two kind of axioms. As a first addition, we take into account all the data properties $P_i \in \mathcal{O}$ and, for each property, we add a new object property R_i, such that $\mathcal{O} = \mathcal{O} \cup R_i$. Moreover, we add a data property *has_value* to $\mathcal{O}$. These additions are required for transforming data property assertions into object property assertions. The second addition is performed only if the semantic complexity of the ontology chosen by the designer allows the usage of inverse properties. In this case, we take into account all the object properties R_i that are not already associated with an inverse property and we add to $\mathcal{O}$ a new property K_i such that $K_i \equiv R_i^-$.

Deletion/Addition of Individuals. The SWING approach allows the evaluation designer to select a portion of individuals that must be deleted and/or duplicated in the new ontology. The reason behind this functionality is to obtain a new ontology where each original individual can have none, one, or more matching counterparts. The goal is to add some noise in the expected mappings in such a way that the resulting benchmark contains both test cases where each original instance has only one matching counterpart (i.e., one-to-one mappings) and test cases where each original instance may have more than one matching counterpart (i.e., one-to-many mappings). This is a required feature in a benchmark to avoid that those matching applications that produce only one-to-one mappings are favored. In particular, the evaluation designer can choose the percentage t_d (expressed in the range [0,1]) of individuals that are candidate for deletion and the percentage t_a of individuals that are candidate for addition. Then, given the number N_I of individuals in the initial ontology $\mathcal{O}$, SWING calculates the number C_I of individuals that have to be deleted as $C_I = \lfloor t_d \cdot N_I \rfloor$. Given C_I, two strategies, called *deterministic* and *non-deterministic* strategies, are defined to randomly choose the individuals to eliminate. In the deterministic strategy, we randomly choose C_I individuals from $\mathcal{O}$. The assertions in the descriptions of the chosen individuals are simply not submitted to transformation and, thus, do not appear in the new ontology $\mathcal{O}_i$. In the non-deterministic strategy, we take into account all the individuals in $\mathcal{O}$ once at a time. For each individual, we generate a random value r in the range [0,1]. If $r \leq t_d$, the individual is not submitted to transformation. Every time the transformation is not executed, we add 1 to a counter c. This procedure is iterated until $c < C_I$ or all the individuals in $\mathcal{O}$ have been considered. The advantage of the deterministic strategy is that it is possible to control the exact number of transformations that are generated. The advantage of a non-deterministic choice is to keep the transformation process partially blind even for the evaluation designer. Similarly, the number A_I of individuals to be added is calculated as $A_I = \lfloor t_a \cdot (N_I - C_I) \rfloor$. We randomly select the individuals to be added and for each of these individuals i, we create a new individual i' in the text case by substituting the individual identifier i with a new randomly generated identifier i'. Then, each assertion $\alpha_k \in D_i$ is transformed by substituting any reference to i with i'. In such a way, in the test case we will have a copy of each individual description plus the individual description transformation $\theta(i)$.

Individuals Transformation. For each individual description D_i and for each assertion $\alpha_j \in D_i$, we calculate the transformation $\theta(\alpha_j)_{\mathcal{C}}$ according to the configuration scheme $\mathcal{C}$, that is defined by the evaluation designer. Every transformation $\theta(\alpha_j)_{\mathcal{C}}$ is seen as a ordered sequence of transformation operations. Each operation takes a set A of assertions in input and returns a set A' of transformed assertions as output. The input of the first transformation operation is the singleton set $\{\alpha_j\}$, while the output of the final operation in the sequence is the transformation $\theta(\alpha_j)_{\mathcal{C}}$. Transformation operations are distinguished in three categories, namely *data value transformation*, *data structure transformation*, and *data semantic transformation*. Due to space reasons, we cannot describe operations in detail, but we summarize them in Table 1 and we will provide an example of their application in the following.

Table 1. Summary of data transformation operations provided by SWING

	Data Value	Data Structure	Data Semantic
Add	γ	σ, ζ	ι
Delete	ρ	δ	λ, π, ι
Modify	ρ, κ	τ, ζ	λ, ω, π

γ = Random token/character addition
ρ = Random token/character modification
κ = Specific data modification
σ = Property assertion addition
ζ = Property assertion splitting
δ = Property assertion deletion
τ = Property type transformation
ι = Creation of inverse property assertions
λ = Deletion/modification of class assertions
π = Creation of super-property assertions
ω = Classification of individuals in disjoint classes

Data value transformation operations work on the concrete values of data properties and their datatypes when available. The output is a new concrete value. An example of data value transformation, is shown in Table 2.

Table 2. Examples of data value transformations

Operation	Original value	Transformed value
Standard transformation	Luke Skywalker	L4kd Skiwaldek
Date format	1948-12-21	December 21, 1948
Name format	Samuel L. Jackson	Jackson, S.L.
Gender format	Male	M
Synonyms	Jackson has won multiple awards [...]	Jackson has gained several prizes [...]
Integer	10	110
Float	1.3	1.30

Data structure transformation operations change the way data values are connected to individuals in the original ontology graph and change the type and number of properties associated with a given individual. A comprehensive example of data structure transformation is shown in Table 3, where an initial set of assertions A is transformed in the corresponding set of assertions A' by applying the property type transformation, property assertion deletion/addition, and property assertion splitting.

Finally, data semantic transformation operations are based on the idea of changing the way individuals are classified and described in the original ontology. For the sake of brevity, we illustrate the main semantic transformation operations by means of the following example, by taking into account the portion of $\mathcal{T}_\mathcal{O}$ and the assertions sets A and A' shown in Table 4.

Table 3. Example of data structure transformations

A	A'
$name(n,$ "Natalie Portman"$)$	$name(n,$ "Natalie"$)$
$born\_in(n, m)$	$name(n,$ "Portman"$)$
$name(m,$ "Jerusalem"$)$	$born\_in(n, m)$
$gender(n,$ "Female"$)$	$name(m,$ "Jerusalem"$)$
$date\_of\_birth(n,$ "1981-06-09"$)$	$name(m,$ "Auckland"$)$
	$obj\_gender(n, y)$
	$has\_value(y,$ "Female"$)$

Table 4. Example of data semantic transformations

$\mathcal{T}_{\mathcal{O}}$
$Character \sqsubseteq Creature$, $created\_by \equiv creates^{-}$, $acted\_by \sqsubseteq featuring$, $Creature \sqcap Country \sqsubseteq \bot$

A	A'
$Character(k)$	$Creature(k)$
$Creature(b)$	$Country(b)$
$Creature(r)$	$\top(r)$
$created\_by(k, b)$	$creates(b, k)$
$acted\_by(k, r)$	$featuring(k, r)$
$name(k,$ "Luke Skywalker"$)$	$name(k,$ "Luke Skywalker"$)$
$name(b,$ "George Lucas"$)$	$name(b,$ "George Lucas"$)$
$name(r,$ "Mark Hamill"$)$	$name(r,$ "Mark Hamill"$)$

In the example, we can see how the combination of all the data semantic operations may change the description of the individual k. In fact, in the original set A, k (i.e., the Luke Skywalker of Star Wars) is a character created by the individual b (i.e., George Lucas) and acted by r (i.e., Mark Hamill). In A' instead, k is a more generic "creature" and also the relation with r is more generic (i.e., "featuring" instead of "acted_by"). Moreover, individual k is not longer *created_by* b as it was in A, but it is b that *creates* k. But the individual b of A' cannot be considered the same than $b \in A$, since the class *Creature* and *Country* are disjoint.

According to Table 1, data transformation operations may also be categorized as operations that *add*, *delete* or *modify* the information originally provided by the initial ontology. Table 1 shows also how some operations are used in order to implement more than one action over the initial ontology, such as in case of deletion and modifications of string tokens that are both implemented by means of the operation ρ. Moreover, some operations cause more than one consequence on the initial ontology. For example, the property assertion splitting ζ causes both the modification of the original property assertion and the addition of some new assertions in the new ontology.

5.2 Combining Transformations and Defining the Expected Results

When a benchmark is generated with SWING it is usually a good practice to provide a set of test cases for each category of data transformation plus a fourth bunch of test cases where data value, data structure, and data semantic transformations are combined together. The combination of different transformations in SWING is easy both in case of transformations of the same category and in case of cross-category transformations. In fact, all the transformation operations work on assertions sets and produce other assertions sets. Thus, the combination is obtained by executing the desired transformations one over the output of the other. As an example, we consider the initial assertion set $A = \{name(b, \text{“George Lucas”})\}$ and the transformation sequence $A \rightarrow \rho(A, 0.5) \rightarrow A' \rightarrow \tau(A', 1.0) \rightarrow A''\iota(A'', 1.0) \rightarrow A'''$ that produces the following results: $A = name(b, \text{“George Lucas”})$, $A' = name(b, \text{“YFsqap Lucas”})$, $A'' = obj\_name(b, x)$, $has\_value(x, \text{“YFsqap Lucas”})$, $A''' = obj\_name^{-}(x, b)$, $has\_value(x, \text{“YFsqap Lucas”})$.

As a last step in the benchmark generation, for each test case $\mathcal{O}_i$ we define a set of mappings $\mathcal{M}_{\mathcal{O},\mathcal{O}_i}$ that represents the set of correspondences between the individuals of the ontology $\mathcal{O}$ and the individuals of the the test case $\mathcal{O}_i$ that a matching application is expected to find when matching $\mathcal{O}$ against $\mathcal{O}_i$. For each individual j in $\mathcal{O}$ we define a new individual j' and we substitute any reference to j in $\mathcal{O}_i$ with a reference to j'. Then, if the pair j, j' is not involved in any operation of classification of individuals in disjoint classes, we insert $m(j, j')$ in $\mathcal{M}_{\mathcal{O},\mathcal{O}_i}$.

A comprehensive example of data transformation taken form IIMB 2010 is shown in Figure 3, together with the expected results generated for the test case at hand.

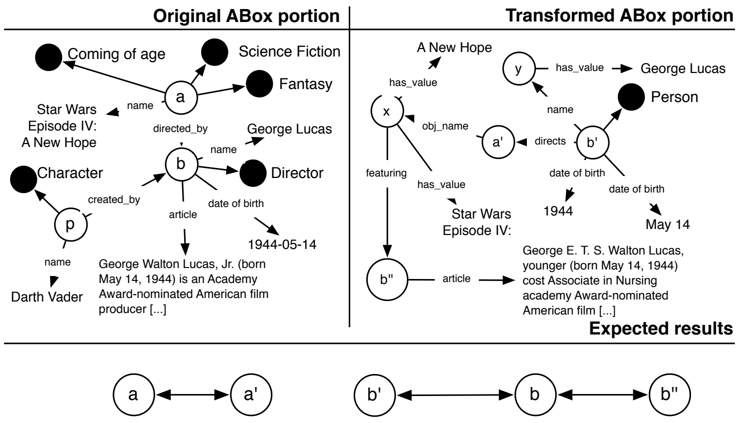


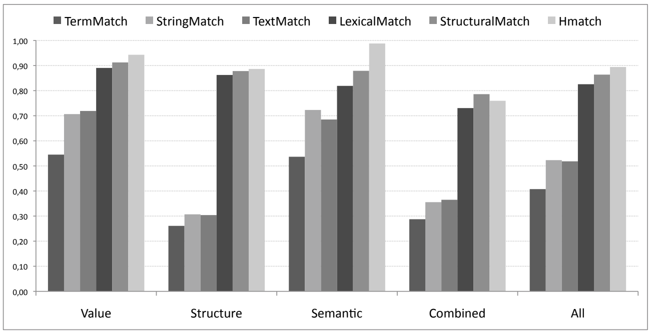
Fig. 3. Example of data transformation taken from IIMB 2010

6 Experimental Results

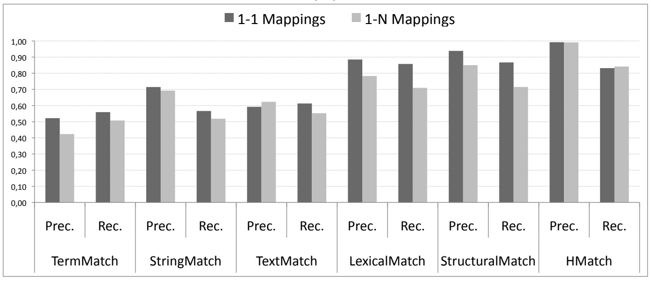
In order to evaluate the applicability of the SWING approach to the evaluation of matching applications, we have executed six different matching algorithms against IIMB 2010 and we have compared precision and recall of each algorithm against the different test cases of the benchmark. The six algorithms have been chosen to represent some of the most popular and reliable matching techniques in the field of instance matching. We recall that the goal of our evaluation is to verify the capability of the IIMB 2010 benchmark generated with SWING to provide a reliable and sufficiently complete dataset to measure the effectiveness of different and often complementary matching algorithms/applications. The considered matching algorithms are divided into two categories, namely *simple matching* and *complex matching*. Simple matching algorithms are three variations of string matching functions that are used for the comparison of a selected number of property values featuring the test case instances. Complex matching algorithms work on the structure and semantics of test cases and on the expected cardinality of resulting mappings. In this category we executed the algorithms *LexicalMatch*, *StructuralMatch*, and *HMatch*. LexicalMatch and StructuralMatch [11] use integer linear programming to calculate the optimal one-to-one alignment based on the sum of the lexical similarity values. In LexicalMatch, no structural information is considered. StructuralMatch uses both, lexical and structural information. Finally, HMatch [4] is a flexible matching suite where a number of matching techniques are implemented and organized in different modules providing linguistic, concept, and instance matching techniques that can be invoked in combination.

A summary of the matching results is reported in Figure 4(a), where we show the average values of the harmonic mean of precision and recall (i.e., FMeasure) for each algorithm over data value, structure and semantic transformation test cases, respectively. The last two chunks of results refer to a combination of transformations and to the benchmark as a whole, respectively.

The goal of our experiments is to evaluate the IIMB 2010 *effectiveness*, that is the capability of distinguishing among different algorithms where they are tested on different kinds of transformations. To this end, we observe that IIMB 2010 allows to stress the difference between simple and complex matching algorithms. In fact, in Figure 4(a), the FMeasure for simple matching algorithms is between 0.4 and 0.5, while we obtain values in the range 0.8-0.9 with complex algorithms. It is interesting to see how simple algorithms have best performances on value transformations and worst performances on structural transformations. This result is coherent with the fact that simple matching does not take into account neither the semantics nor the structure of individuals, and proves that SWING simulates structural transformation in a correct way. In case of semantic transformations instead, simple algorithms have quite good performances because many semantic transformations affect individual classification, which is an information that is ignored by simple algorithms. In Figure 4(b), we show the values of precision and recall of the considered algorithms in the test cases where all the expected mappings were one-to-one mappings (i.e., 1-1 Mappings) and in the test cases where one-to-many mappings were expected for 20% of the individuals (i.e., 1-N Mappings). We note that the algorithms that are not based on the assumption of finding one-to-one mappings (i.e., simple matching algorithms and HMatch) have similar results in case of 1-1 and 1-N mappings. Instead, LexicalMatch and StructuralMatch are based on the idea of finding the best 1-1 mapping set. Thus, precision and recall of these algorithms are lower



(a)



(b)

Fig. 4. Results for the execution of different matching algorithms against IIMB 2010

when 1-N mappings are expected. The number of individuals corresponding to more than one individual is about 20% of the total number of individuals and this percentage corresponds to the degradation of results that we can observe for LexicalMatch and StructuralMatch.

7 Concluding Remarks

In this paper we have presented SWING, our approach to the supervised generation of benchmarks for the evaluation of matching applications. Experiments presented in the paper show that SWING is applicable to the evaluation of real matching applications with good results. Our future work is focused on collecting more evaluations results in the instance matching evaluation track of OAEI 2010, where SWING has been used to generate the IIMB 2010 benchmark. Moreover, we are interested in studying the problem of extending SWING to the creation of benchmarks for evaluation of ontology matching applications in general, by providing a suite of comprehensive evaluation techniques and tools tailored for the specific features of TBox constructs.

Acknowledgements. Heiner Stuckenschmidt is partially supported by the European project SEALS (IST-2009-238975).

References

1. Alexe, B., Tan, W.C., Velegrakis, Y.: STBenchmark: towards a Benchmark for Mapping Systems. Proc. of the VLDB Endowment 1(1), 230–244 (2008)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proc. of the ACM SIGMOD Int. Conference on Management of Data, pp. 1247–1250 (2008)
3. Bressan, S., Li Lee, M., Guang Li, Y., Lacroix, Z., Nambiar, U.: The XOO7 Benchmark. In: Proc. of the 1st VLDB Workshop on Efficiency and Effectiveness of XML Tools, and Techniques, EEXTT 2002 (2002)
4. Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. Journal on Data Semantics V (2006)
5. Duchateau, F., Bellahse, Z., Hunt, E.: XBenchMatch: a Benchmark for XML Schema Matching Tools. In: Proc. of the 33rd Int. Conference on Very Large Data Bases, VLDB 2007 (2007)
6. Euzenat, J., Ferrara, A., Hollink, L., et al.: Results of the Ontology Alignment Evaluation Initiative 2009. In: Proc. of the 4th Int. Workshop on Ontology Matching, OM 2009 (2009)
7. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
8. Ferrara, A., Lorusso, D., Montanelli, S., Varese, G.: Towards a Benchmark for Instance Matching. In: Proc. of the ISWC Int. Workshop on Ontology Matching, OM 2008 (2008)
9. Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 274–288. Springer, Heidelberg (2004)
10. Koepcke, H., Thor, A., Rahm, E.: Evaluation of Entity Resolution Approaches on Real-World Match Problems. In: Proc. of the 36th Int. Conference on Very Large Data Bases, VLDB 2010 (2010)
11. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging Terminological Structure for Object Reconciliation. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 334–348. Springer, Heidelberg (2010)
12. Perry, M.: TOntoGen: A Synthetic Data Set Generator for Semantic Web Applications. AIS SIGSEMIS Bulletin 2(2), 46–48 (2005)
13. Yatskevich, M., Giunchiglia, F., Avesani, P.: A Large Scale Dataset for the Evaluation of Matching Systems. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, Springer, Heidelberg (2007)

Efficiently Evaluating Skyline Queries on RDF Databases

Ling Chen, Sidan Gao, and Kemafor Anyanwu

Semantic Computing Research Lab, Department of Computer Science
North Carolina State University
{lchen10,sgao,kogan}@ncsu.edu

Abstract. Skyline queries are a class of preference queries that compute the pareto-optimal tuples from a set of tuples and are valuable for multi-criteria decision making scenarios. While this problem has received significant attention in the context of single relational table, skyline queries over joins of multiple tables that are typical of storage models for RDF data has received much less attention. A naïve approach such as a *join-first-skyline-later* strategy splits the join and skyline computation phases which limit opportunities for optimization. Other existing techniques for multi-relational skyline queries assume storage and indexing techniques that are not typically used with RDF which would require a preprocessing step for data transformation. In this paper, we present an approach for optimizing skyline queries over RDF data stored using a vertically partitioned schema model. It is based on the concept of a "*Header Point*" which maintains a concise summary of the already visited regions of the data space. This summary allows some fraction of non-skyline tuples to be pruned from advancing to the skyline processing phase, thus reducing the overall cost of expensive dominance checks required in the skyline phase. We further present more aggressive pruning rules that result in the computation of *near-complete* skylines in significantly less time than the complete algorithm. A comprehensive performance evaluation of different algorithms is presented using datasets with different types of data distributions generated by a benchmark data generator.

Keywords: Skyline Queries, RDF Databases, Join.

1 Introduction

The amount of RDF data available on the Web is growing more rapidly with broadening adoption of Semantic Web tenets in industry, government and research communities. With datasets increasing in diversity and size, there have been more and more research efforts spent on supporting complex decision making over such data. An important class of querying paradigm for this purpose is preference queries, and in particular, skyline queries. Skyline queries are valuable for supporting multi-criteria decision making and have been extensively investigated in the context of relational databases [1][2][3][4][5][6][11][12] but in a very limited way for Semantic Web [8]. A *skyline query* over a data set S with D-dimension aims to return the subset of S which contains the points in S that are not *dominated* by any other data point. For two D-dimensional data points $p(u_1, u_2, u_3, \dots, u_d)$ and $q(v_1, v_2, v_3, \dots, v_d)$, point p is said to dominate point q if $p.u_i \succcurlyeq q.v_i$ for all $i \in [1,d]$ and in at least one dimension $p.u_j \succ q.v_j$ where $j \in [1,d]$, $\succcurlyeq$ denotes *better than* or *equal with*, $\succ$ denotes *better than*. For example, assume that a company wants to plan a sales promotion targeting the likeliest buyers (customers that are young with a low amount of debt). Consider three customers A (*age 20, debt $150*), B (*age 28, debt $200*) and C (*age 25, debt $100*). Customer A is clearly a better target than B because A is younger and has less debt. Therefore, we say that A *dominates* B. However, A does not *dominate* C since A is younger than C but has more debt than C. Therefore, the skyline result over the customer set $\{A, B, C\}$ is $\{A, C\}$.

The dominant cost in computing the skyline of a set of n D-dimension tuples lies in the number of comparisons that needs to be made to decide if a tuple is or isn't part of the skyline result. The reason is that for a tuple to be selected as being in the skyline, it would need to be compared against all other tuples to ensure that no other tuples dominate it. Further, each tuple pair comparison involves D comparisons comparing their values in all D dimensions. Consequently, many of existing techniques [1][2][3][4][5][6][11][12] for computing skylines on relational databases focus on reducing the number of tuple pair comparisons. This is achieved using indexing [4][5][6], or partitioning data into subsets [11] where some subsets would contain points that can quickly be determined to be in or pruned from the skyline result.

It is also possible to have skyline queries involving attributes across multiple relations that need to be joined, i.e. multi-relational skyline. This would be a very natural scenario in the case of RDF data since such data is often stored as vertically partitioning relations [7]. However, there are much fewer efforts [11][12][16] directed at evaluating skylines over multiple relations. A common strategy, which was also proposed in the context of preference queries on the Semantic Web [8], is to first join all necessary tables in a single table and then use a single table skyline algorithm to compute the skyline result, i.e. *join-first-skyline-later* (JFSL). A limitation of the JFSL approach is that the join phase only focuses on identifying joined tuples on which skyline computation can then be done. It does not exploit information about the joined tuples to identify tuples that are clearly not in the skyline result. Identifying such tuples would allow pruning them from advancing to the skyline phase and avoid the expensive dominance checks needed for skyline computation on those tuples. Alternative techniques to the JFSL approach [4][5][6][11][12][16] employ specialized indexing and storage schemes which are not typical in RDF data and require preprocessing or storage in multiple formats.

1.1 Contributions

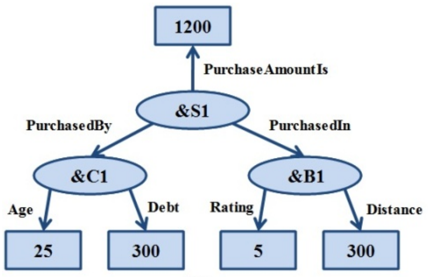
This paper proposes an approach for efficient processing of skyline queries over RDF data that is stored as vertically partitioned relations. Specifically, we propose

- The concept of a Header Point that maintains a concise summary of the already visited region of the data space for pruning incoming non-skyline tuples during join phase. This improves efficiency by reducing number of comparisons needed during later skyline processing phase.

- A complete algorithm and two near-complete algorithms based on an approach that interleaves the join processing with some skyline computation to minimize the number of tuples advancing into skyline computation phase. The near-complete algorithms compute about 82% of skyline results in about 20% of the time needed for the complete algorithm.
- A comprehensive performance evaluation of the algorithms using datasets with different types of data distributions generated by a benchmark data generator.

2 Background and Problem Statement

Assume that we have a company's data model represented in RDF containing statements about *Customers, Branches, Sales and relationships* such as *Age, Debt, PurchasedBy etc.* Figure 1 (a) shows a sample of such database using a graph representation.



(a)

```
SELECT *
WHERE { ?Sale       PurchasedBy   ?Customer
        ?Customer   Age           ?ageVal
        ?Customer   Debt          ?debtVal
        ?Sale       PurchasedIn   ?Branch
        ?Branch     Distance      ?distVal }
SKYLINE OF ?ageVal MIN, ?debtVal MIN,
           ?distVal MIN
```

(b)

Fig. 1. Example RDF Graph Model and Skyline Queries

Consider again the example of sales promotion targeting the young customers with less debt. Also assume that company would like to focus their campaigns on customers that live close to some branch. We express such a query using an extended SPARQL as shown in Figure 1 (b). The properties in front of MIN/MAX keywords are the skyline properties (dimensions) to be considered during the skyline computation. The MIN/MAX keyword specifies that we want the value in the corresponding property to be minimized/maximized. We now formalize the concept of a *skyline graph pattern* that builds on the formalization of SPARQL graph pattern queries.

An RDF *triple* is 3-tuple (s, p, o) where *s* is the *subject*, *p* is the *predicate* and *o* is the *object*. Let *I*, *L* and *B* be the pairwise disjoint infinite set of *IRIs*, *Blank nodes* and *Literals*. Also assume the existence of an infinite set *V* of variables disjoint from the above sets. A *triple pattern* is a query pattern such that *V* can appear in *subject*, *predicate* or *object*. A *graph pattern* is a combination of triple patterns by the binary operators *UNION*, *AND* and *OPT*. Given an RDF database graph, a solution to a graph pattern is a set of substitutions of the variables in the graph pattern that yields a subgraph of the database graph and the solution to the graph pattern is the set of all possible solutions.

Definition 1 (Skyline Graph Pattern). *A Skyline Graph Pattern is defined as a tuple (GP, SK) where GP is a graph pattern and SK is a set of skyline conditions* $\{sk_1, sk_2, \ldots, sk_m\}$*.* sk_i *is of the form* $fn_i(p_i)$ *where* fn_i *is either min() or max() function and* p_i *is a property in one of the literal triple patterns in GP. The Solution to (GP, SK) is an ordered subset* $RS = [S_i, S_j, \ldots, S_k] \subseteq S$ *where* S_i *denotes the solution to a basic graph pattern (an RDF graph with variables) and the following conditions hold: (i) each solution* $S_p \in RS$ *is not dominated by any solution in* S; *(ii) each solution* $S_q \in \{S - RS\}$ *is dominated by some solution in* S*.*

3 Evaluating the Skyline over the Join of Multiple-Relations

Vertically partitioned tables (VPT) are a common storage model for RDF data. A straightforward approach to compute the skyline over a set of vertically partitioned tables $VPT_{1,} VPT_{2,\ldots,} VPT_d$ is as follows: (i) join $VPT_{1,}VPT_{2,\ldots,}VPT_d$ into a complete single table *T* (the determination of dominance between two tuples cannot be made by looking at only a subset of the skyline properties); (ii) compute skyline over this single table *T* by using any single-table skyline algorithm, such as BNL (Block-Nested-Loop). We call this approach as "*Naive*" algorithm. "*Naive*" algorithm maintains only a subset of all already joined tuples (*candidate list*) against which each newly joined tuple is compared to determine its candidacy in the skyline result. However, this approach does not fully exploit the information about the joined tuples and requires too many comparisons to determine one tuple's candidacy in skyline result. Our approach based on the concept of a *Header Point* improves upon this by using information about already joined tuples to make determinations about (i) whether a newly joined tuple could possibly be a member of the skyline and (ii) whether there is a possibility of additional skyline tuples to be encountered in the future. The former allows for pruning a tuple from advancing to the skyline (SL) phase where it would incur additional cost of comparisons with several tuples in a skyline candidate list. The latter allows for early termination.

3.1 Header Point and Its Prunability

Our approach is based on splitting the join phase into iterations where information about earlier iterations is summarized and used to check skyline candidacy of tuples joined in later iterations. In each *join iteration*, we need to join each 2-tuple in each VPT to their corresponding matching 2-tuples in all of the other VPTs. Let J_i be the *table pointer* pointing to the subject value (sub_j) of the *j*th triple tr_j in VPT_i and a *join iteration* would be:

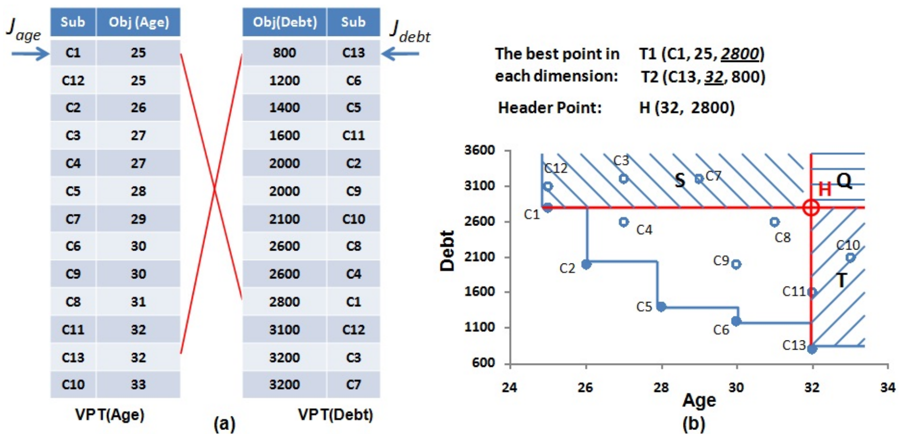
$$\bigcup_{\text{i=1 to d}} \text{join of } tr_j \text{ to matching tuples in } VPT_k(\, k \neq i)$$

In other words, at the end of a join iteration we would have computed *d* tuples and each tuple is based on the 2-tuple pointed to by table pointer in some dimension VPT.

A *Header Point* summarizes the region of data explored in earlier join iterations. It enables a newly joined tuple in the subsequent join iteration to be compared against this summary rather than multiple comparisons against the tuples in the skyline candidate list.

Definition 2 (Header Point). *Let* $\{t_1, t_2, \dots, t_d\}$ *be the set of tuples in the jth join iteration. A Header Point of the computation is a tuple of* $< f_{n_j}(\{t_i[1]\})$, $f_{n_j}(\{t_i[2]\}), \dots, f_{n_j}(\{t_i[d]\}) >$ *where* f_{n_j} *is either min() or max() function. We call the tuples that form the basis of the Header Point (i.e. the* t_i*s), Header Tuples.*

To illustrate the advantage of the header point concept, we will use a smaller version of our motivating example considering only a graph sub pattern with skyline properties, *Age* and *Debt*. We will assume that data is organized as VPT and indexed by Subject (SO) and by Object (OS). Using the OS index, the triples can be accessed in decreasing order of "goodness" when minimizing/maximizing skyline properties, i.e. in increasing/decreasing order of object values. Let us consider the earliest join iteration involving the first tuples of each relation. Figure 2 (a) shows the table pointers (J_{Age} and J_{Debt}) for the two relations and the two red lines show the matching tuples to be joined resulting in the tuples *T1* (*C1, 25, 2800*) and *T2* (*C13, 32, 800*) shown in Figure 2 (b). Since these tuples are formed from the best tuples in each dimension, they have the best values in their respective dimensions, therefore no other tuples dominate them and they are part of the skyline result. We can create a header point based on the worst values (here, the largest values) in each dimension (*Age* and *Debt*) across all the currently joined tuples resulting in a tuple *H* (*32, 2800*). *T1* and *T2* are called Header Tuples.



VPT(Age)

Sub	Obj (Age)
C1	25
C12	25
C2	26
C3	27
C4	27
C5	28
C7	29
C6	30
C9	30
C8	31
C11	32
C13	32
C10	33

VPT(Debt)

Obj(Debt)	Sub
800	C13
1200	C6
1400	C5
1600	C11
2000	C2
2000	C9
2100	C10
2600	C8
2600	C4
2800	C1
3100	C12
3200	C3
3200	C7

Fig. 2. Formation of Initial Header Point and its Prunability

Our goal with the header point is to use it to determine whether newly joined tuples in subsequent iterations should be considered as candidate skyline tuples or be pruned from advancing to the expensive skyline processing phase. For example, assume that in the next join iteration we compute the next set of joined tuples by advancing the table pointer to the next tuple in each VPT. Then, one of the resulting joined tuples is (*C12, 25, 3100*). For the rest of the paper, we will use *C12* to denote the tuple (*C12, 25, 3100*). However, the current candidate skyline list { (*C1, 25, 2800*), (*C13, 32, 800*) } contains a tuple (*C1, 25, 2800*) that clearly dominates *C12*, therefore *C12* should not be considered as a candidate skyline tuple. Further, we should be able to use the information from our header point to make this determination rather than compare every newly joined tuple with tuples in the candidate skyline list. We observe the following relationship between newly joined tuples and the header point that can be exploited: *If a newly joined tuple is worse than the header point in <u>at least one</u> dimension, then that new tuple cannot be a skyline tuple and can be pruned.* Applying this rule to our previous example, we will find that the tuple *C12* is worse than our header point *H* in the *Debt* dimension. Therefore, it can be pruned. In contrast, the other joined tuple (*C6, 30, 1200*) is not worse than *H* in at least one dimension (in fact, it is better in two dimensions) and is not pruned. Figure 2 (b) shows the relationships between these points in a 2-dimensional plane: any subsequent joined tuple located in the regions of *S*, *Q* and *T* can be pruned by header point *H*. We now make a more precise statement about the relationship between a header point and tuples created after the creation of the header point.

Lemma 1. *Given a D-dimensional header point* $H = < h_1, h_2, \ldots, h_D >$, *any "subsequent " (i.e. a point constructed after the current header point) D-tuple whose values are worse than H in at least D - 1 dimensions, are* ***not*** *skyline points.*
Proof. Let $P = (p_1, p_2, \ldots, p_D)$ be a new tuple that is worse than the current header point $H = (h_1, h_2, \ldots, h_D)$ in at least $D - 1$ dimensions but is a candidate skyline point. Let H be the header point that was just formed during the construction of the most recently computed set $B = \{ B_1 = (nbv_1, q_2, \ldots, q_D)$, $B_2 = (t_1, nbv_2, \ldots, t_D)$, … , $B_d = (s_1, s_2, \ldots, nbv_D)$ } of candidate skyline points (header tuples), where nbv_i denotes the next best value in VPT_i. Recall that *B* consists of the last *D* tuples that resulted from the join between best tuples in each dimension and a matching tuple in each of the other dimensions.

Assume that the dimensions in which *P* has worse values than the header point *H* are dimensions 2 to *D*. Then, *H* "partially dominates" *P* in dimensions 2 to *D*. Further, since the header point is formed from the combination of the worst values in each dimension across the current header tuples, *P* is also partially dominated by the current header tuples which are also current candidate skyline tuples. Therefore, the only way for *P* to remain in the skyline is that no candidate skyline tuples dominate it in the only remaining dimension, dimension 1. However, the header point tuple $B_1 = (nbv_1, q_2, \ldots, q_d)$ which is currently a candidate skyline tuple has a dimension-1 value - nbv_1 that is better than p_1. This is because the values are sorted and visited in decreasing order of "goodness" and the tuple $B_1$was constructed before *P*, so the value nbv_1 must be better than p_1. This means that B_1 is better than *P* in all dimensions and therefore dominates *P*. Therefore, *P* cannot be a skyline tuple which contradicts our assumption.

3.2 RDFSkyJoinWithFullHeader (RSJFH)

We now propose an algorithm *RDFSkyJoinWithFullHeader (RSJFH)* for computing skylines using the *Header Point* concept. In the *RSJFH* algorithm, a join iteration proceeds as follows: *create a joined tuple based on the tuple pointed by the table pointer for dimension i. If a resulting joined tuple is pruned, then advancing table i's pointer until it points to a tuple whose joined tuple is not pruned by the current header point. This process is repeated for each dimension to create a set of d header* *tuples*. Since the header point is formed using the worst values in each dimension among the joined tuples, it may represent very loose boundary conditions which will significantly reduce its pruning power. This occurs when these worst values are among the worst overall in the relations. In our example, this occurs in the first join iteration with the construction of the initial header point *H*(*32, 2800*). To strengthen the pruning power of the header point, we can update its values as we encounter new tuples with better values i.e. the next set of *D* tuples whose set of worse values are better than the worse values in the current header point. These tuples will become the next iteration's header tuples.

Figure 3 (a) shows the second join iteration where new header tuples are used to update the Header Point. The next tuple in *Age* VPT is (*C12, 25*) and its joined tuple is (*C12, 25, 3100*). Compared with *H*, *C12* can be pruned since it is worse than *H* in at least *D-1* dimensions (*D* is 2). *RSJFH* advances the table pointer to the next tuple in the *Age* table, (*C2, 26*) whose joined tuple is (*C2, 26, 2000*). Compared with *H*, *C2* is not pruned and this tuple is adopted as a header tuple. Then, *RSJFH* moves to the next VPT *Debt* where the next tuple is (*C6, 1200*) and its joined tuple is (*C6, 30, 1200*). Compared with *H*, *C6* is not pruned. Now, there is one header tuple from each VPT and the header point can be updated to *H'* (*30, 2000*). Similarly, in the third join iteration (Figure 3 (b)), *RSJFH* checks the subsequent joined tuples in tables *Age* and *Debt* and finds (*C5, 28, 1400*) and (*C5, 28, 1400*) are the next header tuples in tables *Age* and *Debt* respectively. Then, the header point is updated to *H''* (*28, 1400*) based on these two header tuples.

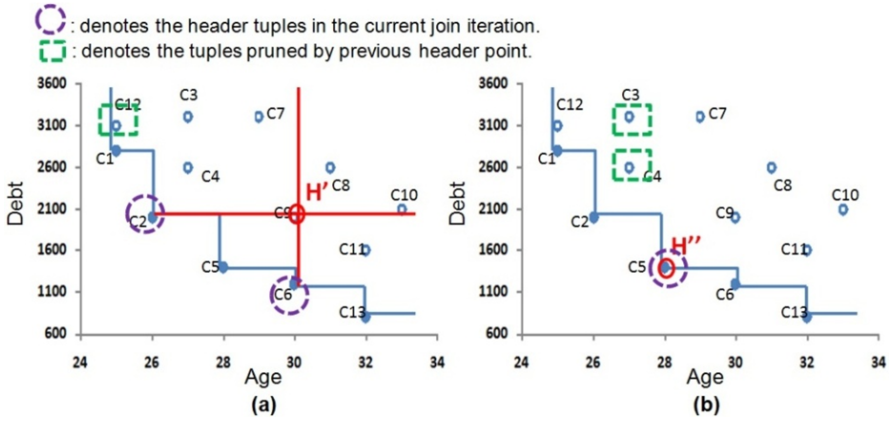


Fig. 3. Updating Header Points

3.2.1 Termination Phase and Post Processing

The *RSJFH* terminates the search for additional candidate skyline tuples when either (i) the header point is not updated or (ii) either one of the table pointers J_i advances down to the end of VPT_i.

Lemma 2. *Let* $H = < h_1, h_2, \dots, h_D >$ *be the last D-dimensional header point computed during the last join iteration i.e. the joining process that resulted in the computation of the last D candidate skyline tuples. If during the current join iteration* *the header point is not updated or either one of the table pointers J_i advances down to the end of VPT_i, then the search for additional candidate skyline tuples can be terminated losslessly.*

Proof. Recall that during a join iteration, we pick the next best tuples that are not pruned in each dimension and join with other dimensions to form the next D candidate skyline tuples. So each resulting tuple should contain the next best value in some dimension, i.e. $B = \{B_1 = (nbv_1,\ q_2, \ldots, q_D), B_2 = (t_1, nbv_2, \ldots, t_D), \ldots, B_d = (s_1,\ s_2, \ldots, nbv_D)\}$. If after computing the set B, the header point is not updated, it implies that each nbv_i is worse than the corresponding h_i in H. It is clear that all these tuples can be pruned because their best dimension values are worse than our current worst seen values in our header tuple, resulting in that the other dimensions are clearly also worse. Thus, the header tuple dominates all of them. Further, since the tuples in each dimension are ordered in the decreasing order of "goodness", the next set of best values cannot be better than those currently considered. Therefore, no better tuples can be found with further scanning of the tables. When either one of the table pointers J_i advances down to the end of VPT_i then all the values for that dimension have been explored. Therefore, no new D-tuples can be formed and the search for additional candidate skyline tuples can be terminated losslessly.

Algorithm 1. RDFSkyJoinWithFullHeader (RSJFH)

```
INPUT: n VPTs which are sorted according to object value, VPTList.
OUTPUT: A set of skyline points.
1.Initialization // to get first header point
2.    read first tuple in each VPT and hash join to get n complete tuple Ti.
3.    take the worst value in each dimension among Ti (i=1, 2,.., n) to compute Header Point H
4.While H is updated or pointers $J_t$ is not pointing to the end of $VPT_t$ do
5.        for each VPT t ∈ VPTList
6.                read one tuple and hash join to get complete tuple T and compare T with H
7.                if T is prunable by H
8.                        T is pruned
9.                else
10.                       T is a Header Tuple for updating H and is inserted into Candidate List C
11.       end for
12.       update Header Point H by Header Tuples
13. end while
14.BNLCalculate(C). // use BNL skyline algorithm to compute skyline results
```

Discussion. Intuitively, the header point summarizes neighborhoods that have been explored and guides the pruning of the tuples in neighborhood around it during each iteration. However, since *RSJFH* uses the worse points in each dimension, and prunes tuples that are really worse (worse in *d-1* dimensions) than the header point, it only allows *conservative* pruning decisions. Therefore, some non-skyline points can still be inserted into the candidate skyline list. For example, assume that *RSJFH* has just processed the following tuples into the candidate skyline list {(*25, 4000*), (*28, 3500*), (*30, 3000*)} and computed the header point (*30, 4000*). Then, an incoming tuple (*29, 3750*) would be advanced to the next stage because it is better than the header point in both dimensions. However, it is unnecessary to advance the tuple (*29, 3750*) into the next stage of the computation because it would eventually be dropped from candidate skyline list since the tuple (*28, 3500*) dominates this tuple.

4 Near-Complete Algorithms

We can try to improve the header point to allow for more aggressive pruning. We may however risk pruning out correct skyline tuples. In the following section, we propose a strategy that strikes a balance between two objectives: increasing the aggressiveness of pruning power of the header point and minimizing the risk of pruning correct skyline tuples. We posit that in the context of the Web, partial results can be tolerated particularly if the result set is "near-complete" and can be generated much faster than the "complete" set. The approach we take is based on performing *partial* updates to the header point after each join iteration rather than updating all the dimensions.

4.1 RDFSkyJoinWithPartialHeader (RSJPH)

Definition 3 (Partial Header Point Update). *Let H be the header point generated in the previous join iteration and* $t_1, t_2, \ldots, t_n$ *be the tuples in the current join iteration. A partial update to the header point means that for the ith dimension of header point, the value is updated only if the worst value of* $t_1, t_2, \ldots, t_n$ *in the ith dimension is better than the ith dimensional value in H.*

This implies that if all values in the *i*th dimension are better than the *i*th dimensional value of the header point, then the *i*th dimension of the header point is updated with worst value as before; otherwise, the *i*th dimension is not updated. Thus, the header point is aggressively updated by the improving (or advancing) dimension values of the joined tuples in the current join iteration.

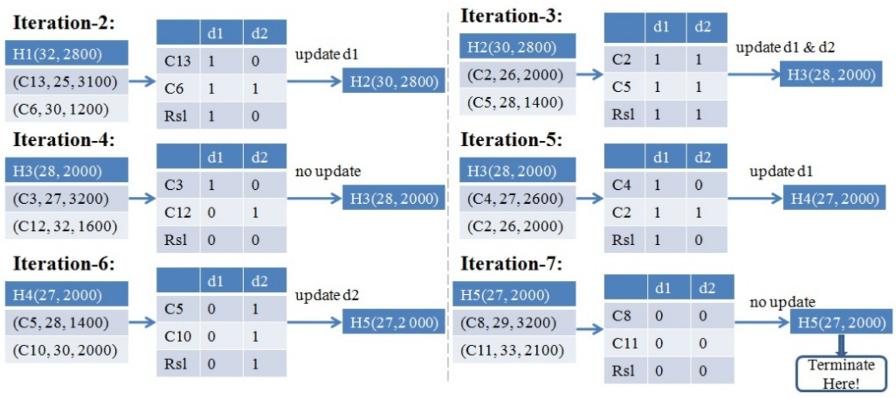


Fig. 4. Partially Update Header Points by Bitmap Computation and its Termination

We now propose an algorithm *RDFSkyJoinWithPartialHeader (RSJPH)* based on the partial update. To efficiently perform a "partial update of header point", *RSJPH* uses *bitmap* representation for *bitwise* operations. Consider the skyline graph sub pattern of *Customer* again. Figure 4 shows the partial update method applied after the first join iteration. In the second iteration, there are two newly joined tuples (*C13, 25, 3100*) and (*C6, 30, 1200*) and a header point *H1*(*32, 2800*) generated in the previous iteration. Since both the dimension-1 values of *C13* and *C6* are better than that of *H1* and only the dimension-2 value of *C6* is better than that of *H1*, header point is partially updated by the worse dimension-1 values of *C13* and *C6*, resulting in *H2*(*30, 2800*). This partial update method is implemented using *Bitmap* computation (bit "1" denotes the dimension that has better value, bit "0" denotes a contrary case and a resulting bit array indicates that in which dimension(s) the header point needs to be updated). Iterations 3 to 6 perform the same way to partially update the header points.

Termination. *RSJPH* revises the termination condition based on the partial update (Line 2 in Algorithm 2). *RSJPH* terminates when there is no update for header point and all the dimensional values in the newly joined tuples are worse than that of current header point. Iteration 7 in Figure 4 shows how *RSJPH* terminates.

Algorithm 2. RDFSkyJoinWithPartialHeader(RSJPH)

INPUT: *n* ***VPT*** which are sorted according to object value, ***VPTList***.
OUTPUT: A set of skyline points.
Define Variables: in n newly-joined tuples $t_1, t_2, \dots, t_n$, let dimension-*1* value of t_1 be V_1, dimension-*2* value of t_2 be V_2, …, dimension-*n* value of t_n be V_n.

```
1.Initialization // to get first header point $H$
2.While ($H$ is updated) && ($<V_1, V_2, \dots, V_n>$ is better than $H$) do
3.    for each VPT $t \in VPTListl$ do
4.        read one tuple and hash join to get complete tuple $T$ and compare $T$ with $H$
5.        use the bitmap representation to record $T$'s better value and worse value
6.        if $T$ is prunable by $H$ then
7.            $T$ is pruned
8.        else
9.            $T$ is inserted into Candidate List $C$
10.   end for
11.   read the bitmap as bit-slice and calculate bit-slice value by bitwise AND operation
12.   for each bit-slice do // partially update $H$
13.       if bit-slice value is 1 then
14.           update the corresponding header point value
15.       else
16.           no update
17.   end for
18.end while
19.BNLCalculate($C$).
```

Discussion. In *RSJFH*, we always update the dimensions using the worse values from the header tuples in that iteration regardless of whether those values are better or worse than the current header point values. Essentially, a header point summarizes only the iterations preceding it. In *RSJPH*, a header point may hold "good" pruning values from earlier iterations and only update those dimensions whose values are improving or advancing. This in a sense broadens the neighborhood that it represents and also allows certain regions in the space to be considered for longer periods than the previous approach. Consequently, we have a header point with more aggressive pruning power. For example, assume that after our previous header point (*30, 4000*), the next iteration would have worse values as (*25, 4500*). However, given our partial update technique, only the first dimension would be updated to 25, resulting in the header point (*25, 4000*). This header point would prune more tuples than the regular header point (*25, 4500*). However, some of the pruned tuples that fall into the gap between (*25, 4500*) and (*25, 4000*) may be skyline tuples, such as (*25, 4250*).

Proposition 1. *Let* $H_1^f, H_2^f, \dots, H_m^f$ *be the header points generated in RSJFH and A be the set of pruned tuples by all the header points. Similarly, let* $H_1^p, H_2^p, \dots, H_n^p$ *be the header points generated in RSJPH and B be the set of pruned tuples by all the header points.* H_i^p *is stronger than* H_i^f *, which means* H_i^p *may wrongly prune some skyline points* WP_i *falling into the interval* $<H_i^f, H_i^p>$*. Then,* $A \subset B$ *and the difference set* $B - A = \bigcup_{i=1}^{n} WP_i$.

To avoid this we can relax the pruning check. Rather than generalizing the checking based on number of dimensions, we do checking based on which dimensions were updated and which were not updated.

4.2 Relaxing Prunability of Partially Updated Header Point

Given a header point, if the *i*th dimension is updated in the last iteration, we regard it as an "*updated dimension*"; otherwise, we regard it as "*non-updated dimension*". Given a tuple *p*, if *p* has *n* dimensions whose values are better than that of the header point *h*, we say that *p* has *n better dimensions*. From **Lemma 1**, we can infer that a tuple *p* needs to have at least two *better dimensions* to survive the pruning check. Assume that we have: (1) $H_2(d_1', d_2')$ is a header point partially updated from a full updated header point $H_1(d_1, d_2)$, where $d_1' \succ d_1$ and $d_2' = d_2$, where denotes *better*; Thus, d_1' is the "updated" dimension of H_2 and d_2' is the "non-updated" dimension of H_2; (2) a tuple $p(p_1, p_2)$, where $d_1' \succ p_1$, $p_1 \succ d_1$ and $p_2 \succ d_2'$. Compared to H_2, *p* can be pruned because *p* has only one *better dimension*. However, when compared to H_1, *p* will survive the pruning check since *p* has two *better dimensions* ($p_1 \succ d_1$ and $p_2 \succ d_2$). Since the partial update approach makes the "updated" dimensions of H_2 too good, the tuples that may survive given the fully updated header point H_1, such as *p*, are mistakenly pruned. To alleviate this situation, we relax the pruning condition with the following crosscheck constraint.

Crosscheck. *If an incoming tuple has some dimensional values better than "non-updated" dimension and some dimensional values worse than "updated" dimension, we add this tuple into candidate list. To implement this algorithm, we basically add this additional condition check between Lines 6 and 7 in Algorithm 2. The resulting algorithm is called RDFSkyJoinWithPartialHeader+ (RSJPH+).*

Proposition 2. *Let* H_i^p *be a header point in RSJPH with the "updated" dimension* d_g *that has been updated in iteration i-1 and the "non-updated" dimension* d_b *that has not been updated in iteration i-1. Let p be a new tuple that has failed the pruning* *check by* H_i^p *. If p survives the crosscheck condition, i.e.,* $H_i^p.d_g \ngtr p.d_g$ *but* $p.d_b \ngtr H_i^p.d_b$*, p is saved and added into candidate list. We regard the saved tuples by crosscheck in iteration i as the set* CC_i*.* $CC_i \subseteq WP_i$ *. Assume C denotes the set of pruned tuples in RSJPH+. Then,* $C = B - \bigcup_{i=1}^{n} CC_i$ *and* $A \subset C \subseteq B$*.*

5 Experimental Evaluation

Experimental Setup and Datasets. In this section, we present an experimental evaluation of the three algorithms presented in above sections in terms of scalability, dimensionality, average completeness coverage and prunability. We use the synthetic datasets with independent, correlated and anti-correlated data distributions generated by a benchmark data generator [1]. Independent data points follow the uniform distribution. Correlated data points are not only good in one dimension but also good in other dimensions. Anti-correlated data points are good in one dimension but bad in one or all of the other dimensions. All the data generated by the data generator is converted into RDF format using JENA API and is stored as VPT using BerkeleyDB. All the algorithms are implemented in Java and the experiments are executed on a Linux machine of 2.6.18 kernel with 2.33GHz Intel Xeon and 16GB memory. The detailed experimental results can be found at sites.google.com/site/chenlingshome.

Scalability. Figure 5 (A), (B) and (C) show the scalability evaluation of *RSJFH*, *RSJPH*, *RSJPH+* and *Naïve* for independent, correlated and anti-correlated datasets (1 million to 4 million triples). In all data distributions, *RSJPH* and *RSJPH+* are superior to *RSJFH* and *Naïve*. The difference in execution time between *RSJPH*, *RSJPH+* and *RSJFH* comes from the fact that partial update method makes the header point stronger (i.e. the header point has better value in each dimension and could dominate more non-skyline tuples resulting in stronger prunability) earlier, which terminates the algorithm earlier. For independent data (Figure 5 (A)), *RSJPH* and *RSJPH+* use only about 20% of the execution time needed in *RSJFH* and *Naïve*. The execution time of *RSJFH* increases quickly in the independent dataset with size 4M of triples. The reason for this increase is that the conservativeness of the full header point update technique leads to limited effectiveness in prunability. This results in an increased size for the candidate skyline set and consequently total number of comparisons with Header Point. *RSJPH+* relaxes the check condition in *RSJPH* and so allows more tuples to be inserted into the candidate list explaining the slight increase in the execution time in Figure 5(A). Figure 5 (B) shows that *RSJFH*, *RSJPH* and *RSJPH+* perform better in correlated datasets than in independent datasets. In the correlated data distribution, the header points tend to become stronger earlier than in the case of independent datasets especially when the data is accessed in the decreasing order of "goodness". The reason for this is that the early join iterations produce tuples that are made of the best values in each dimension. Stronger header points make the algorithm terminate earlier and reduce the number of tuples joined and checked against the header point and the size of candidate skyline set. Figure 5 (C) shows particularly bad performance for the anti-correlated datasets which often have the best value in one dimension but the worst value in one or all of the other dimensions. This leads to very weak header points because header points are constructed from worst values of joined tuples. *RSJFH* have to explore almost the entire search space, resulting in the poor performance shown in Figure 5 (C). Although *RSJPH* seems to outperform the other algorithms, this advantage is attributed to the fact that it computes only 32% of complete skyline result set.

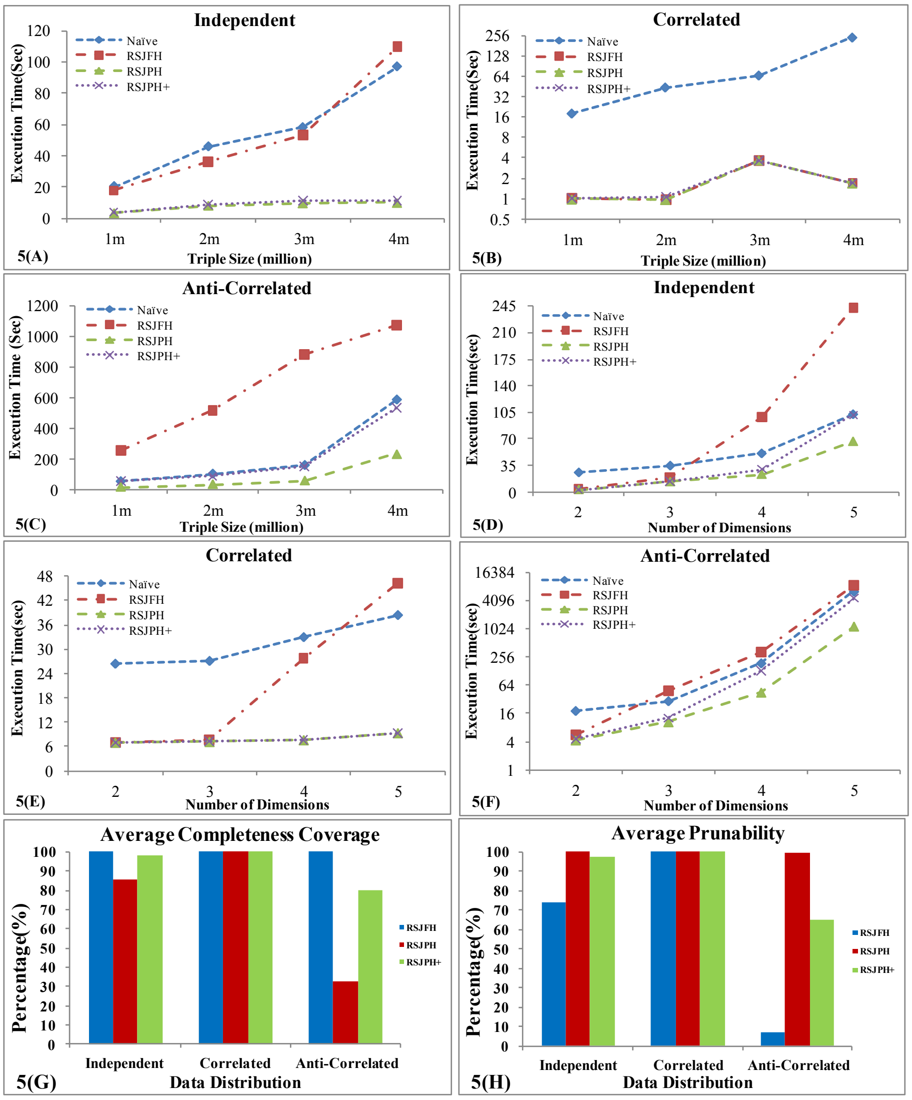


Fig. 5. Experimental Evaluation

Dimensionality. Figure 5 (D), (E) and (F) show the effect of increasing the dimensionality (2-5) on the performance of the four algorithms for different data distributions. As in previous experiments, *RSJPH* and *RSJPH+* consistently outperform *RSJFH* and *Naïve*. The execution time of *RSJFH* starts to increase with number of dimensions greater than 3. The reason is that the conservative way of updating header point makes the header points greatly reduce the pruning power in high dimensional data and the extra comparisons almost double the total execution time. For *RSJPH+*, with the increase in number of dimensions, the size of the saved tuples by the crosscheck condition increases, therefore, the size of candidate skyline set increases and the execution time increases as well.

Average Completeness Coverage and Average Prunability. Figure 5 (G) and (H) show the average completeness coverage (*ACC*) and average prunability (*AP*) of *RSJFH*, *RSJPH* and *RSJPH+*. *ACC* and *AP* are the averages for completeness coverage and the number of pruned triples across all the experiments shown in Figure 5 (A) to (F) respectively. The pruned triples include the ones pruned by header points as well as the ones pruned by early termination. Figure 5 (E) shows that *RSJFH* has 100% of *ACC* in all data distributions. For the correlated datasets, the data distribution is advantageous in forming a strong header point without harming the completeness of skyline results when the data is sorted from "best" to "worst". Thus, *RSJFH*, *RSJPH* and *RSJPH+* have 100% of *ACC* and 99% of *AP* in correlated datasets. For independent datasets, *RSJPH* aggressively updates the header points to increase *AP* with the cost of decreasing *ACC*. *RSJPH+* improves the *ACC* by using crosscheck while only sacrificing 2.7% of the *AP* compared with *RSJPH*. For anti-correlated datasets, the data distribution makes all the algorithms perform poorly. Although *RSJFH* achieves 100% of *ACC*, the *AP* decreases to 7%. *RSJPH* still maintains 99% for *AP* but its *ACC* is only 32%.

RSJPH+ achieves a good tradeoff between completeness coverage and prunability. *RSJPH+* computes about 80% of skyline results when it scans about the first 35% of sorted datasets.

6 Related Work

In recent years, much effort has been spent on evaluating skyline over single relation. [1] first introduced the skyline operator and proposed BNL and D&C and an algorithm using B-trees that adopted the first step of Fagin's A_0. [2][3][9] proposed algorithms that could terminate earlier based on sorting functions. [4][5][6] proposed index algorithms that could progressively report results. Since these approaches focus on single relation, they consider skyline computation independent from join phase, which renders the query execution to be blocking.

Some techniques have been proposed for skyline-join over multiple relations. [11] proposed a partitioning method that classified tuples into three types: general, local and non-local skyline tuples. The first two types are joined to generate a subset of the final results. However, this approach isn't suitable for single dimension tables like VPT [7] in RDF databases because each VPT can only be divided into general and local skyline tuples, neither of which can be pruned, requiring a complete join of all relevant tables. [16] proposed a framework SkyDB to partition the skyline-join process into macro and micro level. Macro level generates abstractions while micro level populates regions that are not pruned by macro level. Since RDF databases only involve single dimension tables, SkyDB is not suitable for RDF databases.

In addition, there are some techniques proposed for skyline computation for Semantic Web data and services. [8] focused on extending SPARQL with support of expression of preference queries but it does not address the optimization of query processing. [13] formulated the problem of semantic web services selection using the notion of skyline query and proposed a solution for efficiently identifying the best match between requesters and providers. [14] computed the skyline QoS-based web service and [15] proposed several algorithms to retrieve the top-k dominating advertised web services. [17] presented methods for automatically relaxing over-constrained queries based on domain knowledge and user preferences.

7 Conclusion and Future Work

In this paper, we have addressed the problem of skyline queries over RDF databases. We presented the concept of *Header Point* and *Early Termination* to prune non-skyline tuples. We have proposed a complete algorithm *RSJFH* that utilized the prunability of *Header Point* and *Early Termination.* Then, we proposed two near-complete algorithms, *RSJPH* and *RSJPH+*, for achieving the tradeoffs between quick response time and completeness of skyline queries over RDF databases. In future, we will integrate cost-based techniques for further optimization. We will also address the issue of incomplete skyline computation over RDF databases.

Acknowledgement. The work presented in this paper is partially funded by NSF grant IIS- 0915865. Thanks to Mridu B. Narang for the draft comments and to Jigisha Dhawan, Vikas V. Deshpande, Amrita Paul and Gurleen Kaur for the discussions.

References

[1] Borzsonyi, S., Kossmann, D., Stocker, K., Passau, U.: The Skyline Operator. In: ICDE (2001)

[2] Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: ICDE (2003)

[3] Bartolini, I., Ciaccia, P., Patella, M.: SaLSa: computing the skyline without scanning the whole sky. In: CIKM, Arlington, Virginia, USA, pp. 405–414 (2006)

[4] Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient Progressive Skyline Computation. In: VLDB, San Francisco, CA, USA, pp. 301–310 (2001)

[5] Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: VLDB, HK, China, pp. 275–286 (2002)

[6] Papadias, D., Fu, G., Morgan Chase, J.P., Seeger, B.: Progressive Skyline Computation in Database Systems. ACM Trans. Database Syst (2005)

[7] Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable semantic web data management using vertical partitioning. In: VLDB, Vienna, Austria, pp. 411–422 (2007)

[8] Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 612–624. Springer, Heidelberg (2006)

[9] Godfrey, P., Shipley, R., Gryz, J.: Maximal Vector Computation in Large Data Sets. In: VLDB, Norway (2005)

[10] Raghavan, V., Rundensteiner, E.A.: Progressive Result Generation for Multi-Criteria Decision Support Queries. In: ICDE (2010)

[11] Jin, W., Ester, M., Hu, Z., Han, J.: The Multi-Relational Skyline Operator. In: ICDE (2007)

[12] Sun, D., Wu, S., Li,J.,Tung, A.K.H.: Skyline-join in Distributed Databases. In: ICDE Workshops, pp. 176–181 (2008)

[13] Skoutas, D., Sacharidis, D., Simitsis, A., Sellis, T.: Serving the Sky: Discovering and Selecting Semantic Web Services through Dynamic Skyline Queries. In: ICSC, USA (2008)

[14] Alrifai, M., Skoutas, D., Risse, T.: Selecting Skyline Services for QoS-based Web Service Composition. In: WWW, Raleigh, NC, USA (2010)

[15] Skoutas, D., Sacharidis, D., Simitsis, A., Kantere, V., Sellis, T.: Top-k Dominant Web Services Under Multi-Criteria Matching. In: EDBT, Russia, pp. 898–909 (2009)

[16] Raghavan, V., Rundensteiner, E.: SkyDB: Skyline Aware Query Evaluation Framework. In: IDAR (2009)

[17] Dolog, P., Stuckenschmidt, H., Wache, H., Diederich, J.: Relaxing RDF Queries based on User and Domain Preferences. JIIS 33(3) (2009)

The Design and Implementation of Minimal* RDFS Backward Reasoning in 4store

Manuel Salvadores[1], Gianluca Correndo[1], Steve Harris[2], Nick Gibbins[1], and Nigel Shadbolt[1]

[1] Electronics and Computer Science, University of Southampton, Southampton, UK
{ms8,gc3,nmg,nrs}@ecs.soton.ac.uk
[2] Garlik Ltd, UK
steve.harris@garlik.com

Abstract. This paper describes the design and implementation of *Minimal* RDFS semantics based on a backward chaining approach and implemented on a clustered RDF triple store. The system presented, called *4sr*, uses 4store as base infrastructure. In order to achieve a highly scalable system we implemented the reasoning at the lowest level of the quad store, the *bind* operation. The *bind* operation runs concurrently in all the data slices allowing the reasoning to be processed in parallel among the cluster. Throughout this paper we provide detailed descriptions of the architecture, reasoning algorithms, and a scalability evaluation with the LUBM benchmark. *4sr* is a stable tool available under a GNU GPL3 license and can be freely used and extended by the community[1].

Keywords: Triple Store, Scalability, Reasoning, RDFS, SPARQL, 4store.

1 Introduction

RDF stores - or triple stores - implement some features that make them very attractive for certain type of applications. Data is not bound to a schema and it can be asserted directly from RDF sources (e.g. RDF/XML or Turtle files) due to their native support of Semantic Web data standards. But the most attractive characteristic is the possibility of implementing an entailment regime. Having entailment regimes in a triple store allows us to infer new facts, exploiting the semantics of properties and the information asserted in the knowledge base. To agree on common semantics, some standards have arisen for providing different levels of complexity encoded in a set of inference rules, from RDF and RDFS to OWL and RIF, each of them applicable to different scenarios.

Traditionally, reasoning can be implemented via forward chaining (FC henceforth), backward chaining (or BC), or hybrid algorithms (a mixture of the two).

* Minimal RDFS refers to the RDFS fragment published in [8].

[1] Preliminary results were presented at the Web-KR^3 Workshop [10] and demoed at ISWC 2010 [9].

FC algorithms tend to apply a set of inference rules to expand the data set before or during the data assertion phase; with this approach, the database will contain all the facts that are needed when a query is issued. On the other hand, BC algorithms are goal directed and thus the system fires rules and/or axioms at runtime to find the solutions. Hybrid algorithms use a combinations of forward and backward chaining. The pros and cons of these approaches are well known to the AI and database community. FC approaches force the system to retract entailments when there is an update or insert, making data transactions very expensive. Complete materialisation of a knowledge base could lead to an explosion of data not manageable by current triple store technology. Backward chaining performs better for data transactions and the size of the KB is smaller, but queries tend to have worse performance.

Reasoners are also classified by their level of completeness. A reasoner can claim to be complete over an entailment regime $\mathcal{R}$ *if and only if*: (a) it is able to detect all entailments between any two expressions; and (b) it is able to draw all valid inferences; according to $\mathcal{R}$ semantics. For some types of applications a complete reasoner might be required but one should assume that higher completeness tends to degrade query response time. There is, therefore, a clear compromise: performance versus completeness and the old AI debate about speed and scalability versus expressiveness. In our specific case, *4sr* excludes a subset of RDFS semantics rarely used by Semantic Web applications and implements the semantics from the *Minimal* RDFS fragment [8]. *4sr* entailment is complete considering *Minimal* RDFS semantics but incomplete for the full normative RDFS semantics [5]. In that sense, our main contribution is a system that proves that *Minimal* RDFS semantics can scale if implemented in a clustered triple store. In comparison to our previous research, this paper formalizes *4sr* against *Minimal* RDFS, and also describes the components to be synchronized among the cluster and benchmarks the bind operation to test its scalability.

The remainder of the paper is as follows: Section 2 describes the related research in the area and introduces basic *4store* concepts and *Minimal* RDFS. Section 3 introduces and formalizes *4sr*'s distributed model. Section 4 explains the design and implementation of *4sr* explaining the modifications undertaken in *4store*. Section 5 studies the scalability of the new *bind* operation by benchmarking it under different conditions, and finally Section 6 analyses the results achieved by this work.

2 Related Work

4sr is a distributed backward chained reasoner for *Minimal* RDF/RDFS, and to our knowledge is the first system with such characteristics. However a number of related pieces of research informed our work.

Some current tools implement monolithic solutions using FC, BC or hybrid approaches. They use different types of back-ends as triple storage such as RDBMS, in memory, XML or native storage. Examples of these tools are Jena [2], Pellet [11] and Sesame [1]. These tools can perform RDFS reasoning with datasets containing up to few million triples. But, even though they have played a key role in helping Semantic Web technologies to get adopted, their scalability and performance is still a major issue.

BigOWLIM[2] is one of the few enterprise tools that claims to perform OWL reasoning over billions of triples. It can run FC reasoning against the LUBM (90K,0) [3] , which comprises around 12 billion triples. They materialize the KBs after asserting the data which means that BigOWLIM has to retract the materialization if the data is updated. There is no information on how this tool behaves in this scenario, even though they claim their inferencing system is retractable.

In the context of distributed techniques, [15] performs FC parallel reasoning to expand the RDFS closure over hundreds of millions of triples, and it uses a C/MPI platform tested on 128 core infrastructure with the LUBM 10k dataset. [13] pursues a similar goal and using MapReduce computes the RDFS closure over 865M triples in less than two hours. A continuation of this work has been presented in [12] providing a parallel solution to compute the OWL Horst regime. This solution, built on top of Hadoop, is deployed on a cluster of 64 machine and has been tested against a synthetic data set containing 100 billion triples and a 1.5 billion triples of real data from the LDSR and UniProt datasets.

[7] presented a novel method based on the fact that Semantic Web data present very skewed distributions among terms. Based on this evidence, the authors present a FC algorithm that works on top of data flows in a p2p infrastructure. This approach reported a materialization of RDFS for 200 million triples in 7.2 minutes on a cluster of 64 nodes.

Obviously, in the last 2-3 years there has been a significant advance on materialization of closures for both RDFS and OWL languages. However very little work has been presented on how to query such vast amounts of data and how to connect those solutions with SPARQL engines. Furthermore, these types of solutions are suitable for static datasets where updates and/or deletes are sparse or non-existent. Applying this mechanism to dynamic datasets with more frequent updates and deletes whose axioms need to be recomputed will lead to processing bottlenecks.

To avoid these bottlenecks, progress on backward chained reasoning is required. To date, there has been little progress on distributed backward chained reasoning for triple stores. [6] presented an implementation on top of DHTs using p2p techniques. So far, such solutions have not provided the community with tools, and recent investigations have concluded that due to load balancing issues they cannot scale [7].

With the SPARQL/Update specification to be ratified soon, we expect more triple/quad stores to implement and support transactions, which makes BC reasoning necessary at this juncture.

[2] http://www.ontotext.com/owlim/big/index.html accessed 21/06/2010

2.1 Minimal RDFS Reasoning

RDFS extends RDF with a schema vocabulary, a regime that contains semantics to describe light-weight ontologies. RDFS focusses mostly on expressing class, property and data type relationships and its interpretations can potentially generate inconsistencies. For instance, by using `rdfs:Literal` as `rdfs:domain` for a predicate `P`, any statement (`S`,`P`,`O`) with `P` as predicate would entail (`S a rdfs:Literal`) which is clearly inconsistent since RDF doesn't allow literals to be subjects (see section 4.3 [5]). Another issue with RDFS interpretaions is decidability. There is a specific case when using container memberships (`rdfs:ContainerMembershipProperty`) that can cause an RDFS closure to be infinite [14]. Other similar cases like these appear when constructing ontologies with different combinations of the RDF reification vocabulary, `rdf:XMLLiteral`, disjoint XSD datatypes, etc. A complete RDFS reasoner must examine the existence of such paradoxes and generate errors when models hold inconsistencies. Consistency checking is computationally very expensive to deal with, and reduces query answering performance, and one should question the applicability of the semantics that generate inconsistencies for most type of applications.

Another known issue with RDFS is that there is no differentiation between language constructors and ontology vocabulary, and therefore constructors can be applied to themselves. `(P rdfs:subPropertyOf rdfs:subPropertyOf)`, for example, it is not clear how an RDFS reasoner should behave with such construction. Thankfully this type of construction is rarely used on Semantic Web applications and Linked Data.

[8] summarizes all the above problems among many others motivating the use of an RDFS fragment, called *Minimal* RDFS. This fragment preserves the normative semantics of the core functionalities avoiding the complexity described in [5]. Because *Minimal* RDFS also avoids RDFS constructors to be applied to themselves, it has been proven that algorithms to implement reasoning can be bound within tight complexity limits (see section 4.2 in [8]).

Minimal RDFS is built upon the *ρdf* fragment which includes the following RDFS constructors: `rdfs:subPropertyOf`, `rdfs:subClassOf`, `rdfs:domain`, `rdfs:range` and `rdf:type`[3]. It is worth mentioning that this fragment is relevant because it is non-trivial and associates pieces of data external to the vocabulary of the language. Contrarily, predicates left out from the *ρdf* fragment essentially characterize inner semantics in the ontological design of RDFS concepts.

2.2 4store

4store [4] is an RDF storage and SPARQL query system that became open source under the GNU license in July 2009. Since then, a growing number of users have been using it as a highly scalable quad store. *4store* provides a stable infrastructure to implement decentralized backward chained reasoning: first

[3] For the sake of clarity we use the same shortcuts as in [8] ([sp], [sc] [dom] and [type] respectively).

because it is implemented as a distributed RDF database and second because it is a stable triple store that has been proven to scale up to datasets with 15G triples in both enterprise and research projects.

4store distributes the data in non-overlapping segments. These segments are identified by an integer and the allocation strategy is a simple *mod* operation over the subject of a quad. In the rest of the paper we will represent a quad as a 4-tuple where the first element is the model URI and the remainder is the typical RDF triple structure (subject, predicate, object), and the quad members will be accessed as q_m, q_s, q_p and q_o respectively.

The distributed nature of 4store is depicted in [4], and can be simplified as follows. The data segments are allocated in *Storage Nodes* and the query engine in a *Processing Node*. The *Processing Node* accesses the *Storage Nodes* via sending TCP/IP messages. It also decomposes a SPARQL query into a query plan made of quad patterns. For each quad pattern, it requests a *bind* operation against all segments in each *Storage Node*. The bind operations are run in parallel over each segment and receive as input four lists $\{B_M, B_S, B_P, B_O\}$ that represent the quad patterns to be matched.

3 *Minimal* RDFS and *4sr*'s Distributed Model

This section takes *Minimal* RDFS algorithms from [8] and reformulates them to be implemented as goal directed query answering system (backward chain) over a distributed infraestructure.

We define a knowledge base (KB) as set of n graphs:

$\mathcal{KB} = \{G_1, G_2, G_3, ...G_n\}$

where a graph G_i is a set of quads of the form (m, s, p, o). We also define the set of segments in a distributed RDF store as:

$\mathcal{S} = \{S_0, S_1, S_2, ...S_{m-1}\}$

Quads can be distributed among segments based on different strategies. In our case, 4store's distribution mechanism applies a *mod* operation over a hash[4] of every quad's subject. Therefore, a quad (m, s, p, o)[5] from graph G_i will be allocated in segment S_j if $j = hash(s)\ mod\ m$.

According to our subject based distribution policy, a quad gets allocated to a segment regardless of the graph G_i they belong to, and a graph G_i will be split among m' number of segments where $0 < m' \leq m$. It is worth mentioning that this type of data distribution disseminates the data evenly among $\mathcal{S}$ making no assumptions about the structure or content of the resources.

We now define the vocabulary of RDFS terms supported:

$\rho df = \{sc, sp, dom, range, type\}$

A quad (m, s, p, o) is an *mrdf*-quad *iff* $p \in \rho df$ - $\{\texttt{type}\}$, and G_{mrdf} is a graph with all the *mrdf*-quads from every graph in $\mathcal{KB}$. It is important to mention that the *mrdf*-quads are the statements we need to have in all the data segments in

[4] Although 4store is parameterizable, most deployments use UMAC as hash function.

[5] For the sake of simplicity we will not develop a whole RDF graph model here, we assume that quads are just four-element entities.

order to apply the deductive rules from [8], this process to replicate G_{mrdf} in all segments is described throughout the rest of this section. The following rules, extracted from [8], implement the *Minimal* RDFS semantics:

$$(sp_0)\quad \frac{(\_, A, sp, B)(\_, B, sp, C)}{(G_e, A, sp, C)} \qquad (sp_1)\quad \frac{(\_, A, sp, B)(\_, X, A, Y)}{(G_e, X, B, Y)}$$

$$(sc_0)\quad \frac{(\_, A, sc, B)(\_, B, sc, C)}{(G_e, A, sc, C)} \qquad (sc_1)\quad \frac{(\_, A, sc, B)(\_, X, type, A)}{(G_e, X, type, B)}$$

$$(dom_0)\quad \frac{(\_, A, dom, B)(\_, X, A, Y)}{(G_e, X, type, B)} \qquad (ran_0)\quad \frac{(\_, A, range, B)(\_, X, A, Y)}{(G_e, Y, type, B)}$$

$$(dom_1)\quad \frac{(\_, A, dom, B)(\_, C, sp, A)(\_, X, C, Y)}{(G_e, X, type, B)} \qquad (ran_1)\quad \frac{(\_, A, range, B)(\_, C, sp, A)(\_, X, C, Y)}{(G_e, X, type, B)}$$

These rules have been reformulated taking into account that we are dealing with a quad system and not just with triples. The m element of the quads is irrelevant for the rule condition; in the consequence the m element takes the value of G_e which we consider the graph of entailments contained in $\mathcal{KB}$. The model element in the quad (m) does not play any role unless that the SPARQL query being processed projects named graphs into the query resultset - see section 6.1 Future Work on named graph semantics.

At this point we have the definition of $\mathcal{KB}$, $\mathcal{S}$, and a set of deductive rules for ρdf. Every segment in $\mathcal{S}$ contains a non-overlapping set of quads from $\mathcal{KB}$. One important aspect of 4store's scalability is that the bind operation runs concurrently on every segment of $\mathcal{KB}$. Therefore, we need to look at data interdependency in order to investigate rule chaining locks that can occur between segments.

The chain of rules dependencies for *Minimal* RDFS is shown in Figure 1. In the figure, *mrdf*-quads are marked with a '*', and quads in bold are initial quads not triggered by any rule. The rest of the quads are entailed by triggering one or more rules. There is an interesting characteristic in the chain of rules that can be triggered in *Minimal* RDFS: in any possible chain of rules, only

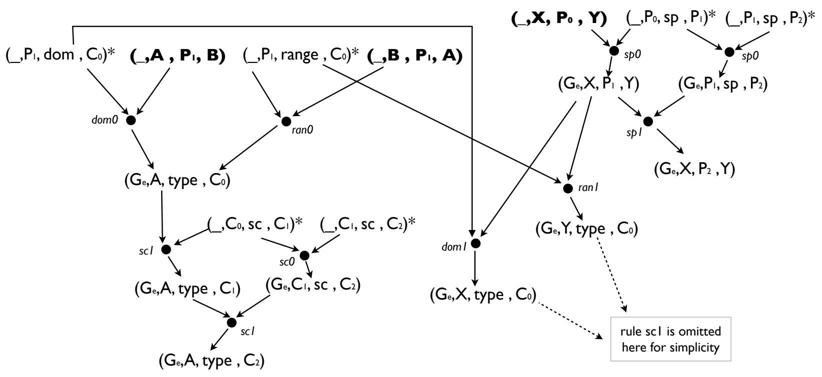


Fig. 1. Rule Chain Tree

one non-*mrdf*-quad that is not in G_e is used. This argument is backed up by the fact that the conditions in the set of deductive rules contain zero or one non-*mrdf*-quads. Therefore to implement distributed reasoning, the only data we need to replicate in every segment of $\mathcal{S}$ is G_{mrdf}, so that G_{mrdf} is accessible to the parallel execution of *bind*. This finding drives *4sr*'s design and it is a novel and unique approach to implement RDFS backward chained reasoning.

4 *4sr* Design and Implementation

The RDFS inferencing in *4sr* is based on two new components that have been incorporated into *4store's* architecture:

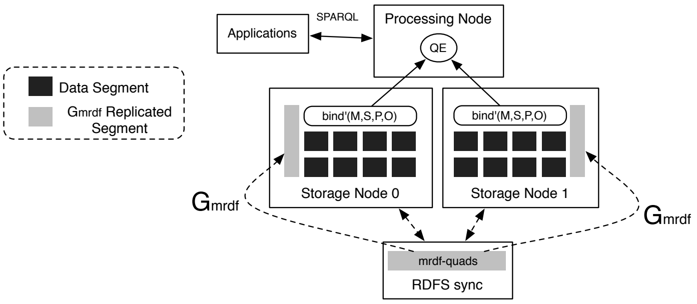


Fig. 2. 4sr Architecture

– **RDFS Sync:** A new processing node to replicate G_{mrdf} called *RDFS sync*. This node gathers all the quads that satisfy the condition to be held in G_{mrdf} from all the segments and replicates them to every Storage Node keeping them synchronized. After every import, update, or delete, this process extracts the new set of quads from G_{mrdf} in the KB and sends it to the Storage Nodes. Even for large KBs this synchronization is fast because G_{mrdf} tends to be a very small portion of the dataset.
– **bind':** The new bind function matches the quads, not just taking into account the explicit knowledge, but also the extensions from the RDFS semantics. This modified *bind'* accesses G_{mrdf} to implement backward chain reasoning. *bind'* is depicted in detail in Section 4.1.

Figure 2 shows in the architecture how *bind'* and RDFS Sync interact for a hypothetical two storage-node deployment. The dashed arrows refer to the messages exchanged between the RDFS Sync process and the Storage Nodes in order to synchronize G_{mrdf}; the arrows between the Processing Node and the Storage Nodes refer to the *bind* operation requested from the Query Engine (QE).

4.1 bind' and *Minimal* RDFS Semantics

In [10] we presented the logical model for *bind'*, a preliminary work that did not take into account the *Minimal* RDFS fragment and was built upon a subset of semantics from [5].

In this paper, we present a remodelled *bind'* that has been reimplemented according to the *Minimal* RDFS semantics and the distributed model described in Section 3. To implement the model, we first consider the following modification of rules dom_1 and ran_1. dom'_1 and ran'_1 - shown below - can replace the original dom_1 and ran_1 keeping *Minimal* RDFS original semantics the same.

$$(dom'_1)\ \frac{(\_, A, dom, B)(\_, C, sp, A)}{(G_e, C, dom, B)} \qquad (ran'_1)\ \frac{(\_, A, range, B)(\_, C, sp, A)}{(G_e, C, range, B)}$$

The rationale for such a replacement is based on the fact chaining dom_0 and dom'_1 generate equivalent entailments to just dom_1. And, similarly, chaining $range_0$ and $range'_1$ is the same as $range_1$.

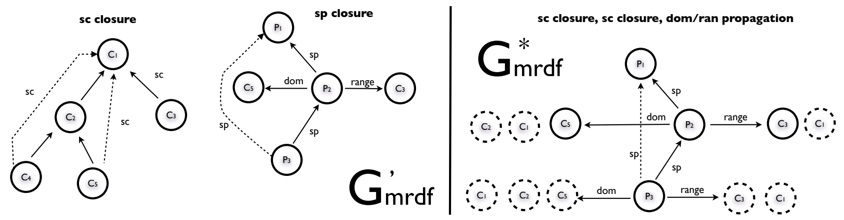


Fig. 3. Construction of G'_{mrdf} and G^*_{mrdf} from G_{mrdf}

Our design relies on the definition of two entailed graphs G'_{mrdf} and G^*_{mrdf}; to deduce these graphs we will entail dom'_1 and $range'_1$ over G_{mrdf}. That process is depicted in Figure 3, and the generated graphs hold the following characteristics:

- G'_{mrdf} is a graph that includes G_{mrdf} and the closure of sp and sc by applying rules sp_0 and sc_0.
- G^*_{mrdf} is a graph that includes G'_{mrdf} plus the deductions from dom'_1 and $range'_1$.

We also define the following operations over G'_{mrdf} and G^*_{mrdf}, where X is considered an arbitrary resource in $\mathcal{KB}$:

- $G'_{mrdf|sc}(X)$ as the subclass closure of X in G'_{mrdf}.
- $G'_{mrdf|sp}(X)$ as the subproperty closure of X in G'_{mrdf}.
- $G^*_{mrdf|dom}(X)$ as the set of properties in G^*_{mrdf} with X as domain.
- $G^*_{mrdf|range}(X)$ as the set of properties in G^*_{mrdf} with X as range.
- $G^*_{mrdf|list}()$ the set of every inferable domain or range in G^*_{mrdf}.
- $G^*_{mrdf|bind}(s, p, o)$ the list of statements that is retrieved from a normal bind operation for a triple pattern (s, p, o).

With these functions we define the access to the graphs G'_{mrdf} and $G_{mrdf}*$, which we should emphasize are accessible to every segment in $\mathcal{S}$. These operations, therefore, will be used for the definition of *bind*'.

A bind operation in 4store is requested by the query processor as we explained in section 2.2. The *bind* receives 4 multisets with the resources to be matched or `NULL` in case some part of the quad pattern is unbound, so a bind to be executed receives as input (B_M, B_S, B_P, B_O). We omit in this paper the description of how the query processor works in 4store, , but for the sake of understanding the system we depict a very simple example:

```
SELECT ?name ?page WHERE { ?x foaf:name ?name .
?x foaf:homePage ?page . ?x foaf:basedNear dbpedia:London }
```

A potential query plan in 4store would issue two bind operations, first the most restrictive query pattern:

$B_0 \Leftarrow bind(NULL, NULL, \{basedNear\}, \{London\})$
$B_1 \Leftarrow bind(NULL, B_{0s}, \{name, homePage\}, NULL\})$

The only point to clarify is that the second bind receives B_{0s} as B_S (B_{0s} refers to the subject element of B_0). The original bind operation in *4store* is made of four nested loops that traverse the indexes in an optimized manner. The ranges of the loops are the input lists (B_M, B_S, B_P, B_O) which are used to build up the combination of patterns to be matched. For simplicity, we draw the following bind function:

Simplified Algorithm of the original bind in 4store:
Input: B_M, B_S, B_P, B_O,segment **Output:** r - a list of quads

```
Let $B^*$ be the list of pattern combinations of $B_M, B_S, B_P, B_O$
For every pattern in $B^*$
    Let T be the radix tree in segment with optimized iterator for pattern
    For every quad in T
        If ($pattern_m$ = $\emptyset$ OR $pattern_m$ = $quad_m$) AND
           ($pattern_s$ = $\emptyset$ OR $pattern_s$ = $quad_s$) AND
           ($pattern_p$ = $\emptyset$ OR $pattern_p$ = $quad_p$) AND
           ($pattern_o$ = $\emptyset$ OR $pattern_o$ = $quad_o$)
           append quad into r
```

4store's real implementation uses radix tries as indexes, and the index selection is optimized based on the pattern to match. This simplified algorithm plays the role of explaining our transformation from original *bind* to the new *bind'* that implements *Minimal* RDFS semantics.

Bind' Algorithm in 4sr:
Input: B_M, B_S, B_P, B_O,segment **Output:** r - a list of quads

```
Let $B^*$ be the list of pattern combinations of $B_M, B_S, B_P, B_O$
For every pattern in $B^*$
```

```
(a) If $|G'_{mrdf|sp}(pattern_p)| > 1$
        append to r bind($pattern_m$,$pattern_s$,$G'_{mrdf|sp}(pattern_p)$,$pattern_o$)
(b) Else If $pattern_p = \emptyset$
        For every pred in segment
            append to r bind'($pattern_m$,$pattern_s$,$pred$,$pattern_o$)
(c) Else If $pattern_p = type$
(c0)    If $pattern_o \neq \emptyset$
            For s in bind($\emptyset$,$\emptyset$,$G^*_{mrdf|dom}(pattern_o)$,$\emptyset$)
                append to r ($G_e$,$sol_s$,$type$,$pattern_o$)
            For s in bind($\emptyset$,$\emptyset$,$G^*_{mrdf|ran}(pattern_o)$,$\emptyset$)
                append to r ($G_e$,$sol_o$,$type$,$pattern_o$)
            append to r bind($pattern_m$,$pattern_m$,$type$,$G'_{mrdf|sc}(pattern_o)$)
(c1)    Else
            For object in $G^*_{mrdf|list}()$
                append to r bind'($pattern_m$,$pattern_s$,$type$,$object$)
(d) Else If $pattern_p \in (sc, sp, range, dom)$
        append to r $G^*_{mrdf|bind}(pattern_s, pattern_p, pattern_o)$
(e) Else
        append to r bind($pattern_m$,$pattern_s$,$pattern_p$,$pattern_o$)
```

This algorithm can be seen as a wrapper of the original *bind* operation; it basically rewrites every combination of quad pattern to be matched according to the *Minimal* RDFS deductive rules. Each of the `if` conditions in the algorithm takes care of one case of backward chain inference. These are described below:

(a) The `if` condition tests whether the closure has more than one element. In such cases we operate `sp` inference by calling the *bind* with an extended *Bp* made by the closure of $pattern_p$. Such a closure is obtained through $G'_{mrdf|sp}(pattern_p)$.

(b) If the predicate pattern is unbound then we recursively call again *bind'* for every predicate in the segment, keeping intact the rest of the elements in the pattern.

(c) This branch is dedicated to the inference with higher complexity, when $pattern_p$ matches RDF `type`. It is divided in two sub-cases:

  (c0) In this case, the pattern matches a specific object ($pattern_o \neq \emptyset$). The first loop binds all the properties for which $pattern_o$ can be inferred as a domain ($G^*_{mrdf|dom}(pattern_o)$). Each solution appends (G_e,sol_s,$type$, $pattern_o$) where sol_s is the subject from the object element from a single solution.
  The second loop is analogous but for ranges. It is worth noticing that this second loop appends as subject sol_o. The reason for this is that in rule ran_o mentions the object that gets the class membership.
  The last append runs the original bind extending $pattern_o$ to the closures of subclasses $G'_{mrdf|sc}(pattern_o)$.

  (c1) This is the opposite case; the object to be matched is null. For such cases we recursively call *bind'* for every inferable class in $G^*_{mrdf|list}()$. The calls generated in this case will be processed in `c0` in subsequent steps.

(d) For patterns where the predicate is any of $(sc, sp, range, dom)$, the pattern match comes down to a simple bind operation over the replicated graph - $G^*_{mrdf|bind}(s, p, o)$.

(e) No reasoning needs to be triggered and a normal bind is processed. The rationale for no reasoning being triggered comes from the fact that in the set of deductive rules, reasoning is processed for patterns where p is one of $(type, sc, sp, range, dom)$ or p is part of a `sp` closure with more than one element. These two conditions are satisfied in (a) and (c). (b) covers the case of $pattern_p = \emptyset$ for which a recursive call for every predicate is requested.

5 LUBM Scalability Evaluation

This evaluation studies *4sr*'s distributed model and its behaviour with different configurations in terms of number of distributed processes - segments - and size of datasets. This analysis is based upon the LUBM synthetic benchmark [3]; we have used 6 different datasets LUBM(100), LUBM(200), LUBM(400), ... , LUBM (1000,0). These datasets progressively grow from 13M triples - LUBM(100,0) - triples to 138M triples LUBM(1000,0). In [10] we presented a preliminary benchmark that demonstrates that *4sr* can handle SPARQL queries with up to 500M triple datasets; this benchmark shows the overall performance of the whole system. The type of benchmark we analyse in this paper, instead of studying performance for big datasets, studies how the bind operation behaves when trying to find solutions that require *Minimal* RDFS reasoning under different conditions, i.e. to see how the bind operations behaves when adding more processors or when the data size is increased. Studying just the bind operation also leaves out components of *4store* that are not affected by the implementation of reasoning, like the query engine.

Our deployment infrastructures are made of two different configurations:

1. **Server set-up:** One Dell PowerEdge R410 with 2 dual quad processors (8 cores - 16 threads) at 2.40GHz, 48G memory and 15k rpm SATA disks.
2. **Cluster set-up:** An infrastructure made of 5 Dell PowerEdge R410s, each of them with 4 dual core processors at 2.27GHz, 48G memory and 15k rpm SATA disks. The network connectivity is standard gigabit ethernet and all the servers are connected to the same network switch.

4sr does not materialize any entailments in the assertion phase, therefore the import throughput we obtained when importing the LUBM datasets is similar to the figures reported by *4store* developers, around 100kT/s for the cluster set-up and 114kT/s for the server set-up[6].

The LUBM benchmark evaluates OWL inference, and therefore there are constructors not supported by *4sr*. We have selected a set of 5 individual triple patterns that cover all the reasoning implemented by *4sr*:

[6] Throughput obtained asserting the data in ntriples, the LUBM datasets had been converted from RDF/XML into ntriples using the rapper tool.

1. **Faculty** {`?s type Faculty`}, Faculty is an intermediate class under the hierarchy rooted by `Person`. To backward entail this query *4sr* will expand `Faculty`'s subclass closure. Moreover, `Faculty` is the `teacherOf`'s predicate domain, therefore, every subject of a triple {`?s teacherOf ?s`} will be part of the solution.
2. **Person** {`?s type Person`}. Similar to the query above, but the closure of `Person` is higher. It contains 16 subclasses, and also Person - or subclasses of it - act as domain and/or range in a number of predicates (`advisor`, `affiliateOf`, `degreeFrom`, `hasAlumnus`, ...). Moreover, these predicates are part of subproperty constructors which makes this query fire all the deductive rules in *4sr*.
3. **Organisation** {`?s type Organisation`}. Binding all the resources type of Organisation will also fire all the deductive rules in *4sr*, but in this case the level predicates that contain Organisation as domain and/or range is lower and also the subclass closure of Person contains fewer elements - just 7.
4. **degreeFrom** {`?s degreeFrom ?o`} such predicates are not instantiated as ground triples and can only be reached by inferring the subproperty closure of it that contains another three predicates - `doctoralDegreeFrom`. `mastersDegreeFrom` and `undergraduateDegreeFrom`.
5. **worksFor** {`?s worksFor ?o`} similar backward chain to be triggered - just subproperty closure.

For the server infrastructure we have measured configurations of 1, 2, 4, 8, 16, and 32 segments. For the cluster infrastructure we measured 4, 8, 16 and 32 - it makes no sense to measure fewer than 4 segments in a cluster made up of four physical nodes.

The queries used in this benchmark have a low selectivity and thus generate large number of solutions; this scenario is suitable for analysing scalability by measuring the number of solutions entailed per second in all the segments for different number of processes (segments) and different size of datasets. The 5 query triple patterns were processed 50 times each, the worst 4 and best 4 measurements were removed from the samples and the time of the worst segment in each measurement was considered, since the binds are processed in parallel and the query processor waits for all to be finished.

Figures 4 and 5 show the results of the benchmark - the Y axis shows the number of solutions and the X axis the number of segments where the data was tested. The benchmark for the server configuration - Figure 4 - shows that the system scales well up to 16 segments, some configurations only up to 8 segments, there is clear degradation for deployments of 32 segments. When the configuration has more segments than CPUs then it is clear that system degrades providing less throughput. In general, the bind operations that require domain and range inference are more expensive than the others - they generate fewer solutions per second. The worst performance was measured for the `Person` bind; this case basically needs to traverse every predicate because the class `Person` happens to be domain and/or range of almost every predicate in the LUBM ontology. The highest throughputs were for `worksFor` and `degreeFrom` binds.

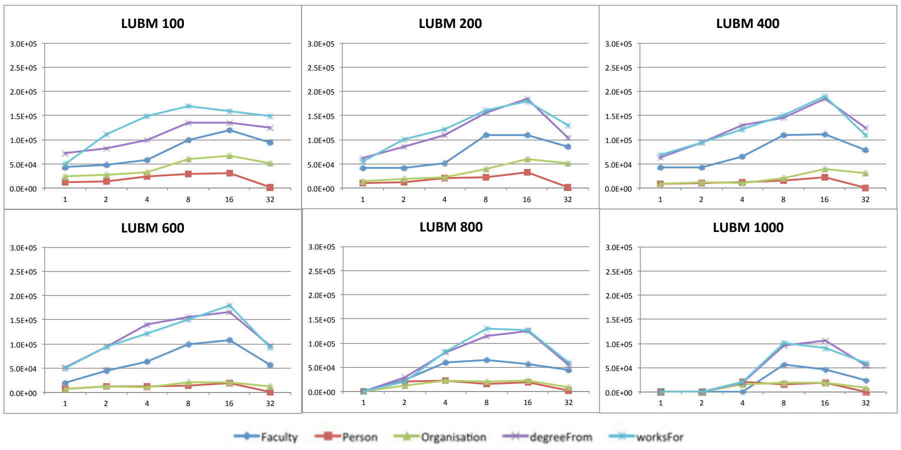


Fig. 4. Server Configuration: Solutions per second per segment

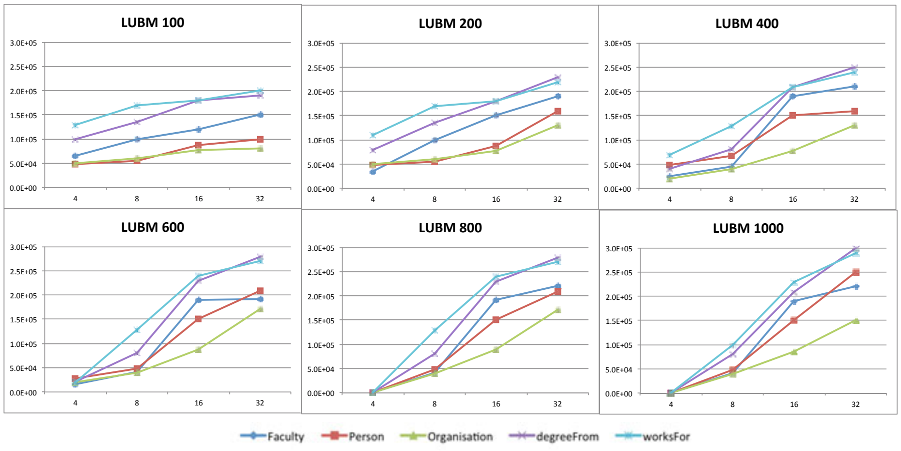


Fig. 5. Cluster Configuration: Solutions per second per segment

For the biggest datasets - LUBM800 and LUBM1000 - the system degraded drastically. For these datasets and 1,2 and 4 segment deployment the system did not respond properly.

The cluster benchmark - Figure - shows better performance. The time needed for transmitting messages over the network gets balanced by the fact that there is a lot better I/O disk throughput. The server configuration has 2 mirrored 15K RPM disks, the same as each of the nodes in the cluster but every node in the cluster can use those disks independently from the other nodes and, the segments collide less on I/O operations.

The performance of the cluster for the biggest datasets - LUBM800 and LUBM1000 - show optimal performance reaching all the binds throughputs between 150K solutions per second and 300K solutions per second. Domain and range inference for `Faculty`, `Organisation` and `Person` show linear scalability and no degradation - unlike in the server configuration. The throughput performance tends to get higher the bigger the dataset is because it generates more solutions without yet reaching the performance limits of the cluster.

In overall, the server configuration sets the scalability limit on the LUBM 400 for the 16-node configuration. For bigger datasets than LUBM400 the server configuration behaves worse. The cluster configuration seems to perform better due to its more distributed nature. It is fair also to mention that, of course, the cluster infrastructure cost is higher than the server and for some applications the performance shown by the server configuration could be good enough.

6 Conclusions and Future Work

In this paper we have presented the design and implementation of *Mininal* RDFS fragment with two novel characteristics decentralisation and backward chaining. We have also disclosed *4sr*'s distributed model and how `subProperty`, `subClass`, `domain`, `range` and `type` semantics can be parallelized by synchronizing a small subset of the triples, namely the ones held in G_{mrdf}. The scalability benchmark showed that the distributed model makes efficient usage of the cluster infrastructure with datasets up to 138M triples. The scalability analysis showed that *4sr* utilises efficiently the cluster infrastructure providing better throughput for bigger datasets.

Since no materialization is processed at the data assertion phase, *4sr* offers a good balance between import throughput and query performance. In that sense, *4sr* will support the development of Semantic Web applications where data can change regularly and RDFS inference is required.

6.1 Future Work

Our plans for future work include the implementation of stronger semantics for named graphs. At the point of writing this paper the research community is discussing how named graphs with attached semantics should behave in a quad store. Our current implementation simply makes G_{mrdf}, G'_{mrdf} and G^{*}_{mrdf} available to every graph and we delegate the semantics of named graphs to the query engine that will treat entailed solutions as part of G_e.

Acknowledgements

This work was supported by the EnAKTing project funded by the Engineering and Physical Sciences Research Council under contract EP/G008493/1.

References

1. Broekstra, J., Kampman, A., Van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, pp. 54–68. Springer, Heidelberg (2002)
2. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations. In: WWW (2004)
3. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics 3(2-3), 158–182 (2005)
4. Harris, S., Lamb, N., Shadbolt, N.: 4store: The Design and Implementation of a Clustered RDF Store. In: Scalable Semantic Web Knowledge Base Systems - SSWS 2009, pp. 94–109 (2009)
5. Hayes, P., McBride, B.: RDF Semantics, W3C Recommendation (February 10, 2004), http://www.w3.org/TR/rdf-mt/
6. Kaoudi, Z., Miliaraki, I., Koubarakis, M.: RDFS Reasoning and Query Answering on Top of DHTs. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 499–516. Springer, Heidelberg (2008)
7. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the Data Skew: Distributed Inferencing by Speeddating in Elastic Regions. In: Proceedings of the WWW 2010, Raleigh NC, USA (2010)
8. Muñoz, S., Pérez, J., Gutierrez, C.: Simple and Efficient Minimal RDFS. Journal of Web Semantics 7, 220–234 (2009)
9. Salvadores, M., Correndo, G., Harris, S., Gibbins, N., Shadbolt, N.: 4sr - Scalable Decentralized RDFS Backward Chained Reasoning. In: Posters and Demos. International Semantic Web Conference (2010)
10. Salvadores, M., Correndo, G., Omitola, T., Gibbins, N., Harris, S., Shadbolt, N.: 4s-reasoner: RDFS Backward Chained reasoning Support in 4store. In: Web-scale Knowledge Representation, Retrieval, and Reasoning, Web-KR3 (2010)
11. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner. Journal of Web Semantics 5(2), 51–53 (2007)
12. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.E.: Owl Reasoning with Webpie: Calculating the Closure of 100 Billion Triples. In: Extended Semantic Web Conference (2010)
13. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 634–649. Springer, Heidelberg (2009)
14. Weaver, J.: Redefining the RDFS closure to be decidable. In: W3C Workshop RDF Next Steps, Stanford, Palo Alto, CA, USA (2010), http://www.w3.org/2009/12/rdf-ws/papers/ws16
15. Weaver, J., Hendler, J.A.: Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 682–697. Springer, Heidelberg (2009)

miKrow: Semantic Intra-enterprise Micro-Knowledge Management System

Víctor Penela, Guillermo Álvaro, Carlos Ruiz, Carmen Córdoba, Francesco Carbone, Michelangelo Castagnone, José Manuel Gómez-Pérez, and Jesús Contreras

iSOCO
Avda. Partenón. 16-18, 28042, Madrid, Spain
{vpenela,galvaro,cruiz,ccordoba,fcarbone,mcastagnone, jmgomez,jcontreras}@isoco.com
http://lab.isoco.net/

Abstract. Knowledge Management systems are one of the key strategies that allow companies to fully tap into their collective knowledge. However, two main entry barriers currently limit the potential of this approach: i) the hurdles employees encounter discouraging them from a strong and active participation (knowledge providing) and ii) the lack of truly evolved intelligent technologies that allow those employees to easily benefiting from the global knowledge provided by them and other users (knowledge consuming). Both needs can sometimes require opposite approaches, tending the current solutions to be not user friendly enough for user participation to be strong or not intelligent enough for them to be useful. In this paper, a lightweight framework for Knowledge Management is proposed based on the combination of two layers that cater to each need: a microblogging layer that simplifies how users interact with the whole system and a semantic powered engine that performs all the intelligent heavy lifting by combining semantic indexing and search of messages and users. Different mechanisms are also presented as extensions that can be plugged-in on demand and help expanding the capabilities of the whole system.

Keywords: enterprise 2.0, knowledge management, social software, web 2.0, microblogging.

1 Introduction

The increasing amount of information generated by enterprises during the last decade has lead to the introduction of the new Knowledge Management (KM) concept, that has grown from a mere accessory to a full discipline that allows companies to grow more efficient and competitive.

Best practices in KM strategies usually attack several key objectives: i) identify, gather and organize the existing knowledge within the enterprise, ii) facilitate the creation of new knowledge, and iii) foster innovation in the company through the reuse and support of workers' abilities. However, in most of the cases, the potential of these approaches doesn't get properly fulfilled by a fundamental flow in their design: prioritizing backend technologies complexity instead of making them easy to use and attractive enough to really encourage final users. This tends to reduce users' participation leading eventually to a loss of the knowledge that these tools are supposed to capture. In order to improve and extend these solutions, the issues detected are approached from two points of view: the system needs to be made both more attractive, so more users get engaged and actively participate, and smarter, so user interaction is minimized as much as possible making the system more proactive.

For increasing the allure of the system, the Web 2.0 paradigm, and in particular the microblogging approach will be used, where end-user involvement is fostered through lightweight and easy-to-use services and applications. These techniques are increasingly penetrating into the context of enterprise solutions, in a paradigm usually referred to as Enterprise 2.0. In particular, the trend of microblogging (of which Twitter[1] is the most prominent example) based on short messages and the asymmetry of its social connections, has been embraced by a large number of companies as the perfect way of easily allowing its employees to communicate and actively participate in the community, as demonstrated by successful examples like Yammer[2], which has implemented its microblogging enterprise solution into more than 70.000 organizations.

Different strategies are used in order to make the system more intelligent. First and foremost a semantically enriched layer supports KM indexing and search processes on top of the atomic information elements, i.e. status updates. Internally the system uses a domain ontology and thesauri related to the particular enterprise in which it is deployed, which can capture the different concepts relating to the company knowledge, and secondly, by making use of other information sources both internal such as already available internal knowledge information and external such as Linked Data[4] resources. Other techniques are also used for expanding and increasing the system intelligence, such as taking into account user Knowledge Processes [18] that can define what is relevant for employees in a particular context as well as reducing the cold start that a system based on user collaboration usually has.

This paper is structured in three main sections: the State of the Art regarding KM and microblogging is described in 2, the proposed theoretical contribution is shown in 3, several extensions that add new value to the original solution are explained in 4 and finally the implementation details and evaluation results are covered in 5.

2 State of the Art

2.1 Knowledge Management

The value of KM relates directly to the effectiveness[3] with which the managed knowledge enables the members of the organization to deal with today's

[1] Twitter: http://www.twitter.com/

[2] Yammer: http://www.yammer.com/

situations and effectively envision and create their future. Due to the new features of the market like the increasing availability and mobility of skilled workers, ideas sitting on the shelf,..., knowledge is not anymore a static resource of the company. It resides in its employees, suppliers, customers,.... If companies do not use the knowledge they have inside, one of their main resources stale.

In recent years computer science has faced more and more complex problems related to information creation and fruition. Applications in which small groups of users publish static information or perform complex tasks in a closed system are not scalable. In 2004, James Surowiecki introduced the concept of "The Wisdom of Crowds"[17] demonstrating how complex problems can be solved more effectively by groups operating according to specific conditions, than by any individual of the group. The collaborative paradigm leads to the generation of large amounts of content and when a critical mass of documents is reached, information becomes unavailable. Knowledge and information management are not scalable unless formalisms are adopted. Semantic Webs aim is to transform human readable content into machine readable. With this goal languages such as RDF Schema and OWL have been defined.

Computer supported collaborative work[10] research analyzed the introduction of Web 2.0 in corporations: McAfee[11] called "Enterprise 2.0", a paradigm shift in corporations towards the 2.0 philosophy: collaborative work should not be based in the hierarchical structure of the organization but should follow the Web 2.0 principles of open collaboration. This is especially true for innovation processes which can be particularly benefited by the new open innovation paradigm[7]. In a world of widely distributed knowledge, companies do not have to rely entirely on their own research, but should open the innovation to all the employees of the organization, to providers and customers.

Web 2.0 tools do not have formal models that allow the creation of complex systems managing large amounts of data. Nowadays solutions like folksonomies, collaborative tagging and social tagging are adopted for collaborative categorization of contents. In this scenario we have to face the problem of scalability and interoperability[9]: making users free to use any keyword is very powerful but this approach does not consider the natural semantic relations between the tags. Semantic Web can contribute introducing computer-readable representations for simple fragments of meaning. As will be seen, an ontology-based analysis of a plain text provides a semantic contextualization of the content, supports tasks such as finding semantic distance between contents and helps in creating relations between people with shared knowledge and interests.

Different mechanisms for leveraging all this scattered enterprise knowledge have been studied during the last decade, particularly trying to ease the pain of introducing new tools in the already overcrowded worker's desktop by adding a semantic layer on top of current applications. CALO[3] based on the use of cognitive systems and NEPOMUK[4] trying to add the social and semantic aspects

[3] CALO is part of the PAL Program: https://pal.sri.com/
[4] NEPOMUK Project: http://nepomuk.semanticdesktop.org/

to the user's personal desktop are two of the main references of ACTIVE[5], a project that aims to increase productivity of knowledge workers with pro-active and contextualized mechanisms and which technology has been used to improve the proposed solution.

2.2 Semantics in Social Networks

Microblogging is one of the recent social phenomena of Web 2.0, being one of the key concepts that has brought Social Web to more than merely early adopters and tech savvy users. The simplest definition of microblogging, a light version of blogging where messages are restricted to less than a small number of characters, does not make true judgment of the real implications of this apparent constraint. Its simplicity and ubiquitous usage possibilities have made microblogging one of the new standards in social communication. There is a large number of social networks and sites, with more blooming every day, that have some microblogging funcionalities, although currently there are two big players in the field: Twitter and Facebook, with 175 and 600 million users respectively.

One of the main issues microblogging has today is the lack of proper semantics, making building any kind of intelligent system on top of them quite hard. Even though different user initiatives have emerged, such as the use of hashtags to define channels of communication and provide a context for the conversation, its use is mostly related to user consumption of the information, not allowing for any real analysis of the meaning of the related data.

Twitter introduced Annotations[6], as a mechanism to add structured metadata about a tweet. It proposes an open key/value structure as properties of a type entity with recommended types such as "place", "movie" or "review". This low level approach is simplistic in the way that it does not define a formal model, but only a mechanism to add metadata to messages.

Facebook has proposed the Open Graph protocol as a mechanism to add metadata to its network, however the target has been quite the opposite, instead of adding metadata to messages as with Twitter Annotations, the main goal is to improve information linking with external resources by proposing a modified RDFa structure for webpages.

SMOB[12] tries to solve this by proposing the use of semantically-enabled hashtags such as #dbp:Eiffel_Tower in #geo:Paris_France. However this approach puts all the burden of explicitly giving meaning to different elements on the user, which is counterproductive with the idea of microblogging as lightweight communication tools.

This lack of semantics is a stronger constraint in a work environment, where employees need to have both faster and more reliable tools for KM while expecting new tools not to disturb their usual work experience and thus not forcing them into having to perform new tasks. Passant et al.[13] extended their previous approach by trying to solve these issues with a mixture of different user-friendly

[5] ACTIVE Project: http://www.active-project.eu/

[6] Twitter Annotations: http://dev.twitter.com/pages/annotations_overview

Web 2.0 interfaces for users to both provide and consume RDF/OWL annotations. This approach still seems quite hard on common employees, experts in their domain but with no basic knowledge on semantic technologies.

3 Semantic Processing in Knowledge Management

In this section, the theoretical contribution of this paper towards KM is described. We address the benefits of applying the the microblogging approach in 3.1, how the processes involved are enriched by the use of semantic indexing and search in 3.2, and the characteristics of the necessary underlying model in 3.3.

3.1 A Lightweight Approach towards Knowledge Management

The proposed interaction platform is a web application designed following the Web 2.0 principles of participation and usability. Our proposal centers interaction around a simple user interface with a single input option for end-users, where they are able to express what are they doing, or more typically in a work environment, what are they working at. This approach diverges from classical KM solutions which are powerful yet complex, following the simplicity idea behind the microblogging paradigm in order to reduce the general entry barriers for end users.

The purpose of the single input parameter where end-users can write a message is twofold: Firstly, the message is semantically indexed so it can be retrieved later on, as well as the particular user associated to it; secondly, because the content of the message itself is used to query the same index for relevant messages semantically related to it, as well as *experts* associated to those messages.

The semantic functionalities are possible thanks to underlying ontologies able to capture the knowledge of the company[6]. Even though there are already one too many ontologies that try to define the global and generic domain of enterprise relationships and knowledge bases, in terms of final performance, the final model must be as coupled as possible with the particular knowledge of each company. Our solution is to be deployed as a standalone service, with no ties with other deployments in other environments (e.g., in other companies) which further emphasizes the need for domain ontologies to be adapted to the particular needs of each company in order to fully tap into its knowledge needs and sources.

These on-demand domain ontologies will be extended with a set of thesauri in order to cover probable variations such as writing mistakes and commonly accepted alterations, making the whole ontology mapping process suitable for a microblogging environment where users feel less inclined to pursue utter correctness.

3.2 Semantic Indexing and Semantic Search

In the proposed approach, status updates (microposts) are stored in the platform knowledge base along with relevant metadata. The text of such messages is analyzed and stored in the message index. The set of terms present in users' statuses compose their entries in the experts index. The text of the messages is used to perform a semantic search against the same index as well.

Semantic Indexing. When a user posts a new status message into the system, its content is analyzed and included into a message index (status repository), allowing future retrieval. Similarly, a repository of expert users (experts repository) is populated by relating the relevant terms of the message with the particular author.

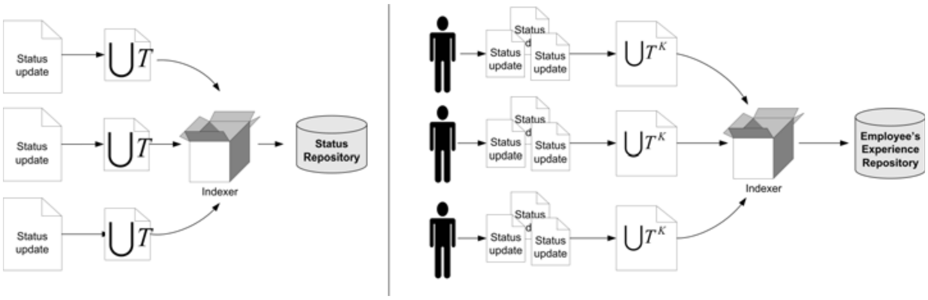


Fig. 1. (a) Message repository creation. (b) Experts repository creation.

Technically, messages that users post to the system are groups of terms T (both key-terms T^K, relevant terms from the ontology domain, and normal terms) $\bigcup T$. The process of indexing each message results in a message repository that contains each document indexed by the different terms it contains, as shown in figure 1(a).

In the case of the update of the semantic repository of experts, which follows the message indexing, each user can be represented by a group of key-terms (only those present in the domain ontology) $\bigcup T^K$. This way, the repository of experts will contain the different users of the systems, that can be retrieved by the key-terms. Figure 1(b) illustrates this experts repository.

Semantic Search. The search process is launched by a user posting a new status update. The new update message is processed by the semantic engine, extracting related concepts from the company knowledge base, modelled as both an ontology and a set of thesauri, and matching them with previously indexed status updates and employees. This is performed seamlessly behind the scenes, i.e., the user is not actively performing a search, but the current status message is used as the search parameter directly.

Two main search approaches are provided by the semantic engine:

– Given the text of a status update, the search on the status index returns semantically related status.

– Given the text of a status update, the search on the experts index returns semantically related people, such as other co-workers with experience on related areas.

From a technical point of view, the semantic repository is queried by using the group of terms $\bigcup T$ of the posted message, as depicted in figure 2(a). This search returns messages semantically relevant to the one that the user has just posted.

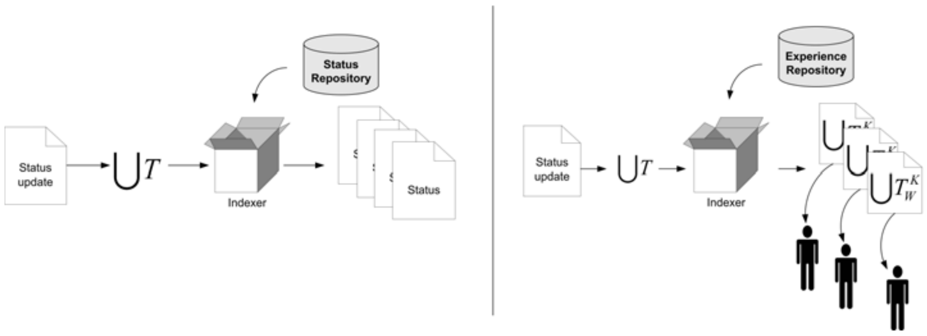


Fig. 2. (a) Detection of related statuses. (b) Expert identification.

It is worth noting that, as it will be covered in 3.3, the search process in the repository is semantic, therefore the relevant messages might contain some of the exact terms present in the current status message, but also terms semantically related through the domain ontology.

As it has been stated above, along with the search for relevant messages, the system is also able to extract experts (identified by the terms present in the messages they have been writing previously) associated with the current status message being posted. In this case, the search over the semantic repository of experts is performed by using the key-terms contained in the posted message $\bigcup T^K$, as depicted in figure 2(b).

3.3 Knowledge Base Modelling

The precision of the modelled knowledge base, which will be built with collaboration from field experts, is a key performance constrain as the semantic engine query process is built upon the defined concepts and relationships. Particularly, the relationships between different elements in both the ontology and the thesauri are exploited through techniques based mainly on morphological variations, orthographic errors and synonyms for the terms defined in the the ontology, in order to expand the initial queries with different approaches that can extend the query recall, without threatening the global query precision.

The analysis of the text is not performed on single words: text fragments and n-grams are considered for ontology matching. A term, composed by one or more words, in a text can match i) general concepts (e.g. the keyword "product" which matches the concept "product" in the ontology), ii) semantic relations between concepts, (e.g. the keyword "target" matches the relation *product_has_target*), or iii) instance entities (e.g., the keyword "Sem10 Engine" which matches the instance *Sem10 Engine*, a particular product of the company). This process can produce any number of matches for each term, strongly depending on the size and number of elements of the ontology, how well it covers the business knowledge base of the company and how the message is related to the core elements in it.

Once associations between terms and ontology entities (concepts, attributes and instances) are identified, the semantic search engine builds a query exploiting these relations defined by the ontology and hence in the company knowledge base. Different expansions can be performed depending on the initial input, with each one being weighed accordingly to the relevance of its relation:

– If a synonym of an ontology term is detected, the ontology term is added to the query.
– If a term corresponding to an ontology class is found, subclasses and instances labels are used to expand the query.
– If an instance label is identified, the corresponding class name and sibling instance labels are added to the query.

4 Knowledge Boosting Techniques

As extensions of the mentioned approach, different mechanisms are proposed in this section in order to extend the original solution with value added information that can improve the final user experience as well as to some extent some of the know issues such as the initial cold start and the limitations and lack of proper up-to-date maintenance of the domain ontology.

These proposed "boosting" features are in no way meant to suppress the original index and search engines, but to expand and upgrade the results provided.

4.1 Tackling the Cold Start Problem by Leveraging Existing Knowledge

One of the key issues the usage of a system like this presents is the delay from its formal deployment and the moment the knowledge base is large enough for its results to be really meaningful. This slow starting path could in many cases be long enough for many companies to desist in their investment in this kind of technologies.

Cold start happens when recommendations from a new item that has not been previously rated or classified are required. Since no user information on the item is available, a classical collaborative filtering approach is useless at that point.

This common issue on recommendation system is usually tackled by a large array of techniques ranging from hard and soft clustering of both users and items to other methodologies based on machine learning and probabilistic methods[15].

In order to overcome this issue, the approach proposed is to leverage current resources available in the prepopulated knowledge base to provide simpler recommendations. That way, even though neither experts or messages will be recommended in the beginning, other secondary elements such as resources, contexts and processes will provide with a first glimpse of the real value of the whole system.

A complementary approach is to query external datasets to perform Named Entity Recognition on the message text, as we cover in the next subsection. This process leverages available datasets in order to provide users with related terms, that even though are not part of the company's knowledge base, could be of some interest. Additionally these relevant terms could also be used as a starting point for the ontology engineering process if seen as having an implicit relevancy as users tend to use them in their conversations.

4.2 Linked Data Consumption

One of the issues of the previous approach is the need of a large ontology that models as close as possible the whole knowledge base of an enterprise, which, depending on the size and the diversity of the company, may differ from difficult to almost impossible (new knowledge concepts being generated almost as fast as they can be modeled). This knowledge curation for the ontology engineering is not only a highly resource consuming process, but also the resulting ontology needs to be maintained and kept up-to-date with new knowledge from the company such as new employees (people), new partners and customers (companies), new business areas (technologies).

As an open approach to tackle this issue the proposed system tries to take advantage of information already available in a structured way via the Linked Data paradigm, providing with an easy and mostly effortless mechanism for adding new knowledge to the system knowledge base. Each new message posted will be processed with NLP methods against the distributed knowledge base that the Linked Data Cloud could be seen as. New concepts or instances extracted from that processing will be added to a temporary knowledge base of terms that could be used to add new information to the system's ontology. These terms would be semiautomatically added to the knowledge via algorithms that weighs the instance usage and the final input of a ontology engineer that decides whether the proposed terms are really valid or is a residue from common used terms with no further meaning to the company.

The main advantage of this approach is that it allows the whole system to adapt and evolve with an organic growth alongside the evolution of the company knowhow. That way, when a new client starts to make business with the company (or even before, when the first contacts are made) some employees will probably start to post messages about it ("Showing our new product to company ACME", "Calling company ACME to arrange a new meeting",...). Querying the Linked Open Data Cloud will automatically detect that this term *ACME* is indeed a company, with a series of properties associated to it (headquarters location, general director and management team, main areas of expertise,...), and would allow for this new knowledge to be easily added to the local knowledge dataset.

4.3 Context-Aware Knowledge Management

In order to extend the relevancy mechanisms proposed, a context-aware[8] approach will extend the current view of messages as the only information element, adding a new layer of external information that could somehow improve the final user experience.

Simple rules will be used for adding a new perspective on top of the previous approach. That way, two employees detected as having a similar level of expertise on a particular topic will be weighed in terms of external data sources such as who is geographically closer (e.g. same office), hierarchically closer (e.g. same department) or available at that particular moment.

For this purpose the original enterprise ontology will be extended by means of an already available context model[5] and the consumption of different services provided by a context-aware infrastructure[14].

4.4 Connecting to Enterprise Information Systems

Even though the global solution is built upon a microblogging environment and obviously focused on lightweight KM, interaction with currently deployed systems in an enterprise environment is a key element in order to ease possible entry barriers as well as leverage already available knowledge information in the company.

As a test use case different levels of information will be extracted from services provided by ACTIVE project[7][16]. ACTIVE aims to increase the productivity of knowledge workers in a pro-active, contextualized, yet easy and unobtrusive way through an integrated knowledge management workspace that reduces information overload by significantly improving the mechanisms through which enterprise information is created, managed, and used. Combining this approach with our microblogging solution will thrive the benefits for workers.

ACTIVE tries to extract information from the whole employee environment, dividing the provided data in three main types of concept:

– Working Context, constructed from a particular set of items (activities, information resources, and people) used to achieve a particular undertaking.
– Resource, seen as placeholder of something that can be used, such as a document or URL.
– Knowledge Process, defined as a loosely defined and structural ramified collection of tasks carried out ky workers as part of their daily activities.

The microblogging tool will extend its classical interface by including links to different instances of each class. These instances will be obtained by consuming ACTIVE services with the detected terms in a particular message as tags for the query and function as interaction channels between both systems, allowing the employee to gather further information and working as a bridge between lightweight KM tool and more resource-intensive platform.

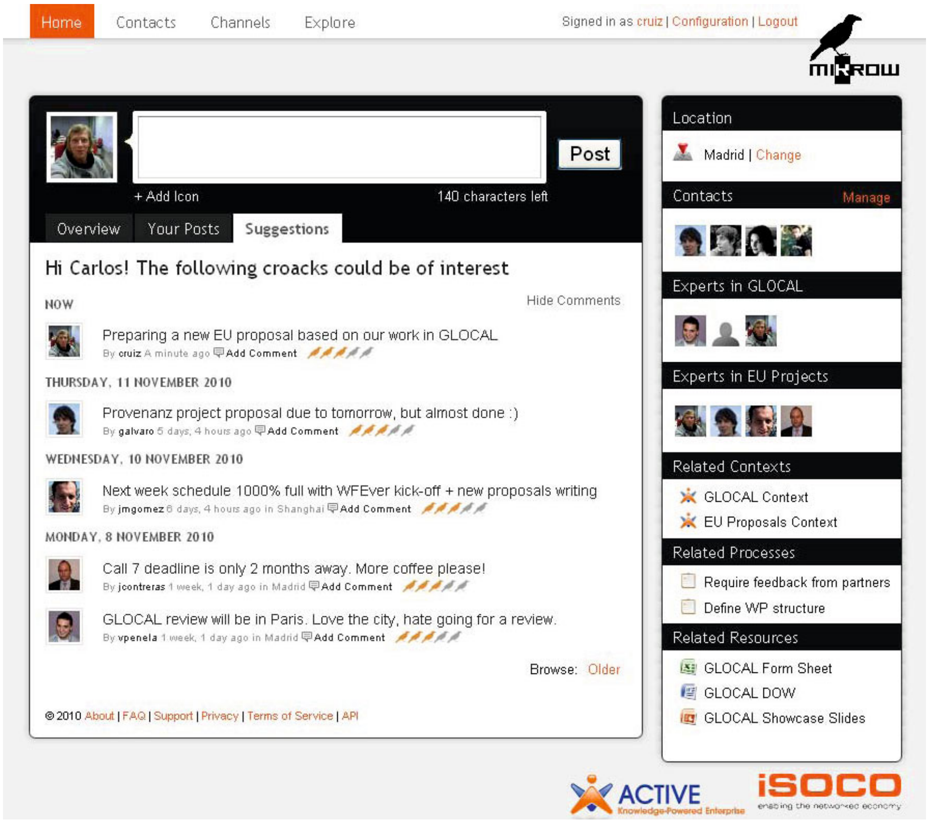
[7] ACTIVE Project: http://www.active-project.eu/

5 Microblogging as a User Interaction Layer

The theoretical contribution covered in the previous section has been implemented as a prototype, codenamed miKrow, in order to be able to evaluate and validate our ideas. In the following subsections, we address the implementation details and the evaluation performed.

5.1 miKrow Implementation

Figure 3 depicts the Web page of the current implementation of miKrow. After a first version that only included the basic indexing and search mechanisms on top of the microblogging layer, as presented in [1], this new iteration has tried to evolve the initial approach, by adding a general improvement on both the backend and frontend, as well as adding new information boosting techniques that try to improve the final user experience, such as integrating functionalities developed



Hi Carlos! The following croacks could be of interest

NOW

Preparing a new EU proposal based on our work in GLOCAL
By cruiz A minute ago Add Comment

THURSDAY, 11 NOVEMBER 2010

Provenanz project proposal due to tomorrow, but almost done :)
By galvaro 5 days, 4 hours ago Add Comment

WEDNESDAY, 10 NOVEMBER 2010

Next week schedule 1000% full with WFEver kick-off + new proposals writing
By jmgomez 6 days, 4 hours ago in Shanghai Add Comment

MONDAY, 8 NOVEMBER 2010

Call 7 deadline is only 2 months away. More coffee please!
By jcontreras 1 week, 1 day ago in Madrid Add Comment

GLOCAL review will be in Paris. Love the city, hate going for a review.
By vpenela 1 week, 1 day ago in Madrid Add Comment

© 2010 About | FAQ | Support | Privacy | Terms of Service | API

Location
Madrid | Change

Contacts Manage

Experts in GLOCAL

Experts in EU Projects

Related Contexts
GLOCAL Context
EU Proposals Context

Related Processes
Require feedback from partners
Define WP structure

Related Resources
GLOCAL Form Sheet
GLOCAL DOW
GLOCAL Showcase Slides

Fig. 3. miKrow implementation snapshot inside ACTIVE project, as well as solve some of the issues raised from the first evaluation performed inside iSOCO[8].

miKrow is divided in two main components, a semantic engine that uses Lucene in order to offer search and indexing functionalities, and a microblogging engine, for which Google's Jaiku[9] has been forked and extended to properly include and show the new type of related information that miKrow offers to the final user.

Microblogging Engine. miKrow microblogging capabilities have been built on top of Jaiku, recently open sourced by Google, using basic microblogging functionalities and UI, relying on it for most of the heavy lifting related to low level transactions, persistence management and, in general, for providing with all the basic needs of a simple social network.

Using Jaiku gives the project a good head start, reducing the burden of middleware and infrastructure development, by reusing already production proved Jaiku's software, and thus allowing to extend that effort and focus on adding the semantically enabled layer.

The choice of Jaiku over other possibilities available is based essentially in its condition of having been extensively tested and the feasibility of being deployed in a Cloud Computing infrastructure[2] such as Google App Engine[10], thus reducing both the IT costs and the burden of managing a system that could have an exponential growth.

Semantic Engine. The semantic functionalities are implemented in a three layered architecture: i) ontology and ontology access, ii) keyword to ontology entity, and iii) the semantic indexing and search as the top layer.

The main functionality is the performance of Named Entity Recognition on each new status update, allowing the extraction of some of the real meaning of a message. This process is performed by parsing each message and analyzing different n-gramms with a base ontology that depicts the enterprise knowledge base and several supporting thesauri, that provides a more extended terms and synonym dataset. Each message is then tagged with the entities that have been extracted from that message.

Lucene[11] is used to create both messages and statuses indices. Each index contains terms tokenized using blank space for word delimitation and ontology terms as single tokens (e.g. if the text contains "credit card" and this is a term of the ontology, "credit", "card" and "credit card" are added as tokens to the index). Ontology terms are detected leveraging the keyword to ontology mapping engine, using the OpenRDF framework[12] as an ontology access mechanism to the ontology, and taking into account possible morphological variations, orthographic errors and synonyms.

[8] iSOCO: `http://lab.isoco.net/`
[9] Jaiku: `http://www.jaiku.com/`
[10] Google App Engine: `http://code.google.com/appengine/`
[11] Lucene: `http://lucene.apache.org/`
[12] OpenRDF: `http://openrdf.org/`

The original semantic engine is also extended by introducing two main additional functionalities, which main goal is to reduce the usual cold start of this type of services:

- Linked Data entities. External services such as OpenCalais[13] are used to connect the messages posted to external entities in the Linked Data paradigm, allowing the system to propose new entities not included in the enterprise ontology.
- Knowledge resources. ACTIVE technology is used to recommend knowledge resources related with the entities extracted from the user messages, lowering the gap between the lightweight tool and more intensive desktop platforms.

Communication between layers. The communication between both layers, the microblogging engine employed as user interface and the semantic engine that provides the business logic on message and experts recommendation as well as the indexing and search functionalities, is highly decoupled and based on Web Services. This approach provides with a more reliable system, since the microblogging engine will keep providing its basic service even if the semantic engine is down or malfunctioning. That way, even though the user experience will be reduced to a simple microblogging environment, lacking any kind of intelligent analysis and recommendation, users will still be able to check messages by themselves and to update their statuses.

5.2 miKrow Evaluation

A first evaluation of the initial and basic version of miKrow, which was not integrated with existing enterprise information systems, was carried in-house inside iSOCO[1], which has around 100 employees distributed in 4 different cities across Spain. A new evaluation has been made by enabling the new miKrow prototype linked with ACTIVE technologies, in order to assess the "knowledge boosting techniques" as well as the semantic benefits.

From a qualitatively point of view, we extracted the following conclusions from the evaluation process:

- The microblogging paradigm has its own rules and syntax, and therefore reusing a knowledge model without adapting it to the special characteristics of this approach implies a decrease in both precision and recall. On one hand, misleading suggestions are caused by stop-words that should not be considered in a microblogging tool, for instance some initial activity gerunds (e.g., *working*, *preparing*). On the other hand, the particular syntax of microblogging implies new ways of expressing the same things in a simpler form (e.g., *ref* instead of *reference*), and hence the thesauri should capture those.
- Temporal relevance of microposts is not to be disregarded. In some occasions, a message is useful only for a short time span, while in others its validity is much longer. User feedback on the suggestions comes in handy to tackle this issue, if they are able to tag messages as no longer valid, etc.

[13] OpenCalais: http://www.opencalais.com/

– Informing users about the reasons for the suggestions (both internal to the tool for messages and experts, and external, for documents found in the existing enterprise information systems) is important, as they perceive some sort of *intelligence* in the system, and are significantly more pleased. Also, if the suggestion is not good, they at least know why it has been produced. Again, letting them provide feedback in these occasions will generate a benefitious loop that will enrich the system.

6 Conclusions

This paper has presented the concept of a semantic microblogging tool to be used within an enterprise network as a lightweight KM service. Even though the Web 2.0 philosophy has been used for a while in work environments, in which is usually called the Enterprise 2.0 paradigm, most of the solutions simply apply a new social layer that does not fulfill the particularities of this kind of environments many times becoming more a resource waste than a added-value tool.

The addition of a semantic layer as an indexing and search engine is the proposed solution in terms of extended intelligence and reliability. This semantic engine is in charge of providing employees with related messages and experts on the topics they are talking about. In order to improve the overall performance a set of ontologies and thesauri will be built to fully model each company knowledge base.

Different extensions have been built in order to improve and extend the current solution by adding new sources of information, while providing the user with a single entry point to the application.

Acknowledgments. This work is partially funded by the European IST projects ACTIVE (ICT-FP7-215040), Wf4Ever (ICT-FP7-270192) and Spanish Avanza I+D Plan project WebN+1 (TSI-020301-2009-24).

References

1. Álvaro, G., Córdoba, C., Penela, V., Castagnone, M., Carbone, F., Gómez-Pérez, J.M., Contreras, J.: mikrow: An intra-enterprise semantic microblogging tool as a micro-knowledge management solution. In: International Conference on Knowledge Management and Information Sharing 2010, KMIS 2010 (2010)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: Above the clouds: A berkeley view of cloud computing. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28 (2009)
3. Bellinger, G.: Systems thinking-an operational perspective of the universe. Systems University on the Net 25 (1996)
4. Berners-Lee, T.: Linked data. International Journal on Semantic Web and Information Systems 4(2) (2006)

5. Cadenas, A., Ruiz, C., Larizgoitia, I., García-Castro, R., Lamsfus, C., Vázquez, I., González, M., Martín, D., Poveda, M.: Context management in mobile environments: a semantic approach. In: 1st Workshop on Context, Information and Ontologies (CIAO 2009), pp. 1–8 (2009)
6. Carbone, F., Contreras, J., Hernández, J.: Enterprise 2.0 and semantic technologies: A technological framework for open innovation support. In: 11th European Conference on Knowledge Management, ECKM 2010 (2010)
7. Chesbrough, H., Vanhaverbeke, W., West, J.: Open Innovation: Researching a new paradigm. Oxford University Press, USA (2006)
8. Dey, A., Abowd, G.: Towards a better understanding of context and context-awareness. In: CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness, pp. 304–307 (2000)
9. Graves, M.: The relationship between web 2.0 and the semantic web. In: European Semantic Technology Conference, ESTC 2007 (2007)
10. Grudin, J.: Computer-supported cooperative work: History and focus. Computer 27(5), 19–26 (1994)
11. McAfee, A.: Enterprise 2.0: The dawn of emergent collaboration. MIT Sloan Management Review 47(3), 21 (2006)
12. Passant, A., Hastrup, T., Bojars, U., Breslin, J.: Microblogging: A semantic and distributed approach. In: Proceedings of the 4th Workshop on Scripting for the Semantic Web (2008)
13. Passant, A., Laublet, P., Breslin, J., Decker, S.: Semslates: Improving enterprise 2.0 information systems thanks to semantic web technologies. In: Proceedings of the 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing (2009)
14. Penela, V., Ruiz, C., Gómez-Pérez, J.M.: What context matters? Towards multidimensional context awareness. In: Augusto, J.C., Corchado, J.M., Novais, P., Analide, C. (eds.) ISAmI 2010. AISC, vol. 72, pp. 113–120. Springer, Heidelberg (2010)
15. Schein, A., Popescul, A., Ungar, L., Pennock, D.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 253–260 ACM,New York (2002)
16. Simperl, E., Thurlow, I., Warren, P., Dengler, F., Davies, J., Grobelnik, M., Mladenic, D., Gomez-Perez, J.M., Ruiz, C.: Overcoming information overload in the enterprise: the active approach. IEEE Internet Computing 14(6), 39–46 (2010)
17. Surowiecki, J., Silverman, M., et al.: The wisdom of crowds. American Journal of Physics 75, 190 (2007)
18. Warren, P., Kings, N., Thurlow, I., Davies, J., Brger, T., Simperl, E., Ruiz, C., Gómez-Pérez, J., Ermolayev, V., Ghani, R., Tilly, M., Bsser, T., Imtiaz, A.: Improving knowledge worker productivity – the active approach. BT Technology Journal 26, 165–176 (2009)

A Faceted Ontology for a Semantic Geo-Catalogue

Feroz Farazi[1], Vincenzo Maltese[1], Fausto Giunchiglia[1], and Alexander Ivanyukovich[2]

[1] DISI - Università di Trento, Trento, Italy
[2] Trient Consulting Group S.r.l., Trento, Italy

Abstract. Geo-spatial applications need to provide powerful search capabilities to support users in their daily activities. However, discovery services are often limited by only syntactically matching user terminology to metadata describing geographical resources. We report our work on the implementation of a geographical catalogue, and corresponding semantic extension, for the spatial data infrastructure (SDI) of the Autonomous Province of Trento (PAT) in Italy. We focus in particular to the semantic extension which is based on the adoption of the S-Match semantic matching tool and on the use of a faceted ontology codifying geographical domain specific knowledge. We finally report our experience in the integration of the faceted ontology with the multi-lingual geo-spatial ontology GeoWordNet.

Keywords: Semantic geo-catalogues, faceted ontologies, ontology integration, entity matching.

1 Introduction

Geo-spatial applications need to provide powerful search capabilities to support users in their daily activities. This is specifically underlined by the INSPIRE[1] directive and regulations [15, 16] that establish minimum criteria for the *discovery services* to support search within the INSPIRE metadata elements. However, discovery services are often limited by only syntactically matching user terminology to metadata describing geographical resources [1]. This weakness has been identified as one of the key issues for the future of the INSPIRE implementation [11, 17, 18, 19].

As a matter of fact, current geographical standards only aim at syntactic agreement [23]. For example, if it is decided that the standard term to denote a harbour (defined in WordNet as "*a sheltered port where ships can take on or discharge cargo*") is *harbour*, they will fail in applications where the same concept is denoted with *seaport.* As part of the solution, domain specific geo-spatial ontologies need do be adopted. In [14] we reviewed some of the existing frameworks supporting the creation and maintenance of geo-spatial ontologies and proposed GeoWordNet - a multi-lingual geo-spatial ontology providing knowledge about geographic classes (features), geo-spatial entities (locations), entities' metadata and part-of relations between them - as one of the best candidates, both in terms of quantity and quality of the information provided, to provide semantic support to the spatial applications.

[1] `http://inspire.jrc.ec.europa.eu/`

The purpose of the Semantic Geo-Catalogue (SGC) project [20] - promoted by the Autonomous Province of Trento (PAT) in Italy with the collaboration of Informatica Trentina, Trient Consulting Group and the University of Trento - was to develop a semantic geo-catalogue as an extension of the existing geo-portal of the PAT. It was conceived to support everyday activities of the employees of the PAT. The main requirement was to allow users to submit queries such as *Bodies of water in Trento*, run them on top of the available geographical resources metadata and get results also for more specific features such as *rivers* and *lakes*. This is clearly not possible without semantic support. As reported in [12], other technological requirements directly coming from the INSPIRE directives included (a) *performance* - send one metadata record within 3s. (this includes, in our case, the time required for the semantic expansion of the query); (b) *availability* - service up by 99% of the time; (c) *capacity* - 30 simultaneous service requests within 1s.

In this paper we report our work on the implementation of the semantic geographical catalogue for the SDI of the PAT. In particular, we focus on the semantic extension of its discovery service. The semantic extension is based on the adoption of the S-Match[2] semantic matching tool [4] and on the use of a specifically designed faceted ontology [2] codifying the necessary domain knowledge about geography and including *inter-alia* the administrative divisions (e.g., municipalities, villages), the bodies of water (e.g., lakes, rivers) and the land formations (e.g., mountains, hills) of the PAT. Before querying the geo-resources, user queries are expanded by S-Match with domain specific terms taken from the faceted ontology. In order to increase the domain coverage, we integrated the faceted ontology with GeoWordNet.

The rest of the paper is organized as follows. Section 2 describes the overall system architecture and focuses on the semantic extension in particular. Section 3 describes the dataset containing the locations within the PAT and how we cleaned it. Sections 4, 5 and 6 provide details about the construction of the faceted ontology, its population and integration with GeoWordNet, respectively. The latter step allows supporting multiple languages (English and Italian), enlarging the background ontology and increasing the coverage of locations and corresponding metadata such as latitude and longitude coordinates. Finally Section 7 concludes the paper by summarizing the main findings and the lessons learned.

2 The Architecture

As described in [1], the overall architecture is constituted by the front-end, business logic and back-end layers as from the standard three-tier paradigm. The geo-catalogue is one of the services of the existing geo-cartographic portal[3] of the PAT. It has been implemented by adapting available open-source tool[4] conforming to the INSPIRE directive and by taking into account the rules enforced at the national level. Following the best practices for the integration of the third-party software into the BEA ALUI

[2] S-Match is open source and can be downloaded from `http://sourceforge.net/projects/s-match/`

[3] `http://www.territorio.provincia.tn.it/`

[4] GeoNetwork OpenSource, `http://geonetwork-opensource.org`

framework[5] (the current engine of the geo-portal), external services are brought together using a portlet[6]-based scheme, where GeoNetwork is used as a back-end. Fig. 1 provides an integrated view of the system architecture. At the front-end, the functionalities are realized as three portlets for:

1. ***metadata management***, including harvesting, search and catalogue navigation functionalities;
2. ***user/group management***, to administer access control on the geo-portal;
3. ***system configuration***, which corresponds to the functionalities of the GeoNetwork's Administrator Survival Tool (GAST) tool of GeoNetwork.

These functionalities are mapped *1-to-1* to the back-end services of GeoNetwork. Notice that external applications, such as ESRI ArcCatalog, can also access the back-end services of GeoNetwork.

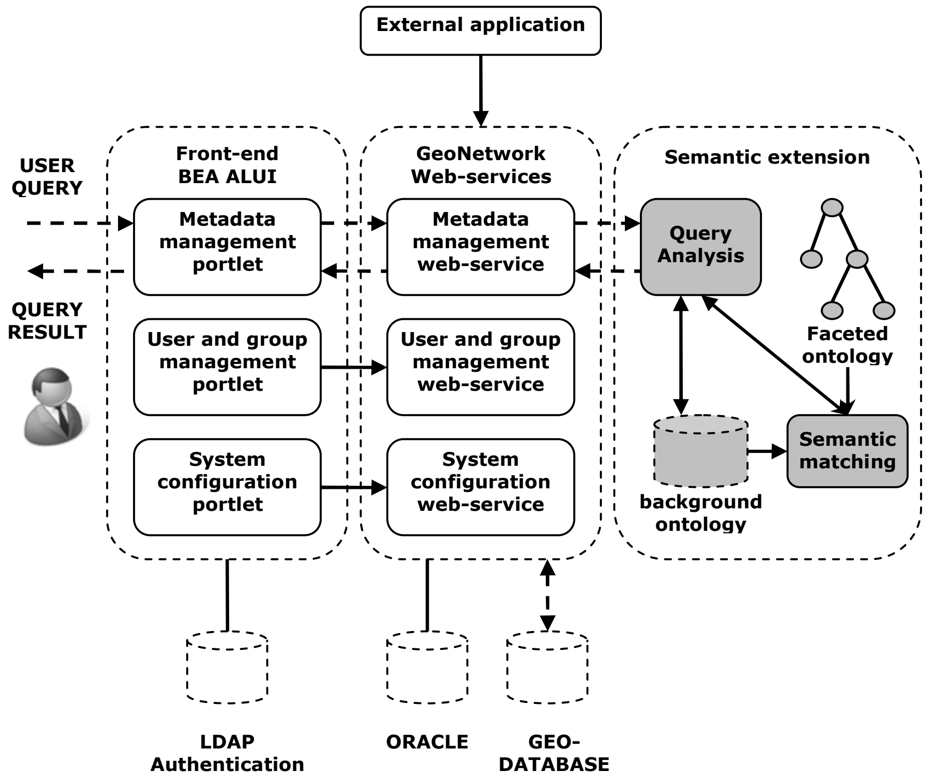


Fig. 1. The overall system architecture

[5] `http://download.oracle.com/docs/cd/E13174_01/alui/`
[6] `http://jcp.org/en/jsr/detail?id=168`

The GeoNetwork catalogue search function was extended by providing ***semantic query processing*** support. In particular, the analysis of the available work in the field, such as [6, 7, 8, 9, 10, 11], summarized in [12], brought to the selection of the S-Match *semantic matching operator* as the best candidate to provide the semantic extension of the geo-catalogue. Given two graph-like structures (e.g., XML schemas) a semantic matching operator identifies the pairs of nodes in the two structures that are semantically similar (equivalent, less or more specific), where the notion of semantic similarity is both at the node level and at the structure level [13, 21, 22]. For instance, it can identify that two nodes labelled *stream* and *watercourse* are semantically equivalent because the two terms are synonyms in English. This allows similar information to be identified that would be more difficult to find using traditional information retrieval approaches.

Initially designed as a standalone application, S-Match was integrated with GeoNetwork. As explained in [1], this was done through a wrapper that provides web services to be invoked by GeoNetwork. This approach mitigates risks of failure in experimental code while still following strict uptime requirements of the production system. Another advantage of this approach is the possibility to reuse this service in other applications with similar needs.

In order to work properly, S-Match needs domain specific knowledge. Providing this knowledge is the main contribution of this paper. Knowledge about the geographical domain is codified into a faceted ontology [1]. A faceted ontology is an ontology composed of several subtrees, each one codifying a different aspect of the domain. In our case, it codifies the knowledge about geography and includes (among others) the administrative divisions (e.g., municipalities, villages), the bodies of water (e.g., lakes, rivers) and the land formations (e.g., mountains, hills) of the PAT.

The flow of information, starting from the user query to the query result, is represented with arrows in Fig. 1. Once the user enters a natural language query (which can be seen as a classification composed by a single node), the query analysis component translates it into a formal language according to the knowledge codified in the background ontology[7]. The formal representation of the query is then given as input to the semantic matching component that matches it against the faceted ontology, thus expanding the query with domain specific terms. The expanded query is then used by the metadata management component to query GeoNetwork and finally access the maps in the geo-database.

At the moment the system supports queries in Italian through their translation in English, uses S-Match to expand feature classes and translates them back to Italian. For instance, in the query *Bodies of water in Trento* only *Bodies of water* would be expanded. Future work includes extended support for Italian and the semantic expansion of the entities such as *Trento* into its (administrative and topological) parts.

3 Data Extraction and Filtering

The first step towards the construction (Section 4) and population (Section 5) of the faceted ontology was to analyze the data provided by the PAT, extract the main

[7] S-Match uses WordNet by default but it can be easily substituted programmatically, for instance by plugging GeoWordNet at its place.

geographical classes and corresponding locations and filter out noisy data. The picture below summarizes the main phases, described in detail in the next paragraphs.

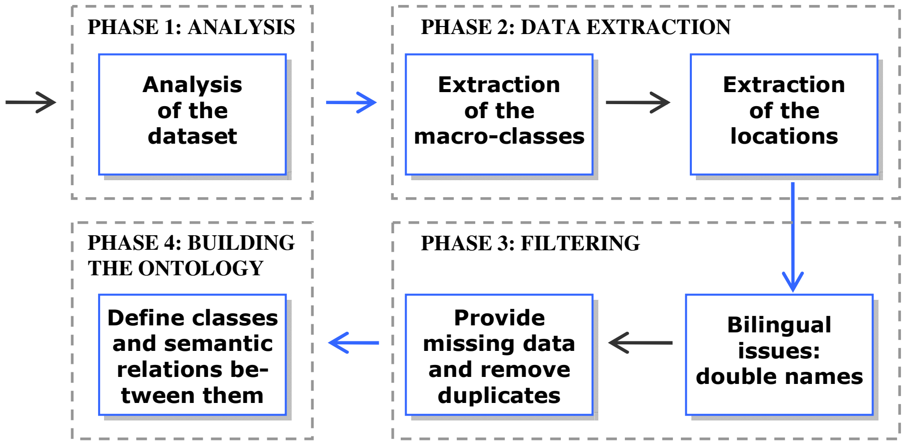

PHASE 1: ANALYSIS
Analysis of the dataset
PHASE 2: DATA EXTRACTION
Extraction of the macro-classes
Extraction of the locations
PHASE 4: BUILDING THE ONTOLOGY
Define classes and semantic relations be-tween them
PHASE 3: FILTERING
Provide missing data and remove duplicates
Bilingual issues: double names

Fig. 2. A global view of the phases for the dataset processing

3.1 The Dataset of the Autonomous Province of Trento

The data are available in MS Excel files (Table 1), and are gathered from the PAT administration. The *features* file contains information about the main 45 geographical classes; the *ammcom* file contains 256 municipalities; the *localita* file contains 1,507 wards and ward parts, that we generically call populated places; the *toponimi* file contains 18,480 generic locations (including *inter-alia* villages, mountains, lakes, and rivers). *Comune*, *frazione* and *località popolata* are the Italian class names for municipality, ward and populated place respectively.

Table 1. The names and descriptions of the files containing PAT data

FILE NAME	DESCRIPTION
features.xls	It provides the name of the feature classes.
ammcom.xls	It provides the name, id, latitude and longitude of the municipalities.
localita.xls	It provides the name, id, latitude and longitude of the wards and ward parts (that we map to populated places). It also provides the id of the municipality a given ward or ward part belongs to.
toponimi.xls	It provides the name, id, class, latitude and longitude of the locations. It also provides the ids of the ward or ward part and municipality a given generic location belongs to.

With the construction of the faceted ontology we identified a suitable name for the rest of the Italian class names from the analysis of the PAT geographical classes in the *features* file. In fact, they are very generic as they are meant to contain several, but similar, kinds of locations. For instance, there is a class that includes springs, waterfalls and other similar entities.

3.2 Extracting the Macro-Classes

We retrieved the main PAT classes, that we call **macro-classes** (as they group different types of locations), from the *features* file. In this file each class is associated an id (e.g., P110) and an Italian name (e.g., *Monti principali*).

Table 2. Names of the administrative classes

CODE	ENGLISH NAME	ITALIAN NAME
E000	Province	provincia
E010	Municipality	comune
E020	Ward	frazione
E021	populated place	località popolata

We did not process the macro-class with id P310 (*Regioni limitrofe*) as it represents locations in the neighbouring region of Trento (out of the scope of our interest) and P472 (*Indicatori geografici*) as it represents geographic codes. Notice that names of the macro-classes needed to be refined as they are too generic and represent many kinds of locations grouped together. As this file lacks classes for the provinces, municipalities, wards and populated places, we created them as shown in Table 2.

3.3 Extracting the Locations

We imported all the locations into a temporary database by organizing them into the part-of hierarchy *province* > *municipality* > *ward* > *populated place* (and other location kinds) as follows:

- ***The province level***. We created an entity representing the Province of Trento. This entity is not explicitly defined in the dataset but it is clearly the root of the hierarchy. We assigned the following names to it: *Provincia Autonoma di Trento*, *Provincia di Trento* and *Trento*. It was assigned to the *province* class.
- ***The municipality level***. Municipalities were extracted from the *ammcom* file. We created an entity for each municipality and a part-of relation between each municipality and the province. They were assigned to the *municipality* class.
- ***The ward and populated place level***. Wards and populated places (sections of wards) were extracted from the *localita* file. Here each ward is connected to the corresponding municipality and each populated place to the corresponding ward by specific internal codes. For each ward and populated place we created a corresponding entity. Using the internal codes, each ward was connected to the corresponding municipality and each populated place to the corresponding ward. They were assigned to the class *ward* or *populated place* accordingly.

- ***All other locations***. All other (non administrative) locations were extracted from the *toponimi* file. Here each of them is connected either to a municipality, a ward or a populated place by specific internal codes. Using the internal codes, we connected them accordingly. A few of them are not connected to any place and therefore we directly connected them to the province. Each location was temporary assigned to the corresponding macro-class.

Locations are provided with latitude and longitude coordinates in Cartesian WGS84 (World Geodetic System 1984) format, a standard coordinate reference system mainly used in cartography, geodesy and navigation to represent geographical coordinates on the Earth[8]. Since in GeoWordNet we store coordinates in WGS84 decimal format, for compatibility we converted them accordingly.

3.4 Double Names: Bilingual Issues

Locations are provided with a name and possibly some alternative names. A few names are double names, e.g., *Cresta di Siusi Cresta de Sousc*. The first (*Cresta di Siusi*) is in Italian and the second (*Cresta de Sousc*) is in Ladin. Ladin is a language spoken in a small part of Trentino and other Alpine regions. The combination of the two names is the official name of the location in the PAT.

In order to identify these cases, the PAT provided an extra text file for each municipality containing individual Italian and Ladin version of the names. In the temporary database, we put the Italian and Ladin names as alternative names. These extra files also contain additional name variants, which are also treated as alternative names. In the end, we found 53 additional Italian names, 53 Ladin names and 8 name variants. For instance, for the location *Monzoni*, the Ladin name *Monciogn* and the name variant *Munciogn (poza)* are provided.

3.5 Provide Missing Data and Remove Duplicates

While importing the entities in the temporary database, we found that 8 municipalities and 39 wards were missing in the *ammcom* and *localita* files respectively, and 35 municipalities were duplicated in the *ammcom* file[9].

Table 3. Objects imported in the temporary database

KIND OF OBJECT	NUMBER OF THE OBJECTS IMPORTED
macro-classes	44
locations	20,162
part-of relations	20,161
alternative names	7,929

[8] `https://www1.nga.mil/ProductsServices/GeodesyGeophysics/WorldGeodeticSystem/`

[9] Note that the missing municipalities are due to the fact that they were merged with other municipalities on 1st January 2010, while the duplicates are related to administrative islands (regions which are not geometrically connected to the main area of each municipality).

We automatically created the missing locations and eliminated the duplicates. At the end of the importing we identified the objects reported in Table 3. Notice that here by part-of we mean a generic containment relation between locations. It can be administrative or topological containment.

4 Building the Faceted Ontology

As mentioned in the previous section, the macro-classes provided by the PAT are very generic and are meant to contain several different, but similar, kinds of locations. This is mainly due to the criteria used by PAT during categorization that where based not only on type but also on importance and population criteria. With the two-fold goal of refining them and determine the missing semantic relations between them, we analyzed the class names and created a multi-lingual faceted ontology. Our goal was to create an ontology that both reflected the specificity of the PAT and respected the canons of the analytico-synthetic approach [5] for the generation of a faceted ontology. A faceted (lightweight) ontology [2] is an ontology divided into subtrees, called facets, each encoding a different dimension or aspect of the domain knowledge. As a result, it can be seen as a collection of hierarchies. The ontology we built encodes the domain knowledge specific to the geographical scope of the PAT and it is therefore suitable for its application in the geo-portal of the PAT administration.

4.1 From Macro-Classes to Atomic Concepts

We started from the 45 macro-classes extracted from the *feature* file that we imported in the temporary database. Notice that they are not accompanied by any description. Therefore, analyzing the locations contained in each macro-class, each macro-class was manually disambiguated and refined - split, merged or renamed - and as a result new classes had to be created.

Table 4. Examples of mapping from macro-categories to atomic concepts

MACRO-CLASSES	**CLASSES**
P410 Capoluogo di Provincia	Province
P465 Malghe e rifugi	Shelter Farm Hut
P510 Antichita importanti P520 Antichita di importanza minore	Antiquity
P210 Corsi dacqua/laghi (1 ord.) P220 Corsi dacqua/laghi (2 ord.) P230 Corsi dacqua/Canali/Fosse/Cond. forz./Laghi (3 ord.) P240 Corsi dacqua/Canali/Fosse/Cond. forz./Laghi (>3 ord.-25.000) P241 Corsi dacqua/Canali/Fosse/Cond. forz./Laghi (>3 ord.)	Lake Group of lakes Stream River Rivulet Canal

This was done through a statistical analysis. Given a macro-class, corresponding locations were searched in GeoWordNet. We looked at all the locations in the part-of hierarchy rooted in the Province of Trento having same name and collected their classes. Only a little portion of the locations where found, but they were used to understand the classes corresponding to each macro-class. Some classes correspond to more than one macro-class. The identified classes were manually refined and some of them required a deeper analysis (with open discussions).

At the end of the process we generated 39 refined classes, including the class *province*, *municipality*, *ward* and *populated place* previously created. Each of these classes is what we call an atomic concept. Some examples are provided in Table 4. They represent examples of 1-to-1, 1-to-many, many-to-1 and many-to-many mappings respectively.

4.2 Arrange Atomic Concepts into Hierarchies

By identifying semantic relations between atomic concepts and following the analytico-synthetic approach we finally created the faceted ontology of the PAT with five distinct facets: *antiquity*, *geological formation* (further divided into *natural elevation* and *natural depression*), *body of water*, *facility* and *administrative division*. As an example, below we provide the *body of water* and *geological formation* facets.

Body of water (Idrografia)
- Lake (Lago)
- Group of lakes (Gruppo di laghi)
- Stream (Corso d'acqua)
    - River (Fiume)
    - Rivulet (Torrente)
- Spring (Sorgente)
- Waterfall (Cascata)
    - Cascade (Cascatina)
- Canal (Canale)

Geological formation (Formazione geologica)
- **Natural elevation (Rilievo naturale)**
    - Highland (Altopiano)
    - Hill (Collina, Colle)
    - Mountain (Montagna, Monte)
    - Mountain range (Catena montuosa)
    - Peak (Cima)
    - Chain of peaks (Catena di picchi)
    - Glacier (Ghiacciaio, Vedretta)
- **Natural depression (Depressione naturale)**
    - Valley (Valle)
    - Mountain pass (Passo)

5 Populating the Faceted Ontology

Each location in the temporary database was associated a macro-class. The faceted ontology was instead built using the atomic concepts generated from their refinement. In order to populate the faceted ontology, we assigned each location in the temporary database to the corresponding atomic concept by applying some heuristics based on the entity names. They were mainly inspired by the statistical analysis discussed in the previous section. As first step, each macro-class was associated to a facet. Macro-classes associated to the same facet constitute what we call a block of classes. For instance, the macro-classes from P110 to P142 (11 classes) correspond to the *natural elevation* block, including *inter-alia* mountains, peaks, passes and glaciers. Facet specific heuristics were applied to each block.

For instance, entities with name starting with *Monte* were considered as instances of the class *montagna* in Italian (*mountain* in English), while entities with name starting with *Passo* were mapped to the class *passo* in Italian (*pass* in English). The general criterion we used is that if we could successfully apply a heuristic we classified the entity in the corresponding class otherwise we choose a more generic class, which is the root of a facet (same as the block name) in the worst case. For some specific macro-classes we reached a success rate of 98%. On average, about 50% of the locations were put in a leaf class thanks to the heuristics.

Finally, we applied the heuristics beyond the boundary of the blocks for further refinement of the instantiation of the entities. The idea was to understand whether, by mistake, entities were classified in the wrong macro-class. For instance, in the *natural depression* block (the 5 macro-classes from P320 to P350), 6 entities have name starting with *Monte* and therefore they are supposed to be mountains instead. The right place for them is therefore the *natural elevation* facet. In total we found 48 potentially bad placed entities, which were checked manually. In 41.67% of the cases it revealed that the heuristics were valid, in only 8.33% of the cases the heuristics were invalid and the rest were unknown because of the lack of information available on the web about the entities. We moved those considered valid in the right classes.

6 Integration with GeoWordNet

With the previous step the locations in the temporary database were associated to an atomic concept in the faceted ontology. The next step consisted in integrating the faceted ontology and corresponding locations with GeoWordNet.

6.1 Concept Integration

This step consisted in mapping atomic concepts from the faceted ontology to GeoWordNet concepts. While building GeoWordNet, we integrated GeoNames classes with WordNet by disambiguating their meaning manually [14]. This time we utilized the experience we gathered in the previous work to automate the disambiguation process with a little amount of manual intervention. An atomic concept from the faceted ontology might or might not be available in GeoWordNet. If available we identified the corresponding concept, otherwise we selected its most suitable parent. This can be done using the Italian or the English name of the class. In our case we used the Italian version of the name. The procedure is as follows:

1. ***Identification of the facet concepts.*** For each facet, the concept of its root node is manually mapped with GeoWordNet. We call it the *facet concept*.
2. ***Concept Identification.*** For each atomic concept C in the faceted ontology, check if the corresponding class name is available in GeoWordNet. If the name is available, retrieve all the candidate synsets/concepts for it. We restrict to noun senses only. For each candidate synset/concept check if it is more specific than the facet concept. If yes, select it as the concept for C. If none of the concepts is more specific than the facet concept, parse the glosses of the candidate synsets. If the facet name is available in any of the glosses, select the corresponding candidate synset/concept as the concept of C.

3. ***Parent Identification.*** If the class name starts with either "group of" or "chain of", remove this string from the name and convert the remaining part to the singular form. Identify the synset/concept of the converted part. The parent of the identified concept is selected as the parent of the class. If the class name consists of two or more words, take the last word and retrieve its synset/concept. Assign this concept as the parent of the atomic concept corresponding to the class. If neither the concept nor the parent is identified yet, ask for manual intervention.

6.2 Entity Matching

Two partially overlapped entity repositories, the temporary database built from the PAT dataset (corresponding to the populated faceted ontology) and GeoWordNet, were integrated. The PAT dataset overall contains 20,162 locations. GeoWordNet already contains around 7 million locations from all over the world, including some locations of the PAT. We imported all but the overlapping entities from the temporary database to GeoWordNet. In order to detect the duplicates we experimented with different approaches. The entity matching task was accomplished within/across the two datasets. We found that the following rules led to a satisfactory result; two entities match if:

Rule 1: name, class and coordinates are the same
Rule 2: name, class, coordinates and parent are the same
Rule 3: name, class, coordinates, parent, children and alternative names are the same

As it can be noticed, Rule 2 is an extension of Rule 1 and Rule 3 is an extension of Rule 2. Parent and children entities are identified using the part-of relation. For example, *Povo* is part-of *Trento*. We have found the following results:

1. ***Within GeoWordNet.*** Applying Rule 1 within GeoWordNet we found 15,665 matches. We found 12,112 matches using Rule 2. Applying Rule 3 we found 12,058 matches involving 22,641 entities. By deleting duplicates these entities can be reduced to 10,583 entities. In fact, if two (or more) entities match by Rule 3 we can safely reduce them by deleting one and keeping the other. Matching entities are clearly undistinguishable.

2. ***Within the temporary PAT database.*** There are 20,162 locations in the PAT dataset. Applying Rule 1 and Rule 2 we found 12 matches and 11 matches, respectively. The result did not change by applying Rule 3 as all of the matched entities are leaves and they have either the same or no alternative name. In total 22 entities matched and we could reduce them to 11.

3. ***Across the two datasets.*** By applying Rule 1 we found only 2 exact matches between the PAT dataset and GeoWordNet, which is far smaller than the number we expected. Therefore, we checked the coordinates of the top level administrative division Trento of the PAT in these two databases manually. We found it with different, nevertheless, very close coordinates. In fact the coordinates were found as (46.0704, 11.1207) in GeoWordNet and (46.0615, 11.1108) in the PAT dataset, which motivated us to allow a tolerance while matching. The result is reported in Table 5. At the end the last one was applied, leading to 174 matches. It corresponds to Rule 3 with an offset of +/- 5.5 Km. We checked most of them manually and they are undistinguishable.

Table 5. Matching coordinates with tolerance

Same name	Same class	Same coordinates	Same parent	Same Children
1385	1160	2 (exact match)	0	0
		11 (using the offset +/-0.0001)	0	0
		341 (using the offset +/-0.001)	13	12
		712 (using the offset +/-0.01)	65	60
		891 (using the offset +/-0.05)	194	174

Note that while matching classes across datasets, we took into account the subsumption hierarchy of their concepts. For example, Trento as *municipality* in the PAT dataset is matched with Trento as *administrative division* in GeoWordNet because the former is more specific than the latter. Note also that the heuristic above aims only at minimizing the number of duplicated entities but it cannot prevent the possibility of still having some duplicates. However, further relaxing it would generate false positives. For instance, by dropping the condition of having same children we found 5% (1 over 20) of false matches.

6.3 Entity Integration

With this step non overlapping locations and part-of relations between them were imported from the temporary database to GeoWordNet following the macro steps below:

1. For each location:
   a. Create a new entity in GeoWordNet
   b. Use the main name of the location to fill the name attribute both in English and Italian
   c. For each Italian alternative name add a value to the name attribute in Italian
   d. Create an instance-of entry between the entity and the corresponding class concept
2. Create part-of relations between the entities using the part-of hierarchy built as described in Section 3.3
3. Generate an Italian and English gloss for each entity created with previous steps

Note that natural language glosses were automatically generated. We used several rules, according to the language, for their generation. For instance, one in English is:

entity_name + " is " + *article* + " " + *class_name* + " in " + *parent_name* + "(" + *parent_class* + " in " + *country_name* + ")";

This allows for instance to describe the *Garda Lake* as *"Garda Lake is a lake in Trento (Administrative division in Trentino Alto-Adige)"*.

7 Conclusions

We briefly reported our experience with the geo-catalogue integration into the SDI of the PAT and in particular with its semantic extension. S-Match, initially designed as a standalone application, was integrated with GeoNetwork. S-Match performs a semantic expansion of the query using a faceted ontology codifying the necessary domain knowledge about geography of the PAT. This allows identifying information that would be more difficult to find using traditional information retrieval approaches. Future work includes extended support for Italian and the semantic expansion of the entities such as *Trento* into its (administrative and topological) parts.

In this work we have also dealt with data refinement, concept integration through parent or equivalent concept identification, ontology population using a heuristic-based approach and finally with entity integration through entity matching. In particular, with the data refinement, depending on the cases, most of the macro-classes needed to be split or merged so that their equivalent atomic concepts or parents could be found in the knowledge base used (GeoWordNet in our case). We accomplished the splitting/merging task manually supported by a statistical analysis, while the integration with the knowledge base was mostly automatic. Working on the PAT macro-classes helped in learning how to reduce manual work in dealing with potentially noisy sources. Entity integration was accomplished through entity matching, which was experimented within and across the entity repositories. The entity matching criteria that perform well within a single repository might need to expand or relax when the comparison takes place across the datasets. Note that entity type specific matchers might be necessary when dealing with different kinds of entities (e.g., persons, organizations, events).

Acknowledgements

This work has been partially supported by the TasLab network project funded by the European Social Fund under the act n° 1637 (30.06.2008) of the Autonomous Province of Trento, by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 231126 LivingKnowledge: LivingKnowledge - Facts, Opinions and Bias in Time and by “Live- Memories - Active Digital Memories of Collective Life” funded by the Autonomous Province of Trento. We are thankful to our colleagues of the Informatica Trentina and in particular to Pavel Shvaiko for the fruitful discussions on the implementation of the geo-catalogue within the geo-portal of the Autonomous Province of Trento. We acknowledge Aliaksandr Autayeu for his support for the integration of S-Match. We are grateful to Veronica Rizzi for her technical support within the SGC project and to Biswanath Dutta for his suggestions for the creation of the faceted ontology. Finally, we want to thank Daniela Ferrari, Giuliana Ucelli, Monica Laudadio, Lydia Foess and Lorenzo Vaccari of the PAT for their kind support.

References

1. Shvaiko, P., Ivanyukovich, A., Vaccari, L., Maltese, V., Farazi, F.: A semantic geo-catalogue implementation for a regional SDI. In: Proc. of the INPSIRE Conference (2010)
2. Giunchiglia, F., Dutta, B., Maltese, V.: Faceted Lightweight Ontologies. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 36–51. Springer, Heidelberg (2009)
3. Giunchiglia, F., Zaihrayeu, I.: Lightweight Ontologies. The Encyclopedia of Database Systems (2007)
4. Giunchiglia, F., Autayeu, A., Pane, J.: S-Match: an open source framework for matching lightweight ontologies. The Semantic Web Journal (2010)
5. Ranganathan, S.R.: Prolegomena to library classification. Asia Publishing House (1967)
6. Cruz, I., Sunna, W.: Structural alignment methods with applications to geospatial ontologies. Transactions in Geographic Information Science 12(6), 683–711 (2008)
7. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
8. Janowicz, K., Wilkes, M., Lutz, M.: Similarity-based information retrieval and its role within spatial data infrastructures. In: Proc. of GIScience (2008)
9. Maué, P.: An extensible semantic catalogue for geospatial web services. Journal of Spatial Data Infrastructures Research 3, 168–191 (2008)
10. Stock, K., Small, M., Ou, Y., Reitsma, F.: OGC catalogue services - OWL application profile of CSW. Technical report, Open Geospatial Consortium (2009)
11. Vaccari, L., Shvaiko, P., Marchese, M.: A geo-service semantic integration in spatial data infrastructures. Journal of Spatial Data Infrastructures Research 4, 24–51 (2009)
12. Shvaiko, P., Vaccari, L., Trecarichi, G.: Semantic Geo-Catalog: A Scenario and Requirements. In: Proc. of the 4th Workshop on Ontology Matching at ISWC (2009)
13. Giunchiglia, F., McNeill, F., Yatskevich, M., Pane, J., Besana, P., Shvaiko, P.: Approximate structure-preserving semantic matching. In: Proc. of ODBASE (2008)
14. Giunchiglia, F., Maltese, V., Farazi, F., Dutta, B.: GeoWordNet: A resource for geo-spatial applications. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 121–136. Springer, Heidelberg (2010)
15. European Parliament, Directive 2007/2/EC establishing an Infrastructure for Spatial Information in the European Community (INSPIRE) (2009)
16. European Commission, COMMISSION REGULATION (EC) No 976/2009 implementing Directive 2007/2/EC as regards the Network Services (2009)
17. Lutz, M., Ostlander, N., Kechagioglou, X., Cao, H.: Challenges for Metadata Creation and Discovery in a multilingual SDI - Facing INSPIRE. In: Proc. of ISRSE (2009)
18. Crompvoets, J., Wachowicz, M., de Bree, F., Bregt, A.: Impact assessment of the INSPIRE geo-portal. In: Proc. of the 10th EC GI & GIS workshop (2004)
19. Smits, P., Friis-Christensen, A.: Resource discovery in a European Spatial Data Infrastructure. Transactions on Knowledge and Data Engineering 19(1), 85–95 (2007)
20. Ivanyukovich, A., Giunchiglia, F., Rizzi, V., Maltese, V.: SGC: Architettura del sistema. Technical report, TCG/INFOTN/2009/3/D0002R5 (2009)
21. Giunchiglia, F., Villafiorita, A., Walsh, T.: Theories of Abstraction. In: AI Communications, vol. 10(3/4), pp. 167–176. IOS Press, Amsterdam (1997)
22. Giunchiglia, F., Walsh, T.: Abstract Theorem Proving. In: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989), pp. 372–377 (1989)
23. Kuhn, W.: Geospatial semantics: Why, of What, and How? Journal of Data Semantics (JoDS) III, 1–24 (2005)

SoKNOS – Using Semantic Technologies in Disaster Management Software

Grigori Babitski[1], Simon Bergweiler[2], Olaf Grebner[1], Daniel Oberle[1], Heiko Paulheim[1], and Florian Probst[1]

[1] SAP Research
{grigori.babitski,olaf.grebner,d.oberle,heiko.paulheim,f.probst}@sap.com
[2] DFKI GmbH
simon.bergweiler@dfki.de

Abstract. Disaster management software deals with supporting staff in large catastrophic incidents such as earthquakes or floods, e.g., by providing relevant information, facilitating task and resource planning, and managing communication with all involved parties. In this paper, we introduce the SoKNOS support system, which is a functional prototype for such software using semantic technologies for various purposes. Ontologies are used for creating a mutual understanding between developers and end users from different organizations. Information sources and services are annotated with ontologies for improving the provision of the right information at the right time, for connecting existing systems and databases to the SoKNOS system, and for providing an ontology-based visualization. Furthermore, the users' actions are constantly supervised, and errors are avoided by employing ontology-based consistency checking. We show how the pervasive and holistic use of semantic technologies leads to a significant improvement of both the development and the usability of disaster management software, and present some key lessons learned from employing semantic technologies in a large-scale software project.

1 Introduction

Disaster management software deals with supporting staff in catastrophic and emergency situations such as earthquakes or large floods, e.g., by providing relevant information, facilitating task and resource planning, and managing communication with all involved parties.

Current situations in disaster management are characterized by incomplete situation pictures, ad-hoc reaction needs, and unpredictability. First of all, these situations require collaboration across organizations and across countries. Second, they expose an ad-hoc need to resolve or prevent damage under extreme time pressure and non-planned conditions. Third, disasters as such are unpredictable by nature, although preventive planning for disasters can be taken.

Besides the ability to adapt to the current situation, software needs to adapt to the end users' needs as well. Members of disaster management organizations are

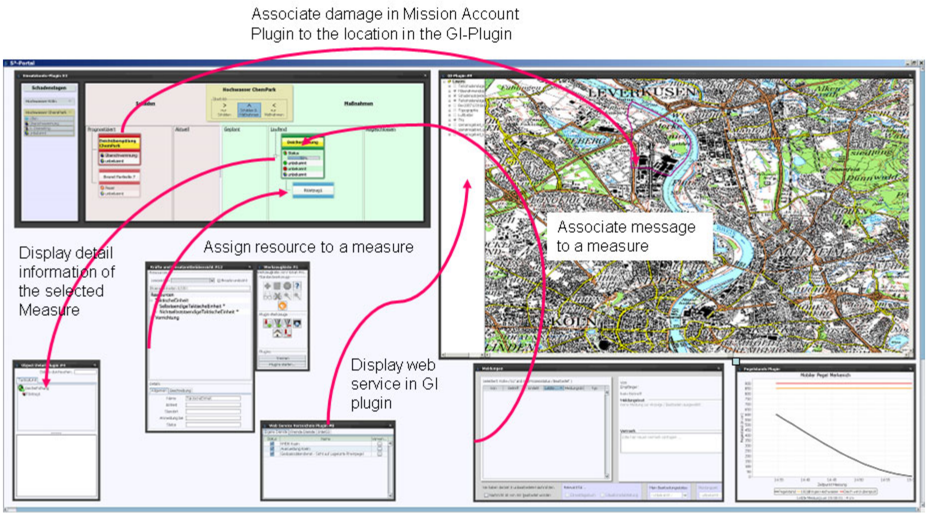

Associate damage in Mission Account Plugin to the location in the GI-Plugin
Display detail information of the selected Measure
Assign resource to a measure
Associate message to a measure
Display web service in GI plugin
LEVERKUSEN
KÖLN

Fig. 1. Screenshot of the SoKNOS prototype [1]

to a large extent non-IT experts and only accustomed to casual usage of disaster management software, since large incidents luckily occur rather infrequently. This casual usage implies that, for example, users may not always have the right terminology at hand in the first place, especially when facing information overload due to a large number of potentially relevant information sources. A large incident poses a stressful situation for all involved users of disaster management software. Users often need to operate multiple applications in a distributed and heterogeneous application landscape in parallel to obtain a consistent view on all available information.

The SoKNOS[1] system [1] is a working prototype (see Fig. 1) for such software using semantic technologies for various purposes. In SoKNOS, information sources and services are annotated with ontologies for improving the provision of the right information at the right time. The annotations are used for connecting existing systems and databases to the SoKNOS system, and for creating visualizations of the information. Furthermore, the users' actions are constantly supervised, and errors are avoided by employing ontology-based consistency checking.

A central design decision for the system was to ensure that any newly created information as well as all integrated sensor information is semantically characterized, supporting the goal of a shared and semantically unambiguous information basis across organizations. In this sense, semantic technologies were used in a holistic and pervasive manner thought the system, making SoKNOS a good example for the successful application of semantic technologies. Fig. 2 shows an overview of the ontologies developed and used in the SoKNOS project.

[1] More information on the publicly funded SoKNOS project can be found at `http://www.soknos.de`

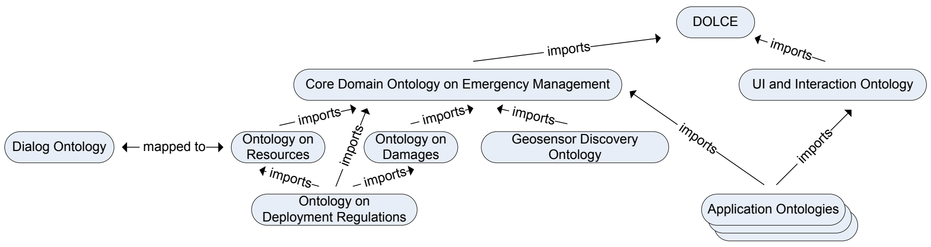

DOLCE
imports
imports
Core Domain Ontology on Emergency Management
UI and Interaction Ontology
Dialog Ontology
mapped to
Ontology on Resources
Ontology on Damages
Geosensor Discovery Ontology
imports
Ontology on Deployment Regulations
Application Ontologies

Fig. 2. Ontologies developed and used in SoKNOS

The central ontology is a core domain ontology on emergency management, aligned to the foundational ontology DOLCE [2], which defines the basic vocabulary of the emergency management domain. Specialized ontologies are used for resources and damages, and deployment regulations defining the relations between resources and damages. Those ontologies have been developed in a close cooperation with domain experts, such as fire brigade officers.

Furthermore, for the definition of system components, ontologies of user interfaces and interactions as well as geo sensors have been developed. Specialized application ontologies can be defined for each application used in the disaster scenario, based on the aforementioned ontologies. For supporting speech based interaction for finding resources, a specialized dialog ontology has been developed, which has a mapping to the resources ontology.

The remaining paper is structured as follows. In section 2, we first give an overview on six use cases where we applied ontologies and semantic technology in the SoKNOS support system. Second, we then detail for each of these use cases the implemented functionality and its benefits for end users (e.g., the command staff in disaster management) and software engineers. As this is a survey on the different usages of semantics in SoKNOS, we only briefly highlight the benefits of applying ontologies, and we refer to other research papers for details on the implementation and evaluation where applicable.

In section 3, we present the lessons learned of the software engineering process. We point out successes and potential for improvements of the semantic technologies employed. Finally, we conclude in section 4 with a summary of the presented use cases and lessons learned.

2 Use Cases and Ontology-Based Improvements in the SoKNOS Disaster Management Application

Six core use cases for ontologies and semantic technologies in disaster management turned out to be of particular interest during the SoKNOS project. We have classified the use cases according to two criteria. First, the use cases provide functionality applicable either during design time (before or after an incident) or during run time (during the incident, including the chaos phase). Second, the functionality is either used by the end users (firefighter, emergency management

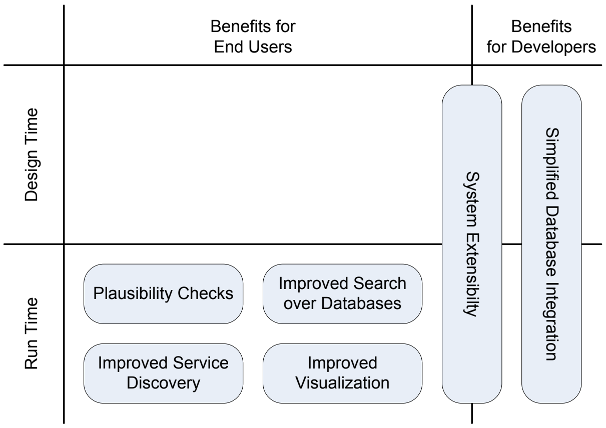


Fig. 3. Six use cases for semantic technologies covered in the SoKNOS project

staff, etc.) or by software engineers. Fig. 3 shows an overview on these use cases identified in SoKNOS, classified according to the two criteria.

Each of the use cases presented in this paper has been implemented in the SoKNOS support system [3]. In this section, we illustrate the benefits of ontologies for the end users in each of these use cases. Furthermore, we explain how this functionality has been implemented using ontologies and semantic technology.

2.1 Use Case 1: System Extensibility

In disaster management, users typically encounter a heterogeneous landscape of IT systems, where each application exposes different user interfaces and interaction paradigms, hindering end users in efficient problem resolution and decision making. For efficiently dealing with emergency situations, *systems have to be extensible* so that applications can be added or integrated with little effort. The SoKNOS prototype features *ontology-based system extensibility* that allows the integration of new system components with comparatively little effort.

For saving as much of the engineering efforts as possible, we integrate applications on the user interface layer, thus being able to re-use as many software components as possible, and confronting the end users with familiar user interfaces. From a software engineering point of view, integration on the user interface level can be tedious, especially when heterogeneous components (e.g., Java and Flex components) are involved. From a usability point of view, we enable the end user to make use of the various existing applications which serve different purposes, such as managing resources, handling messages, or displaying digital maps – some of them may be domain-specific, such as resource handling, others may be domain-independent, such as messaging or geographic information software.

In SoKNOS, we have developed a framework capable of dynamic integration of applications on the user interface level based on ontologies. For the integration, each application is described by an application ontology, based on the SoKNOS set of ontologies (see Fig. 2). Integration rules using the concepts defined in those application ontologies express interactions that are possible between the integrated applications. Integrated applications communicate using events annotated with concepts from the ontologies. A reasoner evaluating those event annotations based on the ontologies and integration rules serves as an indirection between the integrated applications, thus preventing code tangling and preserving modularity and maintainability of the overall system. Details on the integration framework can be found in [4].

Interactions between integrated applications encompass the highlighting of related information in all applications when an object is selected, or the triggering of actions by dragging and dropping objects from one application to another. Moreover, the user is assisted, e.g., by highlighting possible drop locations when dragging an object. This allows users to explore the interaction possibilities of the integrated system. Figure 1 depicts some of the integrated applications in SoKNOS, together with example interactions.

A central innovation is that all components can expect certain events to come with some well-known annotations since all events exchanged between components are annotated using the ontologies from the SoKNOS ontology stack. Therefore, the *integration rules* used to control cross-application interactions can be defined based on those annotations and react to all events based on those annotations. By mapping annotations of events to methods of the integrated components, new components can therefore be added at run-time without having to change the system, as long as the annotations of the events remain constant, which in turn is ensured by using shared ontologies. For adding a new application to SoKNOS, an appropriate application ontology has to be written, and the mapping from events to methods has to be defined.

Although a reasoner is involved in processing the events exchanged between applications, a sophisticated software architecture of the framework ensures that the event processing times are below one second, thus, the user experience is not negatively affected by the use of semantics. Details on the architecture and the performance evaluation can be found in [5].

2.2 Use Case 2: Simplified Database Integration

Numerous cooperating organizations require the integration of their heterogeneous, distributed databases at run time to conduct efficient operational resource management, each using their own vocabulary and schema for naming and describing things. As discussed above, having the relevant information at the right place is essential in disaster management. Such information is often stored in databases. One typical example in SoKNOS are databases of operational resources (e.g. fire brigade cars, helicopters, etc.), maintained by different organizations, such as local fire brigades. Since larger incidents require ad hoc cooperation of such organizations, it is necessary that such databases can be integrated even at run-time.

In the SoKNOS prototype, we have used OntoBroker as an infrastructure for integrating databases. As discussed in [6], OntoBroker can make instance data stored in relational databases or accessed via Web service interfaces available as facts in the ontology. The SoKNOS "Joint Query Engine" (JQE) uses that infrastructure to connect different heterogeneous resource databases in the disaster management domain owned by different organizations.

A simple graphical user interface allows the user to pose queries against the different databases. The JQE processes the query by unifying different names of input-concepts like, e.g. "helicopter", defined in the SoKNOS resources ontology, which is mapped to the different underlying databases' data models. The query is then translated to a number of queries to the connected databases, and the results are unified and returned to the querying application. In [5], we have shown that directly passing the queries to underlying data sources is faster than materializing the facts from the databases in the reasoner's A-box.

For more sophisticated use cases, such as plausibility checking (see section 2.5), the JQE also provides an interface for reasoning on the connected data sources. The query for the integrated reasoning engine must be formulated in frame-logic (F-Logic), a higher order language for reasoning about objects [7]. The user interfaces, however, abstract from the query language and provide simple, graphical access.

For establishing the mappings between the resources ontology and the different databases' data models, SoKNOS provides a user interface for interactively connecting elements from the data model to the resources ontology. We have enhanced the interface described in [6] by adding the SAP AutoMappingCore [8], which makes suggestions for possible mappings based on different ontology and schema matching metrics. Internally, the mappings created by the user are converted to F-Logic rules which call database wrappers. Thus, a unified interface to heterogeneous databases is created using semantic technologies.

2.3 Use Case 3: Improved Search

As discussed in the previous section, one of the challenges in disaster management is to quickly and reliably find suitable and available operational resources to handle the operation at hand, even under stressful conditions. The challenge in SoKNOS was to combine a spoken dialog system and the Joint Query Engine (JQE), described in section 2.2, in a multilevel process in order to arrive at a more intuitive semantic search. This approach enables domain experts to pose a single query by speech that retrieves semantically correct results from all previously integrated databases.

Domain experts formulate queries using flexible everyday vocabulary and a large set of possible formulations referring to the disaster domain. Natural spoken interaction allows a skillful linguistic concatenation of keywords and phrases to express filtering conditions which leads on one hand to more detailed queries with more accurate results and on the other hand shortens the conventional query interaction process itself. For example, the spoken query "Show me all available helicopters of the fire fighters Berlin" may result in the display of the two relevant available helicopters in Berlin along with an acoustical feedback "Two available helicopters of the fire fighters Berlin were found". The ontology-based search approach developed to that end improves conventional search by mouse and keyboard interactions through the addition of spoken utterances. Implementation details on the dialog system can be found in [9].

The core components of the spoken dialog system are a speech recognizer and a speech interpretation module. The recognized speech utterances are forwarded to the speech interpretation module, which decomposes the speech recognizer result into several sub-queries. The speech interpretation module relies on an internal ontology-based data representation, the so called dialog ontology, which defines domain knowledge and provides the basic vocabulary that is required for the retrieval of information from natural language. Based on this dialog ontology, the speech interpretation module can resolve ambiguities and interpret incomplete queries by expanding the input to complete queries using the situational context. For processing the spoken queries with the JQE, the dialog ontology is mapped to the SoKNOS resources ontology, as shown in Fig. 2.

The spoken dialog system translates the natural language query into combined F-Logic expressions which the JQE processes to deliver search results (see above). According to the task of giving an incident command, the development of the dialog system was focusing on rapid and specific response to spoken domain-specific user input, rather than on flexible input allowing for a broad common vocabulary.

Simple measurements for the project work, carried out on a standard desktop PC, have shown that the complete processing times of all integrated parsing and discourse processing modules are in the range of milliseconds. This examination shows that the processing of speech input will take place within the dialog platform in real time. However, with increasing complexity of the knowledge domain, the vocabulary and thus the complexity of the generated grammar also increase, which in turn affects the runtime of the developed module. Detailed examinations of the influence of the complexity of the knowledge domain on the runtime behavior will be a subject of future research.

2.4 Use Case 4: Improved Discovery of External Sensor Observation Services

Semantically correct, hence meaningful integration of measurements, for example a system of water level sensors in a river, is especially problematic in situations with high time pressure and low familiarity with a (sub) domain and its terminology. In such cases, the crisis team member might be in need for additional information and is just missing the appropriate search term. The SoKNOS support system addresses this need by providing an ontology-based service discovery mechanism.

The crisis team member benefits from this functionality by being able to integrate specific sensor information quickly, e.g., the water level of a river or the concentration of a pollutant.

In our approach, Web services designed according to the SOS[2] specification, are semantically annotated. To this end, we have developed a geo sensor discovery ontology which formalizes both observable properties (for example wind speed, substance concentration etc.) and the feature of interest (e.g., a particular river, a lake or a city district). The annotation is performed by extending the standard service description with URLs pointing to the respective categories in the ontology.

To facilitate discovery, we have established a way to determine the observable property of interest, based on the ontology. The crisis team member specifies a substance or geographic object (e.g., river) to which the observable properties may pertain (e.g., water level, or stream velocity). The latter are then determined through the relation between an observable property and its bearer, as formalized in the ontology. To get sensor data, the end users finally specify the area of interest by marking this area on a map, provided by a separate module in the SoKNOS System, and by specifying the desired time interval. Details of the implementation can be found in [10].

2.5 Use Case 5: Plausibility Checks

In an emergency situation, the stress level in the command control room is high. Therefore, the risk of making mistakes increases over time. Mistakes in operating the system can cause severe problems, e.g., when issuing inappropriate, unintended orders. Therefore, it is important to double check the users' actions for adequacy and consistency, e.g. by *performing automatic plausibility checks.* Missing plausibility checks in disaster management solutions further increase stress and hence errors on end user side. For example, the system checks the plausibility of an assignment that a crisis team member issues and warns the user if the assignment of tactical unit (e.g., a fire brigade truck) to a planned task (e.g., evacuating a building) does not appear plausible.

Plausibility checks have been considered very useful by the end users, however, they do not want to be "over-ruled" by an application. Therefore, it is important to leave an open door for doing things differently – the system should therefore warn the user, but not *forbid* any actions explicitly. As emergencies are per definition unforeseeable, the system has to provide means for taking unforeseeable actions instead of preventing them.

While such checks may also be hard-coded in the software, it can be beneficial to perform them based on an ontology, such as proposed in [11]. From an engineering point of view, delegating consistency checking to an ontology reasoner reduces code tangling, as consistency checking code may be scattered way across an application. Furthermore, having all statements about consistency in one ontology eases maintainability and involvement of the end users.

[2] The Sensor Observation Service Interface Standard (SOS) is specified by the Sensor Web Enablement (SWE) initiative of the Open Geospatial Consortium (OGC); `http://www.opengeospatial.org`

In SoKNOS, the ontology on deployment regulations contains the information about which operational resource is suitable for which task. Based on this ontology, a reasoner may check whether the assigned unit is suitable for a task or not. The domain knowledge involved in this decision can be rather complex; it may, for example, include information about the devices carried by a tactical unit, the people operating the unit, the problem that is addressed by the task, and so on.

The implementation is quite straight forward: when the user performs an assignment of a resource to a task, an F-Logic query is generated and passed to the JQE (see above), which asks whether the resource is suitable for the respective task. Based on the knowledge formalized in the ontology, the reasoner answers the query. The processing time of the query is below one second, so the user can be warned instantly, if required, and is not interrupted in her work.

2.6 Use Case 6: Improved Information Visualization

In a typical IT system landscape, information is contained in different IT system. Finding and aggregating the information needed for a certain purpose is often a time consuming task.

In SoKNOS, we have used the semantic annotations that are present for each information object for creating an *ontology-based visualization* of the data contained in the different systems. Each IT system integrated in SoKNOS has to offer its data in a semantically annotated format, comparable to Linked Data [12]. Those different annotated data sets can be merged into a data set comprising the information contained in all the systems.

Based on that merged data set, an interactive graph view is generated by the *Semantic Data Explorer (SDE)*. The user can interact with the SDE and the existing applications' interfaces in parallel, thus allowing a hybrid view for data exploration[3]. User studies have shown that for complex information finding tasks in the domain of emergency management, such as finding resources that are overbooked, the Semantic Data Explorer leads to significant improvements both in task completion time and in user satisfaction. Details on the architecture and the user study can be found in [13].

3 Lessons Learned – Disaster Management Applications and Ontologies

In the three years of research and development within SoKNOS, we have implemented the use cases sketched in this paper, employing semantic technologies in numerous places. We present our lessons learned from that work in three parts: the ontology engineering process, the software engineering process, and the usage and suitability of ontologies in the disaster management domain.

[3] A demo video of the Semantic Data Explorer can be found at `http://www.soknos.de/index.php?id=470`.

3.1 Ontology Engineering Process

Involving end users. The early and continuous involvement of end users actually working in the disaster management domain was a core success factor.

On the one hand, discussions with end users to *gather and verify the domain understanding* are at the core of both the ontology engineering process and the application development. In our case, we built the ontology and applications based on multiple workshops with the German firefighting departments of Cologne and Berlin. Additionally, using real-life documents such as working rules and regulations and user generated knowledge sources such as Wikipedia supported the re-construction and formalization of the knowledge to a great extent.

On the other hand, *evaluating the ontologies* with end users helped to consolidate and validate the engineered ontologies. As a result of frequent evaluation workshops, we were able to consolidate the overall ontology and tighten the representation of the domain.

Establishing the role of an ontology engineer. The ontology engineering task as such *usually cannot be executed by end users.* The person taking the role of an ontology engineer needs to work in close cooperation with the end users as well as the application's business logic developer. Knowledge exchange with end users is important to gather a correct and complete understanding of the domain, while software developers need support in understanding the ontology's concepts.

Working with end users, the role requires knowledge in both ontology engineering and the respective domain that is to be formalized. We found that the task of formalizing the domain terminology cannot be done by end users as it requires ontology awareness, for example in the form of taking modeling decisions and obeying to modeling conventions. Without this awareness, no formally correct ontology will emerge, and the ontology cannot be used for reasoning tasks. Working with software developers, the ontology engineer can communicate the usage of the ontology in the application and thus develop an awareness where and how to place semantic annotations.

We had good experiences with an *ontology engineer in a dedicated role* embedded in the project team. The ontology engineer worked with both end users and software developers. This way, we could map the end users' domain knowledge efficiently into the application's business logic.

Finding the right tools. Current tools are rarely designed with end users, i.e., laymen with respect to ontological modeling, in mind. In fact, ontology editors are especially weak when it comes to complex ontologies [14].

The complex terminology modeling involved in ontology engineering is hard to comprehend for domain experts in case of existing modeling artifacts. We experienced domain experts in the disaster management as not been trained in any modeling environment.

Ontology editors need improvement in their "browsing mechanisms, help systems and visualization metaphors" [14], a statement from 2005 which unfortunately still holds true. Better ontology visualization helps to quickly gain an understanding of the ontology and better browsing mechanisms helps editors get better suited to modeling laymen and domain-knowledgeable end users.

Details on the ontology engineering process in SoKNOS can be found in [15].

3.2 Software Engineering Process and Ontologies

Developing new mechanisms for semantic annotations. In SoKNOS, we have relied on semantic annotation of all data within a SoKNOS system. To this end, data models and ontologies need to be interrelated, so that each data object instance can be semantically annotated. Based on these annotations, various useful extensions to the system, as sketched in section 2, have been implemented.

Current approaches for interrelating class models and ontologies most often assume that a 1:1 mapping between the class model and the ontology exists. With these approaches, the mapping is *static*, i.e. each class is mapped to exactly one ontological category, and each attribute is mapped exactly to one ontological relation. Moreover, most annotation approaches are implemented in an *intrusive* fashion, which means that the class model has to be altered in order to hook it up with the annotation mechanism, e.g., by adding special attributes and/or methods to classes. In SoKNOS, we have learned that both these premises – *static* annotation and *intrusive* implementation – are not practical in a real world software engineering setting.

The goals of ontologies and class models are different: class models aim at simplification and easy programming, while ontologies aim at complete and correct formal representations. Thus, we cannot assume that a 1:1 mapping always exists, and static approaches are likely to fail. A typical example from SoKNOS is the use of one common Java class for predicted as well as for actual problems, distinguished by a flag. While this is comfortable for the programmer, the class cannot be statically mapped to a category in the ontology, because predicted and actual problems are very different ontological categories.

Semantic annotations of data objects must feature a non-intrusive implementation for non-alterable legacy code or binary packages where source code and class model cannot be altered. Class models often come as binary packages or are created by code generators which do not allow changes to the way the classes are generated, or licenses of integrated third-party components forbid altering the underlying class models. In these cases, intrusive implementations for semantic annotation are bound to fail.

As a consequence, we have developed a novel approach for semantic annotation of data objects in SoKNOS. This approach is dynamic, i.e. it allows for computing the ontological category a data object belongs to at run-time, and it is implemented in a non-intrusive way: The rules for annotating objects are stored separately from the class models, and an annotation engine executes those rules by inspecting the objects to annotate, using mechanisms such as Java reflection. Thus, class models which cannot or must not be altered can be dealt with [16].

Addressing performance. An essential requirement for user interfaces in general is high reactivity. On the other hand, introducing a reasoning step in the event processing mechanism is a very costly operation in terms of run time. As reaction time of two seconds for interactions is stated as an upper limit for usability in the HCI literature, high reactivity of the system is a paramount challenge.

We have evaluated different architectural alternatives with respect to run time performance and scalability to larger integrated systems, such as centralized vs. decentralized event processing, or pushing vs. pulling of dynamic instance data into the reasoner's A-box (assertion component). With our optimized system design, we are able to keep the reaction times on a reasonable level, even for a larger number of integrated applications used in parallel, with a relatively high number of integration rules (where each integration rule controls one type of cross-application interaction). Details on the evaluation can be found in [5].

3.3 Ontology Usage and Suitability

Finding the right modeling granularity. Both domain experts and end users face problems in dealing with concepts needed due to ontology formalisms. We observed that end users were irritated by the concepts and methods imposed by the use of a strictly formal top level ontology such as DOLCE, which were not part of their colloquial language, but needed for formally correct modeling. We found two sources of "problematic" concepts where domain experts and end users had problems with.

First, *domain experts were not used to concepts needed to create a formally correct ontology*, e.g., as induced by using top level ontologies. Using reference ontologies, like in our case the DOLCE top level ontology, requires complying with a certain structure and formalism. For example, the SoKNOS ontologies are based on DOLCE which uses concepts like "endurant" and "perdurant" to cater for semantic interoperability between information sources. However, from a domain expert's point-of-view, this terminology is complicated to understand compared to normal, colloquial language usage [17]. In our case, professional firefighters as the domain experts were irritated by these concepts.

Second, *end users were irritated by modeled domain terminology that was not part of their colloquial language.* There are concepts that firefighters don't use in their colloquial language but which are needed for a formally correct modeling of the ontology, e.g., to satisfy reasoning requirements. For example, resources are categorized in the ontology via their usage characteristics. In the given example, the class "rescue helicopter" is sub-class of the classes "equipment", "means of transportation", "motorized means of transportation", "flying motorized means of transportation", "helicopter" and "ground-landing helicopter". Except for the term "equipment", all other terms are not part of a firefighter's colloquial language but are needed have a formally correct ontology and support useful reasoning.

In SoKNOS, we have found that this question cannot be answered trivially, as a heuristic for identifying an optimal proportion between "every day concepts" and "top level concepts" in the ontology is missing. Having a dedicated ontology engineer, as discussed above, involved in the ontology engineering session helped the domain experts understand the need and the intention of top level concepts.

Finding the right visualization depth. Offering only a class browser with a treelike visualization of the ontology's extensive OWL class hierarchy caused confusion among end users. The SoKNOS inventory management application visualized the modeled OWL class hierarchy directly in a class browser. Here, the end user for example can browse resources like cars, trucks and aircrafts. However, due to the numerous concepts and the extensive class hierarchy, the user actions of selecting and expanding nodes were often too complicated for the end user. In case that the end user doesn't know exactly where in the class hierarchy the desired concept is located, browsing involves a high error-rate in the exploration process when the explored classes and sub-classes do not contain the concept the end user looks for. As shown in the example above, the concept "rescue helicopter" has six upper classes. An end user needs to select and expand in this example six times the right node to finally select the concept "rescue helicopter" as resource in this class browser due to the direct OWL class hierarchy visualization. In sum, we found the simple browser visualization of an OWL class hierarchy as not sufficient for an end user interface.

In SoKNOS, we have addressed this challenge by hiding top level categories in the user interface. Only concepts defined in the core domain ontology are used in the user interface (but are still available to the reasoner); the foundational categories from DOLCE are not. Thus, the end user only works with concepts from her own domain. As a further extension, immediate categories that do not provide additional value for the end user, such as "motorized means of transportation", can be suppressed in the visualization.

Finding the right visualization. Various ways of visualizing ontologies and annotated data exist [18]. In the SoKNOS Semantic Data Explorer discussed above, we have used a straight forward graph view, which, like the standard OWL visualization, uses ellipses for instances and rectangles for data values. The user studies have shown that even that simple, straight forward solution provides a large benefit for the end user. Thus, slightly modifying Jim Hendler's famous quote [19], we can state that *a little visualization goes a long way.*

4 Conclusion

In this paper, we have introduced the SoKNOS system, a functional prototype for an integrated emergency management system which makes use of ontologies and semantic technologies for various purposes.

In SoKNOS, ontologies have been used both at design-time and at run-time of the system. Ontologies are used for providing a mutual understanding between developers and end users as well as between end users from different organizations. By annotating information objects and data sources, information retrieval, the discovery of relevant Web services and the integration of different databases containing necessary information, are simplified and partly automated. Furthermore, ontologies are used for improving the interaction with the system by facilitating user actions across application borders, and by providing plausibility checks for avoiding mistakes due to stressful situations.

During the course of the project, we have employed ontologies and semantic technologies in various settings, and derived several key lessons learned. First, the ontology engineering process necessarily should involve end users from the very beginning and foresee the role of dedicated ontology engineers, since ontology engineering is a non-trivial task which is significantly different from software engineering, so it cannot be simply overtaken by a software engineer. Tool support is currently not sufficient for letting untrained users build a useful ontology.

Second, current semantic annotation mechanisms for class models are not suitable. Those mechanisms are most often intrusive and require a 1:1 mapping between the class model and the ontology. When dealing with legacy code, both assumptions are unrealistic. Thus, different mechanisms for semantically annotating class models are needed. Furthermore, relying on a programming model backed by an ontology and using reasoning at run-time imposes significant challenges to reactivity and performance.

Third, it is not trivial to find an appropriate modeling and visualization depth for ontologies. While a large modeling depth is useful for some tasks, the feedback from the end users targeted at the need for simpler visualizations. In SoKNOS, we have addressed that need by reducing the complexity of the visualization, and by providing a straight forward, but very useful graphical visualization of the annotated data.

In summary, we have shown a number of use cases which demonstrate how the employment of ontologies and semantic technologies can make emergency management systems more useful and versatile. The lessons learned can also be transferred to projects with similar requirements in other domains.

Acknowledgements

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grant no. 01ISO7009.

References

1. Paulheim, H., Döweling, S., Tso-Sutter, K., Probst, F., Ziegert, T.: Improving Usability of Integrated Emergency Response Systems: The SoKNOS Approach. In: Proceedings of 39 Jahrestagung der Gesellschaft für Informatik e.V (GI) - Informatik 2009 LNI, vol. 154, pp. 1435–1449 (2009)
2. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: WonderWeb Deliverable D18 – Ontology Library (final) (2003), `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf` (accessed August 2, 2010)
3. Döweling, S., Probst, F., Ziegert, T., Manske, K.: SoKNOS - An Interactive Visual Emergency Management Framework. In: Amicis, R.D., Stojanovic, R., Conti, G. (eds.) GeoSpatial Visual Analytics. NATO Science for Peace and Security Series C: Environmental Security, pp. 251–262. Springer, Heidelberg (2009)
4. Paulheim, H., Probst, F.: Application Integration on the User Interface Level: an Ontology-Based Approach. Data & Knowledge Engineering Journal 69(11), 1103–1116 (2010)

5. Paulheim, H.: Efficient semantic event processing: Lessons learned in user interface integration. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 60–74. Springer, Heidelberg (2010)
6. Angele, J., Erdmann, M., Wenke, D.: Ontology-Based Knowledge Management in Automotive Engineering Scenarios. In: Hepp, M., Leenheer, P.D., Moor, A.D., Sure, Y. (eds.) Ontology Management. Semantic Web and Beyond, vol. 7, pp. 245–264. Springer, Heidelberg (2008)
7. Angele, J., Lausen, G.: Handbook on Ontologies. In: Staab, S., Studer, R. (eds.) International Handbooks on Information Systems, 2nd edn., pp. 45–70. Springer, Heidelberg (2009)
8. Voigt, K., Ivanov, P., Rummler, A.: MatchBox: Combined Meta-model Matching for Semi-automatic Mapping Generation. In: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 2281–2288. ACM, New York (2010)
9. Sonntag, D., Deru, M., Bergweiler, S.: Design and Implementation of Combined Mobile and Touchscreen-based Multimodal Web 3.0 Interfaces. In: Arabnia, H.R., de la Fuente, D., Olivas, J.A. (eds.) Proceedings of the 2009 International Conference on Artificial Intelligence (ICAI 2009), pp. 974–979. CSREA Press (2009)
10. Babitski, G., Bergweiler, S., Hoffmann, J., Schön, D., Stasch, C., Walkowski, A.C.: Ontology-based integration of sensor web services in disaster management. In: Janowicz, K., Raubal, M., Levashkin, S. (eds.) GeoS 2009. LNCS, vol. 5892, pp. 103–121. Springer, Heidelberg (2009)
11. Liu, B., Chen, H., He, W.: Deriving User Interface from Ontologies: A Model-Based Approach. In: ICTAI 2005: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, pp. 254–259. IEEE Computer Society, Washington, DC, USA (2005)
12. 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)
13. Paulheim, H., Meyer, L.: Ontology-based Information Visualization in Integrated UIs. In: Proceedings of the 2011 International Conference on Intelligent User Interfaces (IUI), pp. 451–452. ACM, New York (2011)
14. García-Barriocanal, E., Sicilia, M.A., Sánchez-Alonso, S.: Usability evaluation of ontology editors. Knowledge Organization 32(1), 1–9 (2005)
15. Babitski, G., Probst, F., Hoffmann, J., Oberle, D.: Ontology Design for Information Integration in Catastrophy Management. In: Proceedings of the 4th International Workshop on Applications of Semantic Technologies, AST 2009 (2009)
16. Paulheim, H., Plendl, R., Probst, F., Oberle, D.: Mapping Pragmatic Class Models to Reference Ontologies. In: 2nd International Workshop on Data Engineering meets the Semantic Web, DESWeb (2011)
17. Hepp, M.: Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies. IEEE Internet Computing 11(1), 90–96 (2007)
18. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.G.: Ontology Visualization Methods - A Survey. ACM Comput. Surv. 39(4) (2007)
19. Hendler, J.: On Beyond Ontology (2003), http://iswc2003.semanticweb.org/hendler_files/v3_document.htm; Invited Talk at the International Semantic Web Conference (2003)

Semantic Technologies for Describing Measurement Data in Databases

Ulf Noyer, Dirk Beckmann, and Frank Köster

Institute of Transportation Systems, German Aerospace Center
{ulf.noyer,dirk.beckmann,frank.koester}@dlr.de

Abstract. Exploration and analysis of vast empirical data is a cornerstone of the development and assessment of driver assistance systems. A common challenge is to apply the domain specific knowledge to the (mechanised) data handling, pre-processing and analysis process.

Ontologies can describe domain specific knowledge in a structured way that is manageable for both humans and algorithms. This paper outlines an architecture to support an ontology based analysis process for data stored in databases. Build on these concepts and architecture, a prototype that handles semantic data annotations is presented. Finally, the concept is demonstrated in a realistic example. The usage of exchangeable ontologies generally allows the adaption of presented methods for different domains.

1 Introduction

Reliable and safe automation is one foundation for modern traffic systems and part of concepts for assistance and automation systems. Therefore, for the analysis and reflection of users in automated environments, experimental systems (i.e. vehicles and driving simulators) play an important role. They produce exhaustive amounts of data, which can be used for an evaluation of the considered system in a realistic environment. Long term studies and naturalistic driving studies [7] result in similar datasets. Much data means a lot of information to interpret and potential results to discover. Therefore our motivation is, to store derived meta data closely connected with its original data for an integrated processing. By using semantic technologies a continuous technical and semantic process can be provided.

As a starting point, experimental data is to be considered as already stored in the relational database and should not be changed in the process. As many of the following principles not only match for measurement data, but in general for tables in relational databases, also the term bulk data is used as an equivalent for measurement data. So, it is aspired, to use semantic technologies for describing the database elements to support their interpretation [15]. They allow to formally handle complex data structures very flexible and schema knowledge can be extended easily. This is a demand resulting from the fact, that experimental systems are in continuous development and projects touch different knowledge domains.

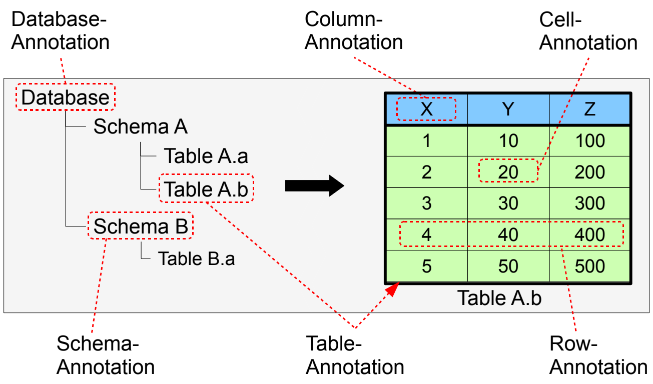


Fig. 1. Tables of a relational database with possible annotations

Figure 1 shows the intended database elements for semantic annotations. Tables containing sensor data of an experiment can be tagged with experiment information. Columns can have a semantic annotation about physical unit of containing elements and used sensor for recording. Rows or cells can get an annotation about bad quality or detected events in the considered elements. A complex application case is presented in Sect. 5. More meaningful examples of generic metadata annotations are also discussed in [16,12].

In some of the following considerations it is assumed, that table records have an order, so the statement *row* is preferred to record or tuple. Also rows are considered with the restriction that they must be uniquely identifiably. The needed database elements from Fig. 1 are transformed in unique URIs to be used in RDF statements. However, data itself is not transformed, since the value itself is not used for the presented purposes. Based on this, annotations are used similar to a formal memo or notepad mechanism. As domain knowledge is solely modelled in exchangeable ontologies, the presented concepts can be also applied for other domains.

2 Background and Related Work

This section introduces other work, which is needed for this paper or is relevant.

2.1 Usage of Semantic Technologies

In the last years several metadata technologies evolved, with the aim to make them better processable by machines. Especially the Resource Description Framework (RDF) is important in this context. Using RDF, a graph with relations of the individual metadata elements can be modelled. Resources to be described in RDF get annotated using triples, consisting of a subject, predicate and an object. On top of this, statements can be flexibly constructed. For serializing i.e. RDF/XML or Turtle syntax can be used.

The Web Ontology Language (OWL) is based on principles of RDF and can be used to formally describe complete ontologies. In an ontology the terms and their relations and rules are described. Especially for plain rules there is the Semantic Web Rule Language (SWRL) as an extension of OWL [13].

2.2 RDF-Stores

A RDF-repository [18,1] (triple-store) is used to manage RDF-statements. It supports adding, manipulating and searching statements in the repository. Such repositories can have different backends to store managed statements. Using relational databases as backend is in focus of this document. The statements consisting of subject, predicate and object are stored in a separate relation in the underlying database. Advantages of different optimization strategies are exhaustively discussed in [22]. Current repositories often directly support the Simple Protocol and RDF Query Language (SPARQL). It has for RDF-stores similar importance as SQL for relational databases.

2.3 Mapping of Relational Databases into RDF

There have been already early approaches to map relational data into RDF for the integration of relational databases and its contents [5]. In principle, a predefined scheme or rules for mapping the schema of the database and its content into RDF statements are used. There are projects ([3,6]), which are concerned with this functionality. However, both projects are not used for this work, because they would be strongly oversized. As shown in Sect. 3.3 just the mapping of database schema is needed (and not of data itself), what is comparatively easy. This work also only requires a mapping of the actually used database elements. If in the future also data itself managed in a database should be included in RDF mapping, usage of [6] could be a valuable option. This work is based on the principles discussed in [5].

2.4 Scientific Workflow Systems

The presented paper is distantly related with the field of scientific workflow systems (SWS, i.e. [2,20]). In SWS a process for evaluation of a data product can be specified. At execution, each processing step runs consecutively on data. The result of one step is input of the next one. Often SWS offer a graphical interface for composition of workflow steps using directed graphs. These systems support processing by offering metadata about processing steps and data lineage. Exchanged data products are considered as black boxes by SWS, which are not interpreted by the system.

But that is the focus of the presented paper, to concentrate on the content level of exchanged data. For that reason micro-annotations are utilized in a similar way as they are used with web pages [9] or multimedia files [11]. The aspect of data flow and its composition has only a minor role. In this work it is assumed that time structured data is stored in a database, an unusual restriction for classical SWS.

3 Data Management

The recorded time structured bulk data is stored in a relational database. All other data either describes these time series or is deduced from it. As a result, all data besides time series is considered as metadata.

3.1 Use Cases

This section presents the identified use cases for semantic annotations.

– Manual examination: Using an interface for interaction with annotations a user can directly interact and manipulate them (cf. Sect. 4). Resources can be referenced for documentation purposes. It also allows exporting tables with annotations into a flat file format for usage in other applications.

– Services: A service is an automated piece of software, which analyses some given data/annotations in the database and creates new data/annotations by applying specific domain knowledge. It has a web service interface for being externally triggered. Web services can be easily reused and orchestrated for more complex workflow setups or automatically started, after new data has been gathered. Furthermore, web services can be also reused by many SWS.

– Application integration: Any arbitrary application can also integrate support for semantic annotations. *Ærogator* is used for explorative data analysis and can display annotations parallel to time series data [12]. For data mining in time series *EAMole* is used and could respect annotations under data mining aspects [12].

3.2 Common Architecture

An overview of data storage and access are shown in Fig. 2 [14]. In the bottom layer, there is a relational database with the internal name Dominion-DataStore. It stores relational bulk data and RDF annotations in completely separated data structures. All elements in the figure with a horizontal pattern are related to bulk data. Elements with a vertical pattern describe elements connected with semantic annotations. Measurement data from an experiment is always stored in a new table, so relational database schema is quite simple. In the database there are *stored procedures* available to directly work on RDF annotations with SQL.

Access to RDF statements is more abstracted for an efficient data handling. The *triple store abstraction* implements the concrete RDF storage. The native RDF handling of an Oracle database is used. An alternative would be a database product independent schema layout (cf. Sect. 2.2). On top of the RDF storage abstraction a common *semantic web framework* handles RDF graphs [18]. Depending on the concrete RDF-graph, also RDFS- or OWL- models can be handled. Another layer is responsible for mapping required database elements into resource URIs, to be referenced in RDF statements. Concrete mapping implementation is based on the principles discussed in Sect. 3.3. The RDF store allows managing strictly divided models. So for each new bulk data table a new RDF

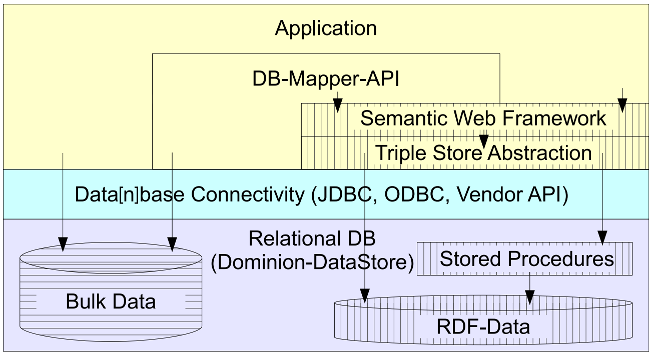


Fig. 2. Overview of architecture used for data access

model is used. In this way a good scalability can be reached. Generic information is stored in a jointly used RDF model.

Strict separation of bulk data and annotations also reflects their different usage. Afterwards measurement data is not modified anymore. Its meaning is hidden in recorded number values to be interpreted. On top of the bulk data, there is an abstract layer of interpretations. Descriptions (i.e. for columns) or identified events (i.e. overtaking manoeuvre) from human imaginations are to be seen there. In contrast to bulk data, there are much less elements. This abstraction layer is used during analysis, so that there is a high dynamic in the statements. These statements are usually assertional (ABox). Schema knowledge (TBox) can be edited using Protégé [19] or a Semantic Wiki [17] as known for RDFS/OWL and then be integrated (cf. Sect. 4). The just presented approach is the adaption of commonly discussed multimedia annotations [9,11] for measurement data. The term micro-annotations is also adapted from this domain.

3.3 Annotation of Relational Data

In the database, bulk data is organized in relations. RDF-annotations can reference single attribute instances, tuples, attributes, relations, schemas or databases (cf. Fig. 1). Theoretically, same mechanisms could be used to annotate other relational database concepts (i.e. keys, constraints). However, those are not considered, since they are not of interest for discussed use cases. For that purpose, an integration of relational elements in RDF is needed (cf. Sect. 2.3). Since RDF requires an unique specifier for every resource, database elements must be mapped into URIs. The used mapping approach builds on principles of [5].

Relations are uniquely identified by their name *table_name*. Also attributes of a table have an unique name *column_name*. For simplicity, slash (/) is not allowed in *table_name* and *column_name*. To identify tuples of a relation, the considered relation must have an *unique key*. Currently, an attribute *row_id* is added to every relation, which contains unique integers in context of a relation and serves as a surrogate key. By combining the identification of a tuple and an attribute, single attribute instances can be referenced. Based on those assumptions, Table 1 shows building instructions for URIs to identify different database elements.

Table 1. Transformation rules for a database element into an URI

DB element	Transformation into URI
Relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*
Attribute of a relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*/ ↪ column/*column_name*
Tuple of a relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*/row/*row_id*
Attribute instance of a relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*/ ↪ column/*column_name*/ ↪ row/*row_id*

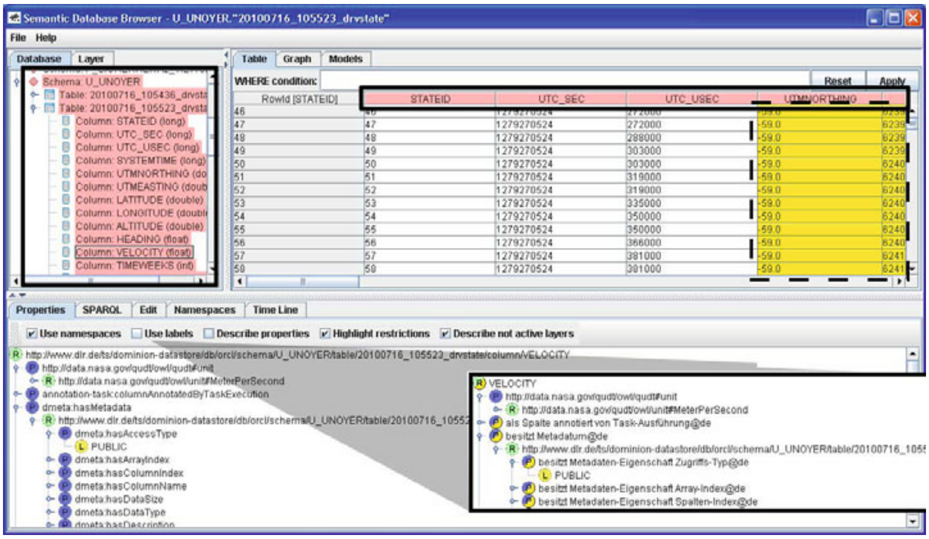
An example of an URI, that references an attribute instance of attribute VELOCITY with a unique key 48 in relation `20100716_105523_drvstate`, would have the following form: `http://www.dlr.de/ts/dominion-datastore/table/20100716_105523_drvstate/column/VELOCITY/row/48`

In the same way, before table name the name of the database instance and the schema name are also included (Fig. 3). The inversion is well-defined and therefore URIs can be assigned to their original database elements. Described approach does not integrate values of relational bulk data itself into RDF, since they are not required (cf. Sect. 2.3).

4 User Interface

The graphical user interface allows to visualize semantic annotations. During sequent processing steps it helps to understand temporary results. Fig. 3 shows a screenshot of the application called *Semantic Database Browser*. In the left window a schema, table or column can be chosen. The main window shows selected tables. Database elements, which have annotations, are highlighted with a colour. When such an annotated element is selected, the bottom window shows the annotations as a tree. In Fig. 3 the column VELOCITY is selected in the left window.

The tree representation was selected, as it is very compact, especially compared to graph visualizations. Since only the really needed sub-tree is fetched from memory model, this approach preserves resources. However, it must be considered, that only outgoing edges of the graph model are drawn. If there are cycles in the graph and the tree is expanded, nodes can appear multiple times. In front of each tree element is a symbol, which indicates the type of the node


Semantic Database Browser - U_UNOYER."20100716_105523_drvstate"
File Help
Database Layer
Table Graph Models
WHERE condition:
Reset Apply
Properties SPARQL Edit Namespaces Time Line
[✓] Use namespaces [ ] Use labels [ ] Describe properties [✓] Highlight restrictions [✓] Describe not active layers

Schema: U_UNOYER
Table: 20100716_105436_drvsta
Table: 20100716_105523_drvsta
Column: STATEID (long)
Column: UTC_SEC (long)
Column: UTC_USEC (long)
Column: SYSTEMTIME (long)
Column: UTMNORTHING (do
Column: UTMEASTING (doub
Column: LATITUDE (double)
Column: LONGITUDE (doubl
Column: ALTITUDE (double)
Column: HEADING (float)
Column: VELOCITY (float)
Column: TIMEWEEKS (int)

RowId [STATEID]	STATEID	UTC_SEC	UTC_USEC	UTMNORTHING	
46	46	1279270524	272000	-59.0	6239
47	47	1279270524	272000	-59.0	6239
48	48	1279270524	288000	-59.0	6239
49	49	1279270524	303000	-59.0	6239
50	50	1279270524	303000	-59.0	6240
51	51	1279270524	319000	-59.0	6240
52	52	1279270524	319000	-59.0	6240
53	53	1279270524	335000	-59.0	6240
54	54	1279270524	350000	-59.0	6240
55	55	1279270524	350000	-59.0	6240
56	56	1279270524	366000	-59.0	6240
57	57	1279270524	381000	-59.0	6241
58	58	1279270524	381000	-59.0	6241

```
R http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/U_UNOYER/table/20100716_105523_drvstate/column/VELOCITY
  P http://data.nasa.gov/qudt/owl/qudt#unit
    R http://data.nasa.gov/qudt/owl/unit#MeterPerSecond
  P annotation-task:columnAnnotatedByTaskExecution
  P dmeta:hasMetadata
    R http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/U_UNOYER/table/20100716_10552
      P dmeta:hasAccessType
        L PUBLIC
      P dmeta:hasArrayIndex
      P dmeta:hasColumnIndex
      P dmeta:hasColumnName
      P dmeta:hasDataSize
      P dmeta:hasDataType
      P dmeta:hasDescription
```

```
R VELOCITY
  P http://data.nasa.gov/qudt/owl/qudt#unit
    R http://data.nasa.gov/qudt/owl/unit#MeterPerSecond
  P als Spalte annotiert von Task-Ausführung@de
  P besitzt Metadatum@de
    R http://www.dlr.de/ts/dominion-datastore/db/orcl/schema/U_UNOYER/table/20100716_105
      P besitzt Metadaten-Eigenschaft Zugriffs-Typ@de
        L PUBLIC
      P besitzt Metadaten-Eigenschaft Array-Index@de
      P besitzt Metadaten-Eigenschaft Spalten-Index@de
```

Fig. 3. User interface for interactive working with annotations including highlighted elements and a montage showing nationalized labels

(named resource, blank node, property or literal). The option *describe properties* lets the tree also show properties of properties. In this way i.e. domain- and range-restrictions of properties can be viewed. Another feature is to optionally show labels (identified by `rdfs:label` [10]) instead of resource names (URIs or anonymous ids). In this case a literal is chosen, which best matches the language chosen in the user interface (montage Fig. 3). Using the context menu it is also possible to directly open URI-resources with the associated program, i.e. a web-page in a semantic wiki with the web browser.

Since annotations are stored in different models for each bulk data table (cf. Sect. 3.2), they are loaded on demand for the currently shown table. For that reason interactions with annotations in the interface are generally performed on the base graph model G_B. This graph is the union of the actually used graphs. That at least includes the generic graph G_G, containing annotations for database instance, schemas and table schema. Annotations on bulk data, which are always stored in new tables, are managed in separate RDF models. For each table T_i there exists a graph model G_{T_i}. Graph G_B is manipulated during runtime and is the union of generic annotations G_G and currently viewed table T_j: $G_B = G_G \cup G_{T_\text{j}}$. Since G_G only growths very moderately and most annotations are managed in the different G_{T_i}, the scalability of the approach for many experiments with their measurement tables is assured. If one table can be processed, this is also possible for a great many of them.

It can be necessary to reference external knowledgebases G_{OWL_i} in form of OWL documents. Therefor $\bigcup_{i \in I} G_{OWL_i}$ with $i \in I$ is the set of relevant OWL documents. As consequence G_{OWL_i} can be added as a sub-graph to G_B, with the

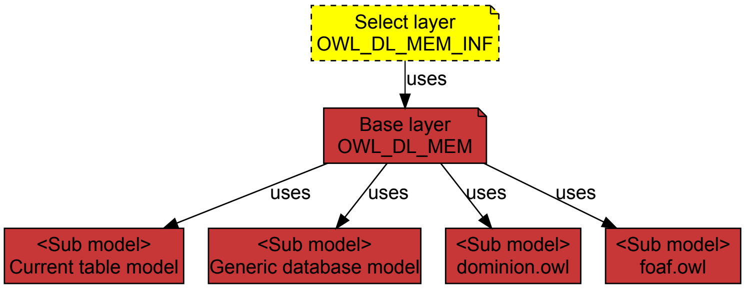


Fig. 4. Relations of a layer with the base graph and sub-graphs

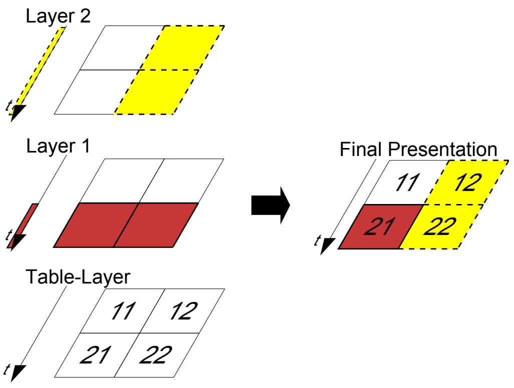


Fig. 5. Overlay of layers for the presentation of database elements

result $G_B = G_G \cup G_{T_j} \cup \bigcup_{i \in I} G_{OWL_i}$. In the case of measurement data handling an additional OWL document primarily contains additional schema knowledge, as assertional knowledge concerning database elements is maintained in G_{T_i}.

The resulting graph of graphs is shown in Fig. 4 in the bottom half, where subgraphs are labelled with full text names. This graphical presentation is available in the main window accessible by a tab. The user can interact with the graph, as he can remove sub-graphs or request some statistics about them. For further support of visualization *layers* are offered, which are configured by the user in the left window (accessed by a tab). A layer is either the graph G_B as complete union of the included knowledge bases (base layer) or a view derived from G_B with an additional layer name and colour (Fig. 4 in the upper half).

If a database element is contained in a layer, it is highlighted in the specific colour of the layer. Furthermore, layers have an order defined by the user. An element contained in several layers is shown in the colour of the layer with the greatest index (Fig. 5). In this way, a layer concept is realized similar to be found in geo information systems. Derived graphs for layers are created using a reasoner or SPARQL queries (CONSTRUCT-, DESCRIBE- or SELECT) on the base layer. The result set of a Select-query is handled as a graph consisting only of nodes and without edges. In Fig. 3 the base layer with all annotated database elements is shown in the background and results in a (red) colouring of schemas, tables and columns (highlighted by two solid boxes). Cells in the Utmnorthing-column are contained in a layer based on a Select-query and are therefore highlighted (yellow, surrounded by a dashed box).

Other features in the user interface cover manually editing annotations and management of namespaces. By using SPARQL individual layers can be searched. The corresponding graph of a selected resource and neighbouring nodes in a given radius can be visualized. A in this way displayed graph also reflects the common colouring of the different layers.

For another kind of annotation presentation the ordering of database rows is necessary. The user interface therefore uses the natural order of the unique key for the tables (cf. Sect. 3.3). In case of measurement data, rows are ordered by time. This kind of presentation is also in terms of colour indicated in Fig. 5 for layers 1 and 2 along t-axis. At this, for every layer row- and cell-annotations are projected on t-axis. As a result, for each layer a kind of bar chart is drawn, to visualize in which intervals annotations are existent (Fig. 6). Since layers can be freely chosen, visual comparisons are possible.

The visual interface, which is described in this section, is in a prototypic but fully operational state.

5 Example Application Cases

This section presents two usage examples for the previously introduced annotations for measurement data.

5.1 Data Quality Management

Data quality has an essential impact for the considered analysis process. An automated service checks recorded measurement data against its specification. The specification is provided by the Dominion vehicular middleware [8] during experiments and is stored as annotations in the Dominion-DataStore. Fig. 3 partially shows these specifications as annotations in namespace `dmeta`.

As result of the quality checks, there are (micro-)annotations generated exactly for those cells, which violate their specification. Those cells are shown in the same figure in the Utmnorthing column and highlighted by a separate layer (yellow). These annotations are linking the violated specifications, to allow searching and presenting different kinds of violations.

5.2 Description of Manoeuvres

The second application case is inferred from the IMoST project [4]. This project focuses on merging into traffic on highways. Basis for analysis are test drives from driving studies of DLR.

An annotation service is triggered to analyse measurement data of a test drive and extract relevant information. In this way (micro-)annotations *blink left*, *near centreline* and *vehicle ahead lost* are created for complete rows. These annotations themselves are divided in the categories *action* and *environmental event*. Annotation *blink left* is always generated, when the appropriate measurement variable shows a 1 to indicate, that the direction indicator has been activated. If the variable for the distance of the vehicle to the centreline is less than 0.3 meters, the annotation *near centreline* is created. When the distance detection to a leading vehicle suddenly looses the vehicle ahead, the service creates the *vehicle ahead lost* annotation. In this case it can be expected, that either the own or the foreign vehicle has performed a lane change. Generally, automated services are just one possibility to create annotations. Alternatives are the manually creation using a video rating or completely automated approaches like data mining.

In this way, just few of the rows get annotations, since most time nothing important happens (depending on the experiment scenario). So described annotations are used as a memo or notepad mechanism, to tag relevant database elements with information for further processing. Fig. 6 shows a time line representation (cf. Sect. 4) for selected events of an experiment. In this picture *blink left* and *near centreline* are layers containing corresponding annotations, which are selected by a SPARQL-query from base layer.

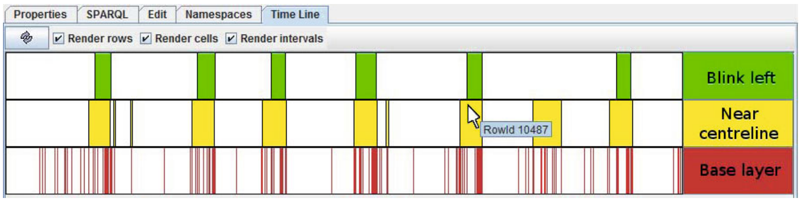


Fig. 6. Time line representation from user interface for the base layer and two layers containing selected events

In the following steps of semantic enrichment every analysis step builds on the previous ones. Thus, in every step the point of view gets more abstract and more comprehensible for domain specialists. Below, the concept *overtaking* is defined. An overtaking manoeuvre occurs, if the events *blink left*, *near centreline* and *vehicle ahead lost* arise at the same time. Based on this, an analyst can search for all these events in an analysis tool and then directly visualize the searched situation. Technically this search is implemented using SPARQL-queries. In the formerly presented user interface PREFIX statements for registered namespaces are optional:

```
PREFIX ts: <http://www.dlr.de/ts/>
SELECT ?x WHERE {
  ?x ts:hasProperty ts:BlinkLeft .
  ?x ts:hasProperty ts:NearCentreline .
  ?x ts:hasProperty ts:VehicleAheadLost .
}
```

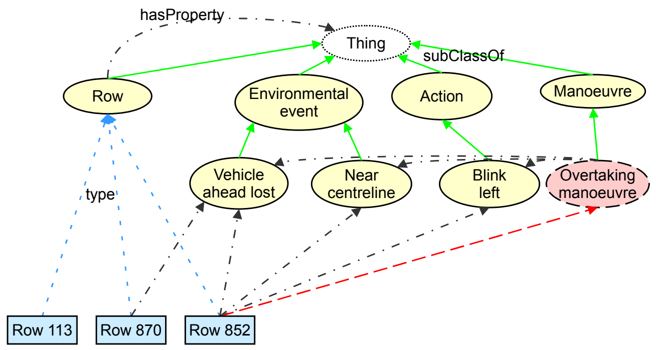


Fig. 7. Extract from the ontology to illustrate concepts of inheritance hierarchy

By using an ontology the concepts and their relationships are systemized. Fig. 7 shows an ontology, which covers concepts for events introduced in this section. Circles mark classes, which are organized in an inheritance hierarchy (green, solid arrows). The rectangular subjects are instances of type *row*. There are only three tuples (row identifier 113, 852 and 870) shown in this simplification. Black arrows with alternating dashes and dots as pattern mark rows with identified events.

Beside introduced sub-concepts in practice there can be much more (actions: i.e. brake, steer; environmental events: i.e. near to other vehicle, off the road). On base of these basic concepts, the concept class *manoeuvre* is introduced. Concrete actions and environmental events can form more complex manoeuvres (cf. [24], i.e. overtaking, turn off) and a more abstract view on street incidents is created. In the shown ontology, there is the newly introduced concept *overtaking manoeuvre* as a subclass of manoeuvre. Of course an algorithm in a service could identify the tuples, where the defined conditions for an overtaking manoeuvre are met, and annotate them. Alternatively, concept overtaking can be modelled in the ontology as an intersection of restrictions on necessary conditions. In Fig. 7 this is illustrated by the three arrows with alternating dashes and dots as pattern and overtaking as a subject. Modelling of restrictions is not shown in the figure. By using this modelling approach a reasoner can be applied accordingly. Its inference mechanisms deduce, in case that all conditions are complied to, an overtaking manoeuvre is existent. In Fig. 7, that is indicated by a red, dashed arrow. Hence, analysts can now directly identify overtaking manoeuvres using SPARQL.

As domain experts decide to redefine the concept overtaking, just the ontology must be changed and results can be inferred. So ontology can serve as an important and central pool of certain expert knowledge. Furthermore, it can formalize used vocabulary and concepts to help avoiding ambiguities and redundancies. Another advantage is, that ontologies can be modularized in sub-ontologies, so experts of a very specific sub-domain can be responsible for a sub-ontology. From the knowledge management's point of view this is very helpful. Bulk data of time series is very static and remains unchanged, so it is stored in traditional, relational tables. But because metadata and domain knowledge are heavily evolving during analysis process and often have complicated structures, RDF is a suitable modelling approach for them. RDF storage isn't as efficient as relational storage, but as long as metadata is less in volume than bulk data, handling and storage is usually not a demanding task.

Modelling of temporal aspects in ontologies is still a question of research (i.e. [21,23]). But that doesn't matter for the presented application cases, since just aspects without temporal aspects are modelled in the ontology. Temporal aspects can be covered using other analysis techniques. In the presented case an algorithm identifies, where radar looses tracking of a vehicle ahead, and creates appropriate annotations.

Nevertheless, the use cases are currently under heavy development, to support current results in this area of interest. In future, continuous row annotations will be modelled using time points and intervals. In this way, number of annotations can be reduced a lot, by just annotating beginning and end of an interval for an occurred event or action. That feature was already used to create Fig. 6, since the two upper layers in the picture are rendered using just start and end points. SWRL will be used to define a finite state machine on top of these events and detect abstract manoeuvres. Advantage of SWRL rules is that they are much more expressive for reasoning purposes than plain ontologies. So, more complex reasoning can be performed on event relationships, what is challenging to cover just using conventional database technologies.

In the discussed use-case, created annotations are only used to search them using SPARQL. But annotations represent common results and should be also reused by other kinds of processing steps. In this way, working with more abstract concepts is iteratively introduced and analysts can work on a more semantic level.

6 Summary and Outlook

The presented article discusses an approach for data management of measurement data and semantic metadata to support an iterative data analysis process. Aim of this iterative process is to provide the results of each processing step as transparent as possible. The data in the data store can be flexibly annotated to support the continuous and iterative process of knowledge discovery, by using RDF annotations and micro-annotations for semantic enrichment incorporating ontologies.

The presented approach is a combination of advantages of classical relational databases with semantic annotations and ontologies for the analysis of sensor data. Using ontologies, concepts and vocabulary of the metadata can be systemized. For database elements annotations serve as a kind of memo or notepad mechanism, which help both analysts and algorithms processing the time series data. In this way table data can be arbitrarily linked and become a part of the Semantic Web. Also the user benefits of the possibility to use inferencing software. The presented user interface allows an intuitive workflow with introduced annotations. Semantic annotations are flexible and well suited for this application of knowledge integration. By storing time structured data using a relational database, it can be easily accessed using standard database connectivity of used applications without any adaptations.

On the contrary there are also challenges in the presented approach. Using two technologies instead of one for accessing specific data increases complexity. That also influences modelling of considered data. The developer must consider whether to store data in a relational way or as annotations. Moreover, the speed of accessing annotations can be slower than a solely relational storage. A RDF-store allows to handle separate data models for the annotation instances. So, separate models can be used for different time structured bulk data tables to ensure scalability. Generally we promote to use a relational design for bulk data, which has a fix schema. For light weighted data and metadata, which constantly evolves and where the interpretation is in focus, we prefer annotations.

The use cases and ontologies are currently refined with a stronger focus on temporal aspects. Furthermore, the integration of automated services in a SWS has to be realized. A further medium-term objective is to polish the user interface and port it to the Eclipse Rich Client Platform for a productive deployment.

References

1. Aduna-Software: OpenRDF.org — ... home of Sesame. WWW (October 2008), `http://www.openrdf.org/`
2. Altintas, I., et.al.: Kepler: An extensible system for design and execution of scientific workflows. In: Scientific and Statistical Database Management (2004)
3. Barrasa, J., Óscar Corcho, Gómez-Pérez, A.: R2O, an Extensible and Semantically Based Database-to-ontology Mapping Language. In: Workshop on Semantic Web and Databases (2004)
4. Baumann, M., et al.: Integrated modelling for safe transportation — driver modeling and driver experiments. In: 2te Fachtagung Fahrermodellierung (2008)
5. Berners-Lee, T.: Relational Databases on the Semantic Web (1998), `http://www.w3.org/DesignIssues/RDB-RDF.html`
6. Bizer, C.: D2R MAP — A Database to RDF Mapping Language. In: World Wide Web Conference (2003)
7. Fancher, P., et al.: Intelligent cruise control field operational test (final report). Tech. rep., University of Michigan (1998)
8. Gačnik, J., et al.: DESCAS — Design Process for the Development of Safety-Critical Advanced Driver Assistance Systems. In: FORMS (2008)
9. Gertz, M., Sattler, K.U.: Integrating scientific data through external, concept-based annotations. In: Data Integration over the Web (2002)
10. Hayes, P.: RDF Semantics. W3C Recommendation (February 2004)
11. Herrera, P., et al.: Mucosa: A music content semantic annotator. In: Music Information Retrieval (2005)
12. Köster, F.: Datenbasierte Kompetenz- und Verhaltensanalyse — Anwendungsbeispiele im selbstorganisierten eLearning. In: OlWIR (2007)

13. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 1, 41–60 (2005)
14. Noyer, U., Beckmann, D., Köster, F.: Semantic annotation of sensor data to support data analysis processes. In: Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM) (2009)
15. Noyer, U., Beckmann, D., Köster, F.: Semantic technologies and metadata systematisation for evaluating time series in the context of driving experiments. In: 11th International Protégé Conference, pp. 17 – 18 (2009)
16. Rothenberg, J.: Metadata to support data quality and longevity. In: Proceedings of the 1st IEEE Metadata Conference (1996)
17. Schaffert, S., Francois Bry, J.B., Kiesel, M.: Semantic wiki. Informatik-Spektrum 30, 434–439 (2007)
18. SourceForge.net. Jena — A Semantic Web Framework for Java (October 2008), `http://jena.sourceforge.net/`
19. Stanford Center for Biomedical Informatics Research: The protégé ontology editor and knowledge acquisition system. WWW (April 2009), `http://protege.stanford.edu/`
20. myGrid team. Taverna workbench project webseite (November 2009), `http://taverna.sourceforge.net/`
21. Tusch, G., Huang, X., O'Connor, M., Das, A.: Exploring microarray time series with Protégé. In: Protégé Conference (2009)
22. Velegrakis, Y.: Relational Technologies, Metadata and RDF, ch. 4, pp. 41–66. Springer, Heidelberg (2010)
23. Virgilio, R.D., Giunchiglia, F., Tanca, L. (eds.): Semantic Web Information Management: A Model-Based Perspective, ch. 11, pp. 225–246. Springer, Heidelberg (2010)
24. Vollrath, M., et al.: Erkennung von Fahrmanövern als Indikator für die Belastung des Fahrers. In: Fahrer im 21. Jahrhundert (2005)

Ontology-Driven Guidance for Requirements Elicitation

Stefan Farfeleder[1], Thomas Moser[2], Andreas Krall[1], Tor Stålhane[3], Inah Omoronyia[4], and Herbert Zojer[5]

[1] Institute of Computer Languages, Vienna University of Technology
{stefanf,andi}@complang.tuwien.ac.at
[2] Christian Doppler Laboratory "Software Engineering Integration for Flexible Automation Systems", Vienna University of Technology
thomas.moser@tuwien.ac.at
[3] Department of Computer and Information Science, Norwegian University of Science and Technology
stalhane@idi.ntnu.no
[4] Irish Software Engineering Research Centre, University of Limerick
inah.omoronyia@lero.ie
[5] Infineon Technologies Austria AG
herbert.zojer@infineon.com

Abstract. Requirements managers aim at keeping their sets of requirements well-defined, consistent and up to date throughout a project's life cycle. Semantic web technologies have found many valuable applications in the field of requirements engineering, with most of them focusing on requirements analysis. However the usability of results originating from such requirements analyses strongly depends on the quality of the original requirements, which often are defined using natural language expressions without meaningful structures. In this work we present the prototypic implementation of a semantic guidance system used to assist requirements engineers with capturing requirements using a semi-formal representation. The semantic guidance system uses concepts, relations and axioms of a domain ontology to provide a list of suggestions the requirements engineer can build on to define requirements. The semantic guidance system is evaluated based on a domain ontology and a set of requirements from the aerospace domain. The evaluation results show that the semantic guidance system effectively supports requirements engineers in defining well-structured requirements.

Keywords: requirements elicitation, domain ontology, elicitation guidance, requirements engineering.

1 Introduction

A major goal of requirements engineering is to achieve a common understanding between all project stakeholders regarding the set of requirements. Modern IT projects are complex due to the high number and complexity of requirements, as well as due to geographically distributed project stakeholders with different backgrounds and terminologies. Therefore, adequate requirements management tools are a major contribution to address these challenges. Current requirements management tools typically work with a common requirements database, which can be accessed by all stakeholders to retrieve information on requirements content, state, and interdependencies.

Requirements management tools help project managers and requirements engineers to keep the overview on large amounts of requirements by supporting: (a) Requirements categorization by clustering requirements into user-defined subsets to help users find relevant requirements more quickly, e.g., by sorting and filtering attribute values; (b) Requirements conflict analysis (or consistency checking) by analyzing requirements from different stakeholders for symptoms of inconsistency, e.g., contradicting requirements; and (c) Requirements tracing by identifying dependencies between requirements and artifacts to support analyses for change impact and requirements coverage. Unfortunately, requirements management suffers from the following challenges and limitations:

– Incompleteness [5] of requirements categorization and conflict identification, in particular, when performed manually.
– High human effort for requirements categorization, conflict analysis and tracing, especially with a large number of requirements [5].
– Tracing on syntactic rather than on concept level: requirements are often traced on the syntactic level by explicitly linking requirements to each other. However, requirements engineers actually want to trace concepts, i.e., link requirements based on their meaning, which can be achieved only partially by information retrieval approaches like "keyword matching" [11][12].

The use of semantic technologies seems promising for addressing these challenges: Ontologies provide the means for describing the concepts of a domain and the relationships between these concepts in a way that allows automated reasoning [18]. Automated reasoning can support tasks such as requirements categorization, requirements conflict analysis, and requirements tracing. While these are very valuable efforts, we think what is missing here is additionally having a proactive and interactive guidance system that tries to improve requirements quality while actually eliciting requirements.

In this work we present the prototypic implementation of a semantic guidance system used to assist requirements engineers with capturing requirements using a semi-formal representation. Compared to the usual flow - write requirements, analyze requirements using the domain ontology, improve requirements - our approach directly uses the domain ontology knowledge in a single step. The semantic guidance system uses concepts, relations and axioms of domain ontologies to provide a list of suggestions the requirements engineer can build on to define requirements.

We evaluate the proposed semantic guidance system based on a domain ontology and a set of requirements from the aerospace domain. The evaluation results show that the semantic guidance system supports the requirements engineer in defining well-structured requirements. The tool managed to provide useful suggestions for filling missing parts of requirements in the majority of the cases (>85%).

This work is organized in the following way: Section 2 presents related work. Section 3 motivates our research; section 4 introduces our approach to ontology-based guidance. Section 5 presents an evaluation of the tool and section 6 concludes and gives ideas about future work.

2 Related Work

This section summarizes related work, going from the broad field of requirements engineering to the more specific areas of elicitation guidance and finally pattern-based requirements.

2.1 Requirements Engineering

Requirements Engineering is a discipline that deals with understanding, documenting, communicating and implementing customers' needs. Thus, insufficient understanding and management of requirements can be seen as the biggest cause of project failure. In order to improve this situation a systematic process to handle requirements is needed [8]. The main activities of a requirements engineering process can be defined as follows [15]:

- **Requirements Elicitation**. Requirements elicitation involves technical staff working with customers to find out about the application domain, the services the system should provide and the system's operational constraints. The goal is to gather raw requirements.
- **Requirements Analysis and Negotiation**. Requirements analysis and negotiation is an activity which aims to discover problems and conflicts with the requirements and reach agreement on changes to satisfy all system stakeholders (people that are affected by the proposed system). The final goal is to reach a common understanding of the requirements between all project participants.
- **Requirements Documentation and Validation**. The defined requirements, written down in a software requirements specification, are validated against criteria like correctness, completeness, consistency, verifiability, unambiguity, traceability, etc.
- **Requirements Management**. Requirements management consists of managing changes of requirements (keeping them consistent), e.g., by ensuring requirements traceability (identification of interdependencies between requirements, other requirements, and artifacts).

These four steps can be summarized as requirements development. In addition, requirements management is a supporting discipline to control all requirements and their changes during the development life cycle and to identify and resolve inconsistencies between the requirements and the project plan and work products. One important method of requirements management is requirements tracing. Traceability can be defined as the *degree to which a relationship between* *two or more products of the development process can be established* [1]. Gotel [7] and Watkins [21] describe why requirements tracing can help project managers in verification, cost reduction, accountability, change management, identification of conflicting requirements and consistency checking of models.

2.2 Elicitation Guidance

There are several approaches to guide users to specify requirements. PROPEL [3] is a tool that provides guidance to define property specifications which are expressed as finite-state automata. For the definition of a property the user is guided by a *question tree*, a hierarchical sequence of questions. There are separate scope trees for a property's behavior and its scope. Based on the answers, the tool chooses an appropriate property template. A team of medical personnel and computer scientists used the tool to formulate properties of medical guidelines.

Kitamura et al. [14] present a requirements elicitation tool that improves requirements quality by analysis. The tool analyzes natural language requirements and finds domain ontology entities occurring in the statements. According to these occurrences and their relations in the ontology, requirements are analyzed in terms of completeness, correctness, consistency and unambiguity. Suggestions are provided to the user in order to improve these metrics, e.g., if the tool finds that a domain concept is not defined in the requirements set, the suggestion would be to add a requirement about the missing concept.

REAS [6] is a tool that interactively detects imprecisions in natural language requirements. It consists of a spelling checker, a rule imposer, a lexical analyzer and a parser. Some of the rules enforced by the tool are writing short requirements, using active voice instead of passive and avoiding possessive pronouns (e.g., "it" and "its"). If the tool detects a violation of a rule, the requirements engineer is asked to improve the offending expression.

Compared to our own approach, PROPEL only deals with a rather specific kind of requirements. The other two approaches try to identify weaknesses after the requirements have been defined and do not actively propose wording while writing them.

2.3 Pattern-Based Requirements

Hull, Jackson and Dick [9] first used the term *boilerplate* to refer to a textual requirement template. A boilerplate consists of a sequence of attributes and fixed syntax elements. As an example, a common boilerplate is "$\langle$*system*$\rangle$ `shall` $\langle$*action*$\rangle$". In this boilerplate $\langle$*system*$\rangle$ and $\langle$*action*$\rangle$ are attributes and `shall` is a fixed syntax element. It is possible to combine several boilerplates by means of simple string concatenation. This allows keeping the number of required boilerplates low while at the same time having a high flexibility. During instantiation textual values are assigned to the attributes of the boilerplates; a boilerplate requirement is thus defined by its boilerplates and its attribute values. The authors did not propose a fixed list of boilerplates[1] but instead envisioned a flexible language that can be adapted or enriched when necessary.

Stålhane, Omoronyia and Reichenbach [20] extended boilerplates with a domain ontology by linking attribute values to ontology concepts. They adapted the requirements analyses introduced by Kaiya [13] to boilerplate requirements and added a new analysis called opacity. The requirement language used in this work is based on their combination of boilerplates and the domain ontology.

Ibrahim et al. [10] use boilerplate requirements in their work about requirements change management. They define a mapping from boilerplate attributes to software design artifacts (e.g., classes, attributes, operations) and add traceability links between requirements and artifacts accordingly. There are several other pattern based languages similar to boilerplates, e.g., requirements based on EBNF grammars [19]. Denger et al. [4] propose natural language patterns to specify requirements in the embedded systems domain. They include a metamodel for requirement statements and one for events and reactions which they use to check the completeness of the pattern language. Compared to boilerplate requirements, their patterns seem to be a bit less generic, e.g., some of the non-functional requirements used in our evaluation would be impossible to express.

Matsuo, Ogasawara and Ohnishi [16] use controlled natural language for requirements, basically restraining the way in which simple sentences can be composed to more complex ones. They use a frame model to store information about the domain. There are three kind of frames. The noun frame classifies a noun into one of several predefined categories. The case frame classifies verbs into operations and contains the noun types which are required for the operation. Finally the function frame represents a composition of several simple operations. The authors use these frames to parse requirements specifications, to organize them according to different viewpoints and to check requirements completeness. In contrast to domain ontologies, the frame-based approach seems to be harder to understand and to adapt by non-experts.

3 Research Issues

There have been presented several approaches to use ontologies to analyze requirements. These approaches try to measure quality aspects like completeness, correctness and consistency on a set of requirements. In [20] there is an analysis called *opacity* that basically checks if, for two concepts occurring in a requirement, there is a relation between them in the domain ontology. A conclusion of our review of this analysis was that, rather than first writing an incorrect requirement, then analyzing and improving it, a better approach would be to actually suggest the very same domain information which is used for the opacity analysis to the requirements engineer in the first place. There are two points to this idea:

[1] J. Dick maintains a list of suggestions at `http://freespace.virgin.net/gbjedi/books/re/boilerplates.htm` though.

- We want a system that automatically proposes at least parts of the requirements by using information originating from a domain ontology.
- We want a system that exploits relations and axioms of the domain ontology, i.e., a system that is more powerful than just a simple dictionary. The relations and axioms of the domain ontology represent agreed upon knowledge of stakeholders; by using them we hope to improve the precision of requirements.

We believe that a tool supporting these two points will increase efficiency by speeding up the process to get a high-quality requirements specification.

A semantic guidance system implementing these two points was added to our boilerplates elicitation tool (DODT). To test our assumption, we used the tool on requirements from the domain of the *Doors Management System* (DMS). The DMS is a use case developed by EADS for experimenting with and evaluating requirements engineering and modeling techniques within the CESAR project[2]. The DMS controls the doors of an airplane; its main objective is to lock the doors of an airplane while it is moving (in the air or on the ground) and to unlock them when the airplane is parked on the ground. The system consists of sensors that measure the state of the doors, actuators to lock and unlock the doors and computing elements that control the entire system.

4 Guidance for Boilerplate Requirements

This section presents our approach for guiding the requirement engineer using information of a domain ontology.

4.1 Boilerplate Requirements Elicitation

Figure 1 shows how requirements elicitation works using the boilerplates method. To specify a requirement, one or more boilerplates are chosen by the requirements engineer. The attribute values refer to entities in the domain ontology. During instantiation the attributes of the chosen boilerplates are set and a final boilerplate-based requirement is produced. The semantic guidance system affects the domain ontology, the attribute values and the instantiation. Its purpose is to suggest potential and suitable values for the attributes to the requirements engineer.

Table 1 lists the boilerplate attributes implemented by the tool. We reduced the number of attributes compared to the original suggestions in [9] and [20]. This was done in accordance with user wishes who were uncomfortable with the subtile differences between similar attributes and had problems deciding on which to use.

[2] http://cesarproject.eu/

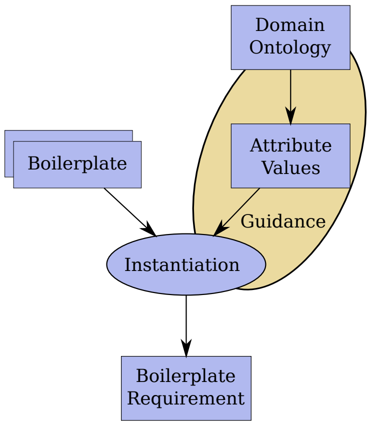


Fig. 1. Boilerplate Requirements Elicitation Flow

Table 1. Boilerplate Attributes and Values

Attribute	Description	Example Value
⟨*action*⟩	A behavior that is expected to be fulfilled by the system, or a capability	open the door
⟨*entity*⟩	A separate entity in the domain; not fitting into ⟨*user*⟩ or ⟨*system*⟩	door status
⟨*number*⟩	A numeric value denoting a quantity	100
⟨*operational condition*⟩	A condition or event that occurs during system operation	the user tries to open the door
⟨*system*⟩	Any part of the system; sub-class of entity	door
⟨*unit*⟩	Unit of measurement	millisecond
⟨*user*⟩	A person somehow interacting with the system, e.g., operating it; sub-class of entity	pilot

4.2 Domain Ontology

The domain ontology contains facts about the domain that are relevant to requirements engineering, i.e., facts that can be used to formulate and to analyze requirements. The domain ontology should be usable to specify requirements for several projects in the same domain. Thus adding concepts which are only relevant to a single product should be avoided. See section 6 for a discussion about combining several ontologies. This approach could be used to split the ontology into a common domain part and a product-specific one.

There are three kinds of facts about the domain stored in the domain ontology. The following list describes them. The tool uses an OWL[2] representation to store ontologies. A detailed mapping to OWL can be found in Table 2.

Table 2. Mapping of domain facts to OWL

Fact	OWL Expressions

Concept(name, definition)	Declaration(Class(*concept-iri*))
	AnnotationAssertion(rdfs:label *concept-iri name*)
	AnnotationAssertion(rdfs:comment *concept-iri definition*)

Relation(subj, label, obj)	Declaration(ObjectProperty(*label-iri*))
	AnnotationAssertion(rdfs:label *label-iri label*)
	SubClassOf(*subj-iri* ObjectAllValuesFrom(*label-iri obj-iri*))

SubClass(sub, super)	SubClassOf(*sub-iri super-iri*)
Equivalence(concept$_1$, concept$_2$)	EquivalentClasses(*concept$_1$-iri concept$_2$-iri*)
Deprecated(concept)	AnnotationAssertion(*deprecated-iri concept-iri* `1`)

Concept: A concept represents an entity in the problem domain. The entity can be material (e.g., a physical component of the system) or immaterial (e.g., a temporal state). OWL classes are used to represent concepts. The reason for using classes instead of individuals is the built-in support for sub-classing.

A concept has two attributes, its name and a textual definition. The definition is intended to provide the possibility to check whether the correct term is used.

Relation: A relation is a labeled directed connection between two concepts. A relation contains a label which is expected to be a verb. The label, the relation's source and destination concepts form a subject-verb-object triple. Relations are used for guidance (section 4.3). Relations map to OWL object properties and object property restrictions.

Axiom: There are two types of axioms that are relevant to guidance: sub-class and equivalence axioms. The first one specifies that one concept is a sub-class of another concept, e.g., *cargo door* is a sub-class of *door*. This information is used to "inherit" suggestions to sub-class concepts, e.g., the guidance system infers the suggestion *the user opens the cargo door* from the base class' *the user opens the door*.

The equivalence axiom is used to express that two concepts having different names refer to the same entity in the domain. An example from DMS is the equivalence of *aircraft* and *airplane*. Ideally each real-world phenomenon has exactly one name. However, due to requirements coming from different stakeholders or due to legacy reasons, at times several names are required. It is possible to mark a concept as being *deprecated*; the tool will warn about occurrences of such concepts and will suggest using an equivalent non-deprecated concept instead.

In this work we assume the pre-existence of a suitable domain ontology. See [17] for ways of constructing new domain ontologies. The tool contains an ontology editor that is tailored to the information described here. We found this editor to be more user-friendly than generic OWL editors like Protégé[3].

4.3 Guidance

When filling the attributes of a boilerplate, the tool provides a list of suggestions to the requirements engineer. The provided guidance depends on the attribute the requirements engineer is currently filling, e.g., the suggestions for $\langle system \rangle$ will be completely different than for $\langle action \rangle$. The idea is to apply an attribute-based pre-filter to avoid overwhelming the user with the complete list of ontology entities. Typing characters further filters this list of suggestions to only those entries matching the typed string.

It is not mandatory to choose from the list of suggestions; the tool will not stop the requirements engineer from entering something completely different. In case information is missing from the domain ontology, an update of the ontology should be performed to improve the guidance for similar requirements. All changes to the domain ontology should be validated by a domain expert to ensure data correctness.

There are three types of suggestions for an attribute; Table 3 provides an overview over the suggestion types.

Concept: The tool suggests to use the name of a concept for an attribute. The tool generates two variants, just the plain name and once prefixed with the article "the". The idea is that most of the times using "the" will be appropriate but sometimes other determiners like "all" or "each" are more suitable and are typed in manually.

Verb-Object: The tool uses a relation from the domain ontology to suggest a verb phrase to the requirements engineer. The suggestion is the concatenation of the verb's infinitive form[4], the word "the" and the relation's destination object. This construction is chosen in order to be grammatically correct following a modal verb like "shall". An example from Figure 2 is the suggestion *check the door status*.

Subject-Verb-Object: For this kind of suggestion the entire relation including subject and object is taken into account. The suggestion text is "the", the subject, the verb conjugated into third person singular form, "the" and the object. An example from Figure 2 is the suggestion *the person tries to open the door*.

It is possible to combine several suggestions simply by selecting the first one, manually typing in "and" and selecting another suggestion.

For the classification of concepts into different attributes a separate ontology, the *attributes ontology*, is used. The attributes ontology contains an OWL class

[3] `http://protege.stanford.edu/`

[4] For building verb infinitives the *morphological analyzer* of the GATE project (`http://gate.ac.uk/`) is used.

Table 3. Suggestion Types

Type	**Suggestion**
Concept	*concept*
	`the` *concept*
Verb-Object	*verb (inf.)* `the` *object*
Subject-Verb-Object	`the` *subject verb (3rd sing.)* `the` *object*

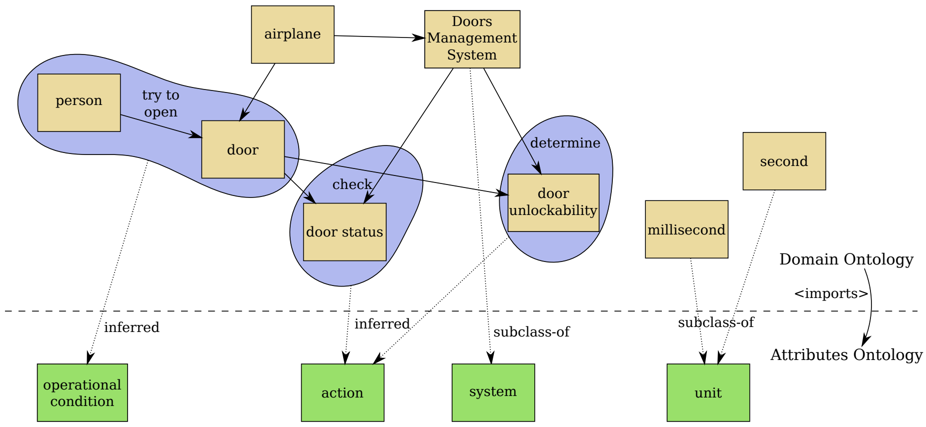


Fig. 2. Domain Ontology and Attributes Ontology

per attribute and the sub-class axioms mentioned in Table 1. The domain ontology imports the attributes ontology to use its classes. Domain concepts are linked to attributes by means of sub-class axioms which are stored in the domain ontology.

An example for the semantic guidance system is given in Figure 2. The domain ontology is shown in the upper part of the figure, the attributes ontology is below. The concept *Doors Management System* is a sub-class of class *system*, which in turn allows the tool to suggest using the *Doors Management System* for a boilerplate containing the attribute ⟨*system*⟩. The blue regions represent verb-object and subject-verb-object suggestions in the domain ontology. Their mapping to the attributes ⟨*action*⟩ and ⟨*operational condition*⟩ is inferred automatically by the tool.

Figure 3 shows the boilerplates for two requirements and some of the suggestions provided by the guidance system. The information that *Doors Management System* is a *system* and that *second* and *millisecond* are values for attribute *unit* is stored in the domain ontology itself. The suggestions *check the door status* and *determine the door unlockability* are inferred from the domain ontology relations. The knowledge to suggest verb-object pairs for the attribute *action* is a built-in feature of the tool. The attribute *operational condition* turns out to be the most difficult one in terms of providing useful suggestions. The reason for this is that

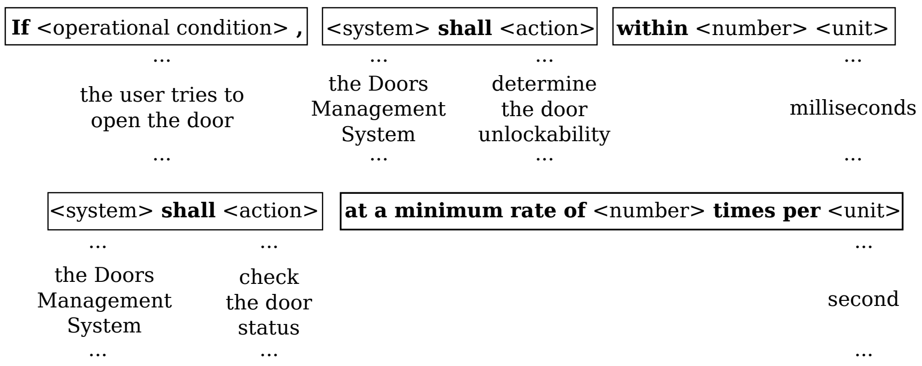

If <operational condition> ,
<system> shall <action>
within <number> <unit>
...
the user tries to open the door
the Doors Management System
determine the door unlockability
milliseconds
<system> shall <action>
at a minimum rate of <number> times per <unit>
the Doors Management System
check the door status
second

Fig. 3. Boilerplates and Suggestions

there are many grammatical ways to describe conditions, a subject-verb-object triple being only one of them. Therefore the tool does not only suggest those triples for conditions; instead all concepts, verb-object pairs and subject-verb-object triples are provided in order to use those phrases in conditions.

5 Evaluation

As mentioned before we evaluated the semantic guidance system with a domain ontology and a set of requirements from the Doors Management System.

5.1 Setting

The use case contains a set of 43 requirements specified using natural language text. Various types of requirements are included: functional, safety, performance, reliability, availability and cost. Each requirement was reformulated into a boilerplate requirement using DODT. The semantic guidance system was used to assist in filling the boilerplate attributes.

The domain ontology for DMS was specifically developed for usage with the boilerplates tool. The data for the DMS ontology was initially provided by EADS and was then completed by the authors. Table 4 lists the number of concepts, relations and axioms of the DMS ontology.

Figure 4 shows the graphical user interface of the tool. At the top of the interface boilerplates can be selected. The center shows the currently selected boilerplates and text boxes for the attribute values of the requirements. The list of phrases below the text boxes are the suggestions provided by the semantic guidance system. Typing in the text boxes filters the list of suggestions. The tool shows the textual definitions of the concepts as tooltips. Selecting a list entry will add the text to the corresponding text box. The bottom of the interface lists all requirements. Expressions that refer to entities from the domain ontology are underlined with green lines; fixed syntax elements of boilerplates with black. If nouns missing from the domain ontology were to be seen, they would be highlighted with the color red.

Table 5 present statistics about the suggestions produced by the guidance system for the DMS ontology.

Table 4. Ontology Measurements

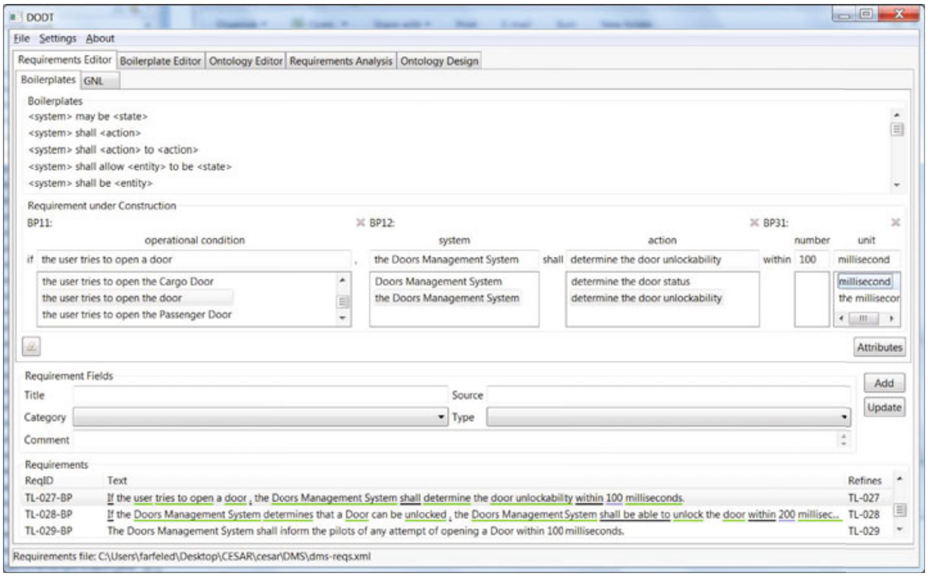
Entity	Count
Concepts	107
Relations	70
Axioms	123
SubClass	108
to Concepts	15
to Attributes	93
Equivalence	15

Table 5. Guidance Suggestions

Type	Count
Concept	212
Verb-Object	69
Subject-Verb-Object	101
Total	382

Table 6. Evaluation Results

Item	Count
Requirements	43
Boilerplates	21
Attributes	120
Complete suggestions	36
Partial suggestions	39



Boilerplates
<system> may be <state>
<system> shall <action>
<system> shall <action> to <action>
<system> shall allow <entity> to be <state>
<system> shall be <entity>

Requirement under Construction
BP11: BP12: BP31:

	operational condition		system		action		number	unit
if	the user tries to open a door	,	the Doors Management System	shall	determine the door unlockability	within	100	millisecond
	the user tries to open the Cargo Door		Doors Management System		determine the door status			millisecond
	the user tries to open the door		the Doors Management System		determine the door unlockability			the millisecor
	the user tries to open the Passenger Door							

Requirement Fields
Title
Source
Category
Type
Comment

Requirements

ReqID	Text	Refines
TL-027-BP	If the user tries to open a door , the Doors Management System shall determine the door unlockability within 100 milliseconds.	TL-027
TL-028-BP	If the Doors Management System determines that a Door can be unlocked , the Doors Management System shall be able to unlock the door within 200 millisec...	TL-028
TL-029-BP	The Doors Management System shall inform the pilots of any attempt of opening a Door within 100 milliseconds.	TL-029

Requirements file: C:\Users\farfeled\Desktop\CESAR\cesar\DMS\dms-reqs.xml

Fig. 4. DODT Screenshot

5.2 Results

Table 6 lists the major results of the evaluation. For 43 requirements, we used 21 different boilerplates. The boilerplate which was used most often (16 times) is ⟨*system*⟩ `shall` ⟨*action*⟩. The 43 boilerplate requirements have a total of 120 attributes. For 36 attributes out of 120 (30%) the semantic guidance system was able to suggest the entire attribute value without any need for a manual change. For another 59 attributes (57.5%) the guidance could suggest at least parts of the attribute value. This leaves 25 attribute values (12.5%) for that the guidance was no help. For partial matches, these are some of the reasons the attribute values had to be modified:

- A different determiner is used than the suggested "the", e.g., "a", "each" or "all".
- The plural is used instead of singular.
- A combination of two or more suggestions is used.
- A subordinate clause is added, e.g., "each door that could be a hazard if it unlatches".

Reasons for no guidance are these:

- Numbers for the ⟨*number*⟩ attribute cannot be suggested.
- Words are used that do not exist in the domain ontology.

Future work will include setting up an evaluation to compare the elicitation time with and without the semantic guidance system. However, due to the high percentage where the guidance was able to help (>85%) we are confident that efficiency improved, even though the presentation of explicit numbers has to be postponed to future work.

We also hope to improve the quality of requirements using the tool. We did a qualitative comparison of the original DMS requirements and the boilerplate requirements. These are our findings:

- Boilerplate requirements encourage using the active voice. In our evaluation the original requirement "Information concerning the door status shall be sent from the Doors Management System to ground station..." was turned into "The Doors Management System shall send information concerning the door status to ground station..." 8 requirements were improved in this way. In some cases missing subjects were added.
- Requirements like "There shall be..." and "It shall not be possible to..." were changed into "The *subject* shall have" and "The *subject* shall not allow...". Such changes make it obvious what part of the system is responsible to fulfill the requirement. To determine the right value for *subject* the original stakeholders should be asked for clarification. Due to timing constraints this was not possible and plausible values were inserted by the authors.
- During the requirements transformation we found that the original requirements used different expressions for seemingly identical things, e.g., "provision to prevent pressurization" and "pressure prevention means" or "airplane"

and "aircraft". Such synonyms are either stored as an equivalence axiom in the domain ontology or, preferably, stakeholders agree upon the usage of one term.
– Using boilerplates improved the overall consistency of the requirements. The original requirements set contains a mixture of "must" and "shall". While using one or the other is probably a matter of taste, one form should be picked and used consistently.
– The guidance system corrected a few typographic errors in the original requirements, e.g., "miliseconds".

We found the tool to be easy to use and user-friendly. This sentiment is shared by the partners in the CESAR project who are also currently evaluating the tool.

6 Conclusion and Future Work

Requirements should be made as consistent, correct and complete as possible to prevent detecting and correcting errors in later design phases. With respect to the three points raised in the introduction about requirements engineering, this work intends to be the foundation for further analyses, e.g., by facilitating requirements categorization with ontology knowledge.

We presented a tool for the elicitation of boilerplate requirements that includes a semantic guidance system which suggests concept names and phrases that were built from relations and axioms of the domain ontology. The tool managed to provide useful suggestions in the majority of the cases (>85%). We realize that the tool needs to be evaluated in a larger context, i.e., more requirements and a larger ontology. We will address this in our future research work.

The selection of suitable boilerplates for a requirement is not always trivial and requires a bit of experience with the boilerplates method. Thus a feature we want to explore and possibly add to the tool is the semi-automatic conversion of natural language requirements into boilerplate requirements.

While creating the DMS ontology, we found there are concepts for which the suggestions provided by the semantic guidance system really help but which are not specific to the problem domain. The most prominent example are measurement units like "second" or "kg". So what we want to do is to collect such entities into a separate ontology and to extend the tool to be able manage several ontologies. This could be handled by adding OWL import declarations to the "main" domain ontology. Such domain-independent ontologies can then be easily reused for guidance in other domains.

Acknowledgments

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement N° 100016 and from specific national programs and/or funding authorities. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria.

References

1. IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830 (1998)
2. OWL 2 Web Ontology Language Direct Semantics. Tech. rep., W3C (2009), http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/
3. Cobleigh, R., Avrunin, G., Clarke, L.: User Guidance for Creating Precise and Accessible Property Specifications. In: 14th International Symposium on Foundations of Software Engineering, pp. 208–218. ACM, New York (2006)
4. Denger, C., Berry, D., Kamsties, E.: Higher Quality Requirements Specifications through Natural Language Patterns. In: 2003 IEEE International Conference on Software - Science, Technology and Engineering, pp. 80–90. IEEE, Los Alamitos (2003)
5. Egyed, A., Grunbacher, P.: Identifying Requirements Conflicts and Cooperation: How Quality Attributes and Automated Traceability Can Help. IEEE Software 21(6), 50–58 (2004)
6. Elazhary, H.H.: REAS: An Interactive Semi-Automated System for Software Requirements Elicitation Assistance. IJEST 2(5), 957–961 (2010)
7. Gotel, O., Finkelstein, C.: An Analysis of the Requirements Traceability Problem. In: 1st International Conference on Requirements Engineering, pp. 94–101 (1994)
8. Gottesdiener, E.: Requirements by Collaboration: Workshops for Defining Needs. Addison-Wesley, Reading (2002)
9. Hull, E., Jackson, K., Dick, J.: Requirements Engineering. Springer, Heidelberg (2005)
10. Ibrahim, N., Kadir, W., Deris, S.: Propagating Requirement Change into Software High Level Designs towards Resilient Software Evolution. In: 16th Asia-Pacific Software Engineering Conference, pp. 347–354. IEEE, Los Alamitos (2009)
11. Jackson, J.: A Keyphrase Based Traceability Scheme. IEEE Colloquium on Tools and Techniques for Maintaining Traceability During Design, 2/1–2/4 (1991)
12. Kaindl, H.: The Missing Link in Requirements Engineering. Software Engineering Notes 18, 30–39 (1993)
13. Kaiya, H., Saeki, M.: Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. In: 5th Int. Conf. on Quality Software, pp. 223–230 (2005)
14. Kitamura, M., Hasegawa, R., Kaiya, H., Saeki, M.: A Supporting Tool for Requirements Elicitation Using a Domain Ontology. Software and Data Technologies, 128–140 (2009)
15. Kotonya, G., Sommerville, I.: Requirements Engineering. John Wiley & Sons, Chichester (1998)
16. Matsuo, Y., Ogasawara, K., Ohnishi, A.: Automatic Transformation of Organization of Software Requirements Specifications. In: 4th International Conference on Research Challenges in Information Science, pp. 269–278. IEEE, Los Alamitos (2010)
17. Omoronyia, I., Sindre, G., Stålhane, T., Biffl, S., Moser, T., Sunindyo, W.: A Domain Ontology Building Process for Guiding Requirements Elicitation. In: 16th REFSQ, pp. 188–202 (2010)
18. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A Core Ontology for Business Process Analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 49–64. Springer, Heidelberg (2008)
19. Rupp, C.: Requirements-Engineering und -Management. Hanser (2002)
20. Ståhane, T., Omoronyia, I., Reichenbach, F.: Ontology-Guided Requirements and Safety Analysis. In: 6th International Conference on Safety of Industrial Automated Systems (2010)
21. Watkins, R., Neal, M.: Why and How of Requirements Tracing. IEEE Software 11(4), 104–106 (1994)

The Semantic Public Service Portal (S-PSP)

Nikolaos Loutas[1,2], Deirdre Lee[1], Fadi Maali[1], Vassilios Peristeras[3], and Konstantinos Tarabanis[2]

[1] DERI, National University of Ireland Galway (NUIG), Galway, Ireland
{nikos.loutas,deirdre.lee,fadi.maali}@deri.org
[2] Information Systems Lab, University of Macedonia, Thessaloniki, Greece
kat@uom.gr
[3] European Commission, Directorate-General for Informatics, Interoperability Solutions for European Public Administrations
vassilios.peristeras@ec.europa.eu

Abstract. One of government's responsibilities is the provision of public services to its citizens, for example, education, health, transportation, and social services. Additionally, with the explosion of the Internet in the past 20 years, many citizens have moved online as their main method of communication and learning. Therefore, a logical step for governments is to move the provision of public services online. However, public services have a complex structure and may span across multiple, disparate public agencies. Moreover, the legislation that governs a public service is usually difficult for a layman to understand. Despite this, governments have created online portals to enable citizens to find out about and utilise specific public services. While this is positive, most portals fail to engage citizens because they do not manage to hide the complexity of public services from users. Many also fail to address the specific needs of users, providing instead only information about the most general use-case. In this paper we present the Semantic Public Service Portal (S-PSP), which structures and stores detailed public-services semantically, so that they may be presented to citizens on-demand in a relevant, yet uncomplicated, manner. This ontology-based approach enables automated and logical decision-making to take place semantically in the application layer of the portal, while the user remains blissfully unaware of its complexities. An additional benefit of this approach is that the eligibility of a citizen for a particular public service may be identified early. The S-PSP provides a rich, structured and personalised public service description to the citizen, with which he/she can consume the public service as directed. In this paper, a use-case of the S-PSP in a rural community in Greece is described, demonstrating how its use can directly reduce the administrative burden on a citizen, in this case is a rural Small and Medium Enterprise (SME).

Keywords: eGovernment, public service, semantic, portal.

1 Introduction

Public service provision is an important duty of all government departments. Independent of what kind of service it is, every public service can be broken down into two distinct but complementary phases: the *informative* phase and the *performative* phase [1]. During the informative phase, the service provider provides information about the service to the citizen/ business[1], while during the performative phase the citizen utilises the public service. The informative phase is essential for optimal service utilisation and public-administration efficiency, however it is often overlooked by governments. In order to effectively use a public service, citizens must identify which public services address their needs and find answers to their questions regarding these services, e.g. *"am I eligible for this service", "what is the outcome of the service", "which public agency provides the service",* etc. In this paper, we present a portal that facilitates the informative phase of public-service provision.

An additional factor of public services is that they often have complex structures, and may be specialized into a variety of service versions. For example a public service that is concerned with the issuing of a driving license, may have alternative versions if this is the first license of the applicant, if the applicant is over 60, if the applicant wishes to drive lorries, etc. It is therefore not enough for citizens to identify a public service in general, but they must also go one step further and identify the specific service version for which they are eligible. The service versions refine the generic public service further and may be differentiated from one another according to:

i. the profile of the citizen that wishes to consume the service;
ii. the service inputs and outputs; and/or
iii. the service workflow.

However, traditional governmental portals still follow a one-size-fits-all approach. Thus the portal cannot react differently and tailor the offered public services to the needs and the profile of each individual citizen. Moreover, the citizen has to figure out on their own whether they are eligible for the service by reading lengthy public service descriptions (which very often include legal terminology). These are common problems in all existing national eGovernment portals. According to [2], the most typical problems of eGovernment portals can be grouped into the following categories:

- The user is unable to find the desired information or service.
- The user is unable to achieve his goal, even though the system supports it and he has started along the path to achieve it.
- The user is able to accomplish his goal, but not efficiently, e.g., easily and quickly.

In order to enhance the informative part of public service provision and improve existing governmental portals, this paper introduces the Semantic Public Service Portal (S-PSP), which aims:

- To inform citizens whether they are eligible for a specific public service;
- To personalize the public-service-related information according to the profile and the specific needs and wants of the citizen and identify the specific public service version;

[1] For the remainder of the paper we refer only to citizens for the sake of brevity, but citizens and businesses are implied.

- To provide complete and well-structured information for the public service; and
- To allow citizens to invoke public services that are available online (if a service execution environment is in place).

The S-PSP was initially developed in the context of the SemanticGov project[2], where it played the role of the national Member State portal [3]. It served as an entry point for the citizens to the public services offered by the SemanticGov platform. Two prototypes of the portal were installed at the Region of Central Macedonia in Greece and the City of Turin in Italy [4]. Currently, the portal is one of the three building blocks of the Rural Inclusion[3] platform, as will be described in more detail in section 5. A running prototype of the portal is available at http://vmsgov03.deri.ie:8080/rural-inc/services?pilot=gr&pageLanguage=en

The remainder of the paper is structured as follows. Section 2 discusses related efforts. The S-PSP is presented in section 3, along with an overview of its architecture. The ontologies that the S-PSP used are discussed in section 4. An example of the S-PSP in use in the context of the Chios Chamber of Commerce is presented in section 5. Finally, our conclusions and future research directions are discussed in section 6.

2 Related Work

Researchers have tried to solve parts of the problem that we described in the previous section, focusing mostly on facilitating service search and discovery. Fang et al. [5] support the selection of an optimal set of featured service-links. These links will then appear on the homepage of an eGovernment portal, thus helping users to locate services more easily by reducing the number of steps that they have to perform until the desired service is found. This is expected to improve the citizens' satisfaction and consequently increase the number of people using the portal. Therefore, a heuristic Web-mining algorithm called ServiceFinder is proposed, which aims to help citizens find the services that they are looking for in eGovernment portals. ServiceFinder uses three metrics to measure the quality of eGovernment service selection, which will then appear as links on the homepage of the portal. These are effectiveness (degree of easiness to locate the desired service), efficiency (probability to locate the desired service) and utilization (sum of easily located desired services). The metrics are calculated using patterns either extracted from the structure of the eGovernment portal or mined from a Web log. Although this approach may improve the service discovery by organizing better the available services within the portal, the process of finding a service in the portal is still based on a trial and error approach. This means that the user is still browsing the eGovernment portal in order to find the desired service. Another drawback of this approach as compared to ours is that it provides no information to the citizen with respect to his/her eligibility for the identified public service.

[2] `http://www.semantic-gov.org`

[3] `http://www.rural-inclusion.eu`

Sacco [6] proposes a solution enabled by dynamic taxonomies, which support different facets that may be used by citizens. The facets that the system provides are: services, events of life, type of information, location, type of citizenship, person with special rights and person profile. The use of dynamic taxonomies makes this approach very flexible and fast. This is due to the fact that dynamic taxonomies adapt dynamically to the subset of the universe on which the user is focusing. The use of multiple facets enhances further the agility of the approach. Nonetheless, this work suffers from similar problems as other approaches that organize services in hierarchical category trees. The user may have to browse deep into the dynamic taxonomy and should also be aware of the way that public administration has decided to organize services (even if multiple facets are made available), which may differ from his/her perspective. Therefore, the cognitive effort of citizens is not reduced as much as expected. The eligibility question remains unanswered in this case as well.

Recently, Stollberg and Muth [7] proposed an approach for service customization which is based on simplified service variants that expose only the service features that are relevant to the profile of a specific user or to the characteristics of a specific usage scenario. Hence, the authors define *(i)* the service variability metamodel, *(ii)* the main actors and *(iii)* the lifecycle of the service customization process. They also provide a set of tools based on the metamodel to support the service variant creation. This work approaches the problem of service customization from a different perspective than the one implemented in our work. However, we too acknowledge the need to formalize the process, we thus use the Public Service Ontology described in section 4 to model public services and the SBVR standard [8] for formally expressing the business rules that lead to different service versions.

The OneStopGov project[4] delivered a life-event portal that supports the active, citizen-centric approach [9]. The portal follows a structured dialogue approach, based on workflows that model life-events in order to personalise them to citizen profiles and facilitate their execution. Hence, the OneStopGov approach adapts the life-event to citizen's profile, which practically means that citizens with different profiles will most likely execute different versions of the same life-event. In the FIT project[5], an adaptive eGovernment portal has been implemented [10]. The approach, which employs Semantic Web and Web 2.0 technologies, proposes a framework which captures the user's behaviour in the portal and adapts the portal accordingly. FIT's approach also recognizes the service version problem that we have described earlier and tries to overcome this by providing personalized eGovernment services. In order to achieve this, the FIT portal follows an iterative ontology-driven adaptation approach (monitor, analyze, plan, execute). The FIT portal also uses OWL ontologies to model users, events and adaptation rules.

The portal presented in this paper tries to fulfil similar objectives like the related efforts described so far. Nevertheless, some differences can be spotted both in terms of functionalities provided and in terms of the technologies used. For example, none of the related efforts decide on the eligibility of the citizen for a public service before

[4] `www.onestopgov-project.org`
[5] `http://www.fit-project.org/`

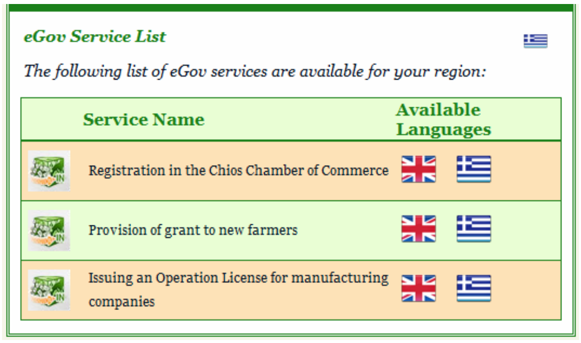
the execution of the service. This is a very strong asset of our approach, as the eligibility check at an early stage during the informative part of service provision can save the citizen a lot of time and money. It is interesting that the work of [6] bears some resemblance with ours in the way that services are modelled and organized, but what is different, apart from the use of ontologies versus taxonomies, is the fact that in our work services are described at a greater level of granularity e.g. the distinction between service type and service version. This difference is very important and due to this Sacco's work is not able to personalize services, but only provide generic info about them. Moreover, it does not answer the eligibility question.

3 Semantic Public Service Portal (S-PSP)

The Semantic Public Service Portal (S-PSP) provides information about available public-services, which a user may browse and search in a customisable and user-friendly manner. Through the use of the S-PSP, a user can quickly identify:

- which public service(s) are applicable to their individual use-case,
- whether they are eligible for these public service(s), and
- what is required from the user to complete these public service(s) for their individual use-case.

Fig. 1 shows the homepage of the S-PSP with the list of all currently available public services and the languages they are available in.



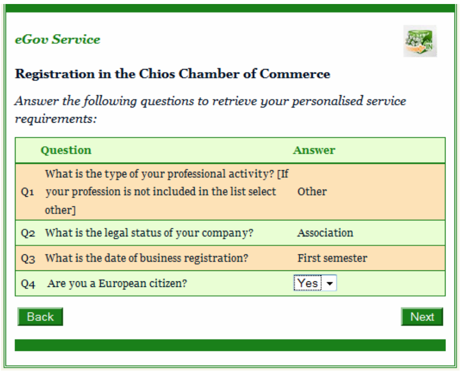
eGov Service List

The following list of eGov services are available for your region:

Service Name	Available Languages
Registration in the Chios Chamber of Commerce	🇬🇧 🇬🇷
Provision of grant to new farmers	🇬🇧 🇬🇷
Issuing an Operation License for manufacturing companies	🇬🇧 🇬🇷

Fig. 1. The list of public services available in the S-PSP

Once a user selects the public service they are interested in, the dialogue page appears, as shown in Fig. 2. At this step, the user answers a series of questions, which will determine if a user is eligible for this service and what information they will need to provide/complete to utilise this service. Fig. 3 shows the customised information that this particular users requires to utilise this service, moving from a one-size-fits-all approach that is unrealistic in the case of public services.



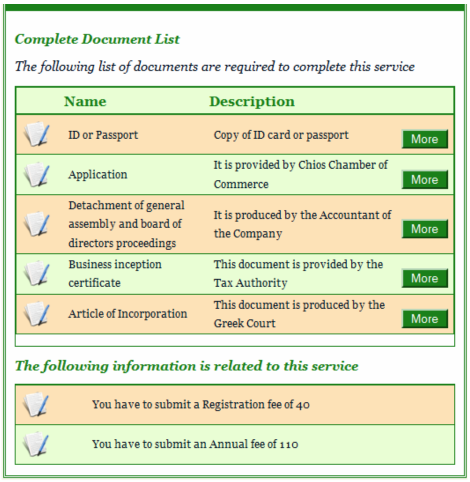
eGov Service

Registration in the Chios Chamber of Commerce

Answer the following questions to retrieve your personalised service requirements:

	Question	Answer
Q1	What is the type of your professional activity? [If your profession is not included in the list select other]	Other
Q2	What is the legal status of your company?	Association
Q3	What is the date of business registration?	First semester
Q4	Are you a European citizen?	Yes

Fig. 2. The public-service dialogue to customise the public-service information



Complete Document List

The following list of documents are required to complete this service

Name	Description	
ID or Passport	Copy of ID card or passport	More
Application	It is provided by Chios Chamber of Commerce	More
Detachment of general assembly and board of directors proceedings	It is produced by the Accountant of the Company	More
Business inception certificate	This document is provided by the Tax Authority	More
Article of Incorporation	This document is produced by the Greek Court	More

The following information is related to this service

You have to submit a Registration fee of 40

You have to submit an Annual fee of 110

Fig. 3. The customised information required to utilise this public service

3.1 S-PSP Architecture

The S-PSP follows a three-tier architecture, as shown in Fig. 4, which comprises of:

- the *User Interface Layer*, which facilitates the interaction between the citizens and the portal, acting as an entry-point to the portal's functionalities.
- the *Application Layer*, which implements the functionalities provided to the citizens. This layer consists of two components:
  - the Service Tree Locator (STL) and
  - the Query Mechanism.
- the *Semantic Repository Layer* where all the semantic artefacts (ontologies) used by the portal are stored.

While the user interacts directly with the S-PSP interface, the service provider collaborates with an ontology manager to define the public-service descriptions, which are then added to the Semantic Repository. This process of semantic public-service description is currently in the process of being automated, so that the service provider may create the service description using a tool that will substitute the ontology manager.

The User Interface Layer

The User Interface (UI) Layer provides citizens with the means to interact with the portal. Its main functionality includes presenting the questions asked by the Query Mechanism to the citizens and collecting their answers. The answers are then returned to the Query Mechanism. It is important to clarify that all information that is made available through the UI, e.g. list items in dropdown lists, questions and possible answers etc., comes from the underlying ontologies stored in the Semantic Repository.

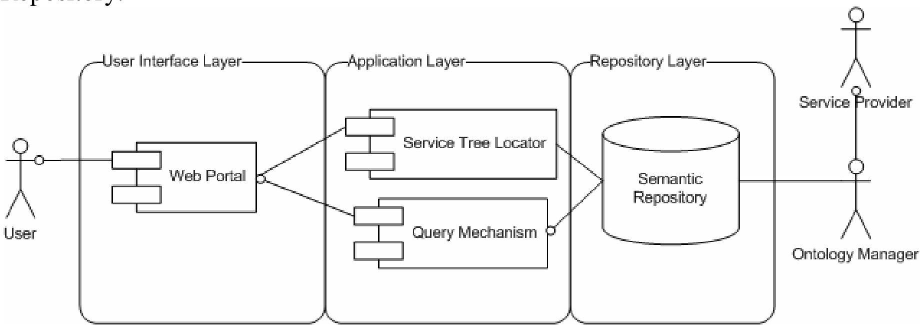


Fig. 4. S-PSP Architecture

The Application Layer

The Application Layer consists of the Service Tree Locator (STL) and the Query Mechanism components. *The Service Tree Locator (STL)* identifies the appropriate Service Tree Ontology (STO), which models the public service that addresses the user's requirements. Citizens can enter keywords in the portal's UI to describe the service that they are looking for. These keywords are sent to the STL, which then queries the semantic repository using SPARQL[6] queries to find matching public-service descriptions. The STL may contact WordNet[7] in order to find synonyms and hypernyms/hyponyms for the keywords entered by the user, thus making the keyword search more effective. Finally, the resulting public services are returned to the citizens in order for them to select the appropriate one. SPARQL was chosen as the semantic query language, as it is a W3C Recommendation and has a large, active community.

The Query Mechanism (QM) is the core component of the S-PSP as it identifies the questions to include in the public-service dialogue, based on a user's previous

[6] http://www.w3.org/TR/rdf-sparql-query/
[7] http://wordnet.princeton.edu/

answers. The QM continually checks the user's eligibility for the public service and, if eligible, it stores the user's answers in order to determine the personalised information required to provide the service.

The Repository Layer

The Repository Layer contains the semantic repository component, which houses all of the ontologies of the S-PSP. These will be discussed in more detail in the next section.

4 Semantic Description of Public Services

The S-PSP structures and stores detailed public-services semantically. This ontology-based approach enables automated and logical decision-making to take place in the application layer of the portal, while the user remains unaware of its complexities. Adopting an ontology-based approach reaps the benefit of the flexibility of the RDF model. This, together with the highly abstract, conceptual nature of RDF triples, enable direct translation of fairly complex service scenarios into a machine-readable form. Inference over OWL ontologies helps achieve more concise and readable service descriptions. As a simple example, defining `SchengenCountry` as a `rdfs:subClassOf EuropeanCountry` class, means that any instance of `SchengenCountry` is also an instance of `EuropeanCountry` and eliminates the need for explicit listing of such a condition. However, in order to create semantic public service descriptions in this way requires a detailed analysis of the public description. This is usually carried out by an ontology engineer, in close conjunction with a civil servant who is extremely familiar with the intricacies of the public service. While this may be seen as a limitation of the approach, it results in a simple and effective tool from the citizen side. We are also in the process of creating a public-service description tool, which will replace the need for an ontology engineer to assist the civil servant in describing a public service semantically. The different kinds of ontologies that are utilised by the S-PSP are:

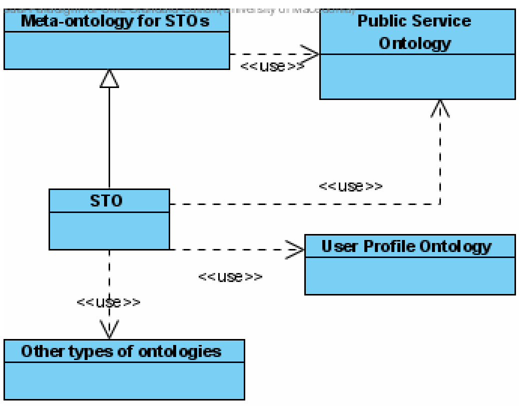


Fig. 5. Ontologies used by the S-PSP

- Service Tree Ontology (STO)
- Public Service Ontology
- User Profile Ontology: models user preference and localization information.
- Other domain-specific Ontology: captures business information needed for service description.

We describe the details of the main ontologies used in the following sections.

4.1 Service Tree Ontology

A Service Tree Ontology (STO) formally defines the dialogue that would usually take place between a public-service provider and a citizen for a particular public service. STOs have a tree-like structure and are written in OWL. The dialogue starts from a generic public service, which is stepwise refined after every question/answer pair. In the case that the citizen is eligible for the specific public service, the dialogue leads to the public-service version that matches their profile and a detailed structured description of the public service version is made available. Otherwise the citizen is informed that they are not eligible for the specific public service. STOs contain the business rules from which the different service versions derive as well as the questions that will be asked to the citizen in order to collect information, which enables the portal to personalize the public service and decide on the eligibility of the citizen and on the matching service version. Moreover, the user interface of the portal is dynamically created based on information encoded in the STOs.

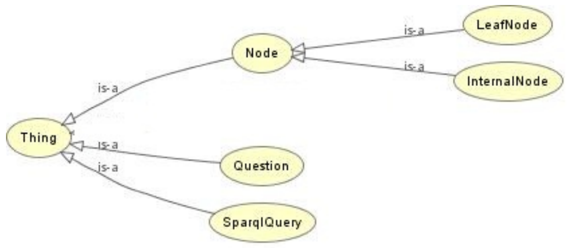


Fig. 6. The meta-ontology for STOs

In order to formalize the development of STOs, we have created a meta-ontology that defines the classes that comprise an STO. Each STO is then created as an instance of the meta-ontology. The meta-ontology contains classes that have been derived from the modelling of the aforementioned dialogue, as shown in Fig. 6.

- **Node.** Nodes of an STO represent different states of the dialogue. A node has the following attributes:
  - The *hasDescription* attribute provides a brief description of the node, as to what the node represents in the STO.
  - The *hasEvidencePlaceholder* attribute refers to the administrative documents, e.g. certificates, which relate to a specific service version. A *Node* may contain zero or more *EvidencePlaceholders*.

  - The *containsPieceOfInformation* attribute refers to other types of information related to a specific service version, e.g. the amount of a fee that has to be paid. A *Node* may contain zero or more *PieceOfInformation*.

The following three classes, i.e. *InternalNode* and *LeafNode* have been defined in the ontology as subclasses of *Node*. They thus inherit all its attributes.

- **InternalNode.** This class represents those nodes of the STO that have descendants. Apart from the attributes that they inherit from *Node*, *InternalNodes* have also:
  - The *hasChildNode* attribute which indicates the descendants of the current node. There can be more than one descendants, which constitute specializations of their parent node, mainly by containing more information about the citizen's profile.
  - The *hasQuestion* attribute which refers to a specific question asked to the citizen.
  - The *isRoot* attribute which indicates whether the specific node is the initial node of the dialogue or not.
- **LeafNode.** This class represents those nodes of the STO that have no descendants. *LeafNodes* indicate the termination of the dialogue, whether successful or not. Apart from the attributes that they inherit from *Node*, *LeadNodes* have also the *isNotEligible* attribute which if true indicates that the citizen is not allowed to use the specific public service.
- **Question.** This class represents the questions that the portal poses to the citizen. It has two attributes:
  - *hasData* models the question itself, e.g. "What is your marital status?"
  - *hasAnswer* models the possible answers, e.g. in the previous question "married, single, divorced, widow".
- **SparqlQuery.** This class represents formally a business rule of a specific public service. The business rule is expressed as a SPARQL query which is stored in the *hasData* attribute.

4.2 Public Service Ontology

The S-PSP capitalizes on the GEA Public Service Model implementation in OWL (Public Service Ontology) [11]. The Public Service Ontology, which is depicted in Fig. 7, is used for representing public service related information by creating instances of the various classes that comprise the description of the public service. A brief description of the Public Service Ontology's classes follows. *Societal Entities* (e.g. citizen, business) have *Needs* related to specific *Goals*. A *Societal Entity* requests a *Public Administration (PA) Service* to serve its *Goals*. PA Services are categorized in several *Domains* (e.g. Health, Transportation). Each *Domain* object is divided into several *SubDomain* objects (e.g. Domain Transportation has *SubDomains* Ground Transportation, Air Transportation and Water Transportation). There are several types of *Social Entities* (e.g. legal entity, physical person). There are two categories of *Governance Entities* participating in service provision: *Political* Entities and *Public Administration Entities*. Based on the role which *PA Entities* can acquire during the service execution phase, we identify four *Roles*:

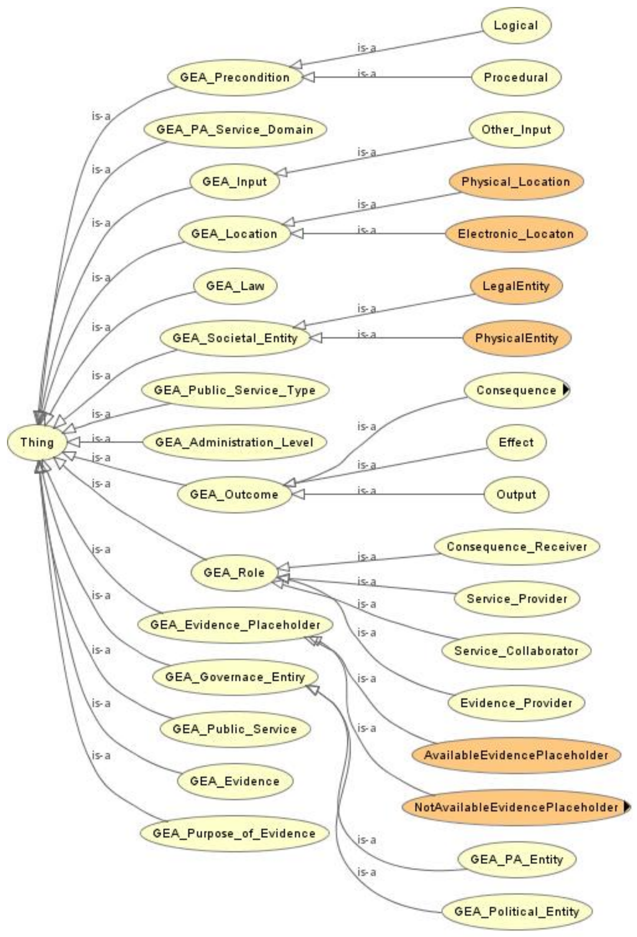


Fig. 7. The Public Service Ontology

- *Service Provider* is the PA Entity that provides the service to the Societal Entities (clients). The PA Entities belong to an *Administrative Level* (e.g. municipality, regional).
- *Evidence Provider* is the PA Entity that provides necessary Evidence to the Service Provider in order to execute the PA Service.
- *Consequence Receiver* is the PA Entity that should be informed about a PA Service execution.
- *Service Collaborator* is the PA Entity that participates in the provision of a public service (but is not the service provider).

Political Entities define *PA Services* which are governed by *Preconditions* usually specified in *Legal Acts - Laws*. *Preconditions* set the general framework in which the service should be performed and the underlying business rules that should be fulfilled for the successful execution of the *PA Service*. *Preconditions* can be formally expressed as a set of clauses. *Preconditions* are validated by *Piece of Evidence* serving a *Purpose*. As *Evidence* is primarily pure information, it is stored in *Evidence Placeholders*, thus the *Evidence Placeholder* contains *Pieces of Evidences*. *PA Service* use specific types of *Evidence Placeholders* as *Input*. The *Outcome* refers to the different types of results a PA Service may have. GEA defines three types of *Outcome*:

- *Output*, which is the documented decision of the *Service Provider* regarding the service asked by a *Societal Entity*. This is currently embedded and reaches the client in the form of an *Evidence Placeholder*.
- *Effect*, which is the change in the state of the real world (e.g. transfer money to an account) caused by the execution of a service. In cases where administration refuses the provision of a service, there is no Effect.
- *Consequence*, which is information about the executed *PA Service* that needs to be forwarded to interested parties.

4.3 Other Ontologies

In addition to the STOs, the meta-ontology for STOs, and the Public service ontology, the following OWL ontologies are also used by the S-PSP:

- Ontologies that model the profile of businesses and citizens, for example the brand name, type, or legal status.
- Ontologies that contain public service related information, such as the administrative documents that are required as input for the different versions of the public service (modelled as instances of the EvidencePlaceholder class of the Public Service Ontology).
- Ontologies that include listings of countries, nationalities, and business types.

4.4 Query Mechanism's (QM) Usage of an STO

The QM, as discussed in section 3.1, is the core component of the S-PSP, as it identifies the questions to include in the public-service dialogue, by traversing the corresponding public-service STO. During the traversal of the STO, the public service that the citizen has selected is being personalized according to their answers. This is achieved by resolving the generic service type into the appropriate service version. It is important to note that at each stage of the traversal, the next step option is unique. This means there is no case where the same citizen could follow two different paths in the same STO. If the current node is an *InternalNode* then the QM has to verify the conditions of all its descendants, which are expressed as SPARQL queries. Therefore, the QM takes the appropriate question from the STO and forwards it to the UI so that the question can be displayed to the citizen. In case the current node is a *LeafNode*, i.e. it has no descendants, then the end of the structured conversation has been reached. At this point the portal has collected all the necessary information for identifying the specific public service version that matches the citizen's profile and for deciding on their eligibility. In case the citizen is not eligible for one of the service versions that are modelled in the STO (*isNotEligible* is set to *true)*, then the QM terminates its execution and returns a notification message, for example, 'You are not eligible for this service because you are under 18 years old'.

QM Traversal Process in Pseudocode:

```
  BEGIN
Let IN be the set of InternalNodes,
Let LN be the set of LeafNodes such as IN ∪ LN = N, where N the set of Node
instances defined in the STO
Let root be the first IN of the STO
Let curr be the Node (either InternalNode or LeafNode) to be processed
Let validated  be a variable that stores the result of the evaluation of
IN's SparqlQuery
Let ServiceBasedUserProfile be the citizen ontology instance that models the
profile of a citizen/business
curr := root
while (curr ∉ LN)
               validated := false
               askQuestions(curr)
               ServiceBasedUserProfile := readAnswers()
               foreach descendant d of curr
                 if (evaluate(d)=true)
                   curr := d
                   validated := true
                   break
                 end_if
               end_foreach
               if (validated = false) //in case there is no valid descendant
node
                 break
               end_if
end_while
 showPublicServiceDescription(curr)
END
```

5 S-PSP In-Use: Rural Inclusion Trial-Site Chios

The S-PSP is currently being utilised by Rural Inclusion[8], an EC project funded under the Competitiveness and Innovative Framework Programme. Rural Inclusion aims at adopting a state-of-art infrastructure that will facilitate the offering of innovative services by public administration in rural areas. The S-PSP is one of three components that make up the Rural Inclusion Platform, which is being rolled out across five European trial-sites in France, Martinique, Greece, Latvia and Spain.

The Chios Chamber of Commerce is one of the trial-partners. It is supervised by the Greek Ministry of Development and serves as a consultant of a wide range of business-related matters for the Greek island of Chios. Public services that the chamber provides and that are presented in the Rural Inclusion platform include:

[8] http://www.rural-inclusion.eu

- Registration of a new business in Chios
- Provision of grants to new farmers
- Issuing of operation licenses for manufacturing companies

In this section, we will present how the ‘registration of a new business in Chios’ public service is modelled using the S-PSP ontologies. Although this service sounds straight-forward, there are actually more than 20 different versions which stem from variations in the legal status of the enterprise/entrepreneur, on the type of their activity, and on their nationality. Initially, a documentation exercise was carried out with the service provider, in order to gather all the details with regards to the provision of the specific service. Hence, the business rules that are associated with this public service were identified and the activity diagram of the structured dialogue was designed. This was then encoded into an OWL ontology by the ontology manager.

For example, one of the business rules of the ‘registration of a new business’ public service expressed in structured English (following SBVR [8]) reads *“It is obligatory that each SA company with equity capital less than 64,000 Euros pays an annual fee of 160 Euros.”* In order to collect the information that validates this rule, a question was created asking the equity capital of the SA company, i.e. *“Please provide the equity capital of your company”*. In the STO developed for this public service, this question was then attached to one of the *InternalNodes*. This specific *InternalNode* has three descendants. One of them will correspond to the case of an SA company with capital less than 64,000 Euros. This will be expressed by a SPARQL expression linked to this specific node, as shown below.

The InternalNode instance

```
 <owl:Thing rdf:about="#ObligesRegistration_SA">
        <rdf:type rdf:resource="&www;PortalOntology.owl#InternalNode"/>
        <prtl:containsPieceOfInformation rdf:datatype="&xsd;string"
>Registration fee 160</prtl:containsPieceOfInformation>
        <prtl:hasEvidencePlaceholder
rdf:resource="&www;DocumentOntology.owl#Application"/>
        <prtl:hasEvidencePlaceholder rdf:resource=
"&www;DocumentOntology.owl#ArticleOfIncorporation"/>
        <prtl:hasEvidencePlaceholder rdf:resource="
&www;DocumentOntology.owl#BusinessInceptionCertificate"/>
        <prtl:hasEvidencePlaceholder
rdf:resource="&www;DocumentOntology.owl#Chios_Perfecture_Statement"/>
        <prtl:hasEvidencePlaceholder
rdf:resource="&www;DocumentOntology.owl#IDorPassport"/>
        <prtl:hasCondition rdf:resource="#LicencedSACompany"/>
        <prtl:hasChildNode rdf:resource="#SA_equitybetween"/>
        <prtl:hasChildNode rdf:resource="#SA_equityless"/>
        <prtl:hasChildNode rdf:resource="#SA_equitymore"/>
        <prtl:hasQuestion rdf:resource="#qstnEquityCapital"/>
    </owl:Thing>
```

The Question instance

```
<owl:Thing rdf:about="#qstnEquityCapital">
        <rdf:type rdf:resource="&www;PortalOntology.owl#Question"/>
        <prtl:hasAnswer rdf:datatype="&xsd;string"></prtl:hasAnswer>
        <prtl:hasData xml:lang="en"> Please provide the equity capital of
your company </prtl:hasData>
    </owl:Thing>
```

The SparqlQuery instance

```
<owl:Thing rdf:about="#EquityLess">
        <rdf:type rdf:resource="&www;PortalOntology.owl#SparqlQuery"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Checks if the EU applicant requires licensing and is
registering an association</rdfs:comment>
        <prtl:hasData rdf:datatype="&xsd;string"
            >PREFIX bo:&lt;http://www.owl-
ontologies.com/BusinessOntology.owl#&gt;            PREFIX
rdf:&lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt;            PREFIX
xsd:&lt;http://www.w3.org/2001/XMLSchema#&gt;            SELECT ?x FROM
&lt;http://www.owl-ontologies.com/BusinessOntology.owl&gt;
WHERE { ?x rdf:type ?t.  FILTER(?t=bo:SME). ?x bo:hasEquityCapital ?it.
FILTER(?it=bo:Fund_less_Than_64000).}</prtl:hasData>
    </owl:Thing>
```

6 Evaluation

The evaluation of the S-PSP is ongoing as part of the Rural Inclusion Project. As stated previously, the S-PSP is one of three components that make up the Rural Inclusion Platform, which is being rolled out across five European public-sector trial-sites in France, Martinique, Greece, Latvia and Spain. Initial results are positive, with the main constructive criticism focusing on improving the intuitive integration of the S-PSP into the actual trial-partner sites, and where they currently provide the actual public services. A complete evaluation will be published at a future date.

7 Conclusion

This paper presents an ontology-based, public-services portal, the S-PSP, which facilitates the informative phase of public service provision. It checks the eligibility of the citizens for a specific public service before the actual execution of the service, thus saving them time, effort and money. Also the public service related information is personalised according to the profile and the specific needs and wants of the citizen and the specific public service version required is identified, thus providing targeted, tailored and comprehensive information. The S-PSP's architecture is modular and as such it is easily extendable. This has been shown with the Rural Inclusion Platform, where the S-PSP has been integrated with other components for a specific solution. The S-PSP is also decoupled from the public-service execution environment that may be available in different technologies and communicates with it using Web Services. The main advantage of this portal is its use of semantics to describe all aspects of public-services, resulting in reusable, extensible public-service data. New public-services may be added to the portal through the creation of a new STO.

Acknowledgments. The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) and the European Union under Grant No. CIP-ICT PSP-2008-2/238900 (Rural Inclusion). The authors would like to thank all the Rural Inclusion project partners for the creative discussions and ideas.

References

1. Peristeras, V., Tarabanis, K.: The Governance Architecture Framework and Models. In: Advances in Government Enterprise Architecture, IGI Global Information Science Referencece (2008)
2. Thomas, S., Schmidt, K.U.: D4: Identification of typical problems in eGovernment portals, in Technical Report, FIT Consortium (2006)
3. Loutas, N., Peristeras, V., Tarabanis, K.: Providing Public Services to Citizens at the National and Pan-European level using Semantic Web Technologies. In: 6th Eastern European eGov Days, Prague, Czech Republic (2008)
4. Loutas, N., et al.: A Semantically Enabled Portal for Facilitating the Public Service Provision. In: Semantic Technologies for E-Government, pp. 287–314. Springer, Berlin (2010)
5. Fang, X., Liu Sheng, O.R., Chau, M.: ServiceFinder: A Method Towards Enhancing Service Portals. ACM Transactions on Information Systems (TOIS) 25(4) (2007)
6. Sacco, G.M.: Interactive Exploration and Discovery of eGovernment Services. In: 8th Annual International Conference on Digital Government Research: Bridging Disciplines & Domains, Philidelphia, US (2007)
7. Stollberg, M., Muth, M.: Service Customization by Variability Modeling. In: Service Customization by Variability Modeling, 5th International Workshop on Engineering Service-Oriented Applications (WESOA 2009) co-located with the ICSOC-ServiceWave, Vienna, Austria (2009)
8. OMG, *Semantics of Business Vocabulary and Business Rules (SBVR) V1.0* (2008), `http://www.omg.org/spec/SBVR/1.0/`
9. Tambouris, E., Vintar, M., Tarabanis, K.: A life-event oriented framework and platform for one-stop government. In: 4th Eastern European eGov Days, Prague, Czech Republic (2006)
10. Schmidt, K.U., et al.: Personalization in e-Government: An Approach that combines Semantics and Web 2.0, in: Semantic Technologies for E-Government. In: Vitvar, T., Peristeras, V., Tarabanis, K. (eds.) Semantic Technologies for E-Government, pp. 261–285. Springer, Berlin (2010)
11. Peristeras, V., et al.: Ontology-Based Search for eGovernment Services Using Citizen Profile Information. Journal of Web Engineering 8(3), 245–267 (2009)

DataFinland—A Semantic Portal for Open and Linked Datasets

Matias Frosterus, Eero Hyvönen, and Joonas Laitio

Semantic Computing Research Group (SeCo)
Aalto University and University of Helsinki
firstname.lastname@tkk.fi
http://www.seco.tkk.fi/

Abstract. The number of open datasets available on the web is increasing rapidly with the rise of the Linked Open Data (LOD) cloud and various governmental efforts for releasing public data in different formats, not only in RDF. The aim in releasing open datasets is for developers to use them in innovative applications, but the datasets need to be found first and metadata available is often minimal, heterogeneous, and distributed making the search for the right dataset often problematic. To address the problem, we present DataFinland, a semantic portal featuring a distributed content creation model and tools for annotating and publishing metadata about LOD and non-RDF datasets on the web. The metadata schema for DataFinland is based on a modified version of the voiD vocabulary for describing linked RDF datasets, and annotations are done using an online metadata editor SAHA connected to ONKI ontology services providing a controlled set of annotation concepts. The content is published instantly on an integrated faceted search and browsing engine HAKO for human users, and as a SPARQL endpoint and a source file for machines. As a proof of concept, the system has been applied to LOD and Finnish governmental datasets.

1 Metadata for Linked Datasets

Linked Data refers to data published on the web in accordance with four rules[1] and guidelines [2] that allow retrieving metadata related to data entities, and linking data within and between different datasets. The datasets and their relations are represented using RDF (Resource Description Framework) and entities are identified by Uniform Resource Identifiers (URIs)[2], which allows the use of the Hypertext Transfer Protocol (HTTP) to retrieve either the resources themselves, useful descriptions of them, or links to related entities [3].

The Linked Open Data community project[3] has collected a large number of datasets and mappings between them. However, little metadata about the datasets is provided aside from short, non-uniform descriptions. As the number of linked datasets [8] grows, this approach does not allow for easy understanding of what kind of dataset are offered, who provides them, what is their subject, how they interlink with each other, possible

[1] http://www.w3.org/DesignIssues/LinkedData.html
[2] http://www.w3.org/TR/uri-clarification/
[3] http://linkeddata.org

licensing conditions, and so on. Such information should be available both to human users as well as machines of the Semantic Web.

Aside from properly linked datasets in RDF format, various organizations have also began publishing open data in whatever format they had it in. The governments of the United States and the United Kingdom have been releasing their governmental data in an open format[4] and other governments are following suit. This provides another source of datasets which have their own unique challenges in classifying and subsequently finding them in that they are released in arbitrary formats with varying amounts of associated metadata. Setting up a uniform schema and vocabulary for annotating these datasets as well as providing effective search tools helps developers find these sets in order to use them for new applications [6].

There are search engines for finding RDF and other datasets, such as ordinary search engines, SWSE [11], Swoogle[5], Watson[6], and others. However, using such systems based on the Google-like search paradigm it is difficult to get an idea of the *whole* cloud of the offered datasets. Furthermore, finding suitable datasets based on different selection criteria such as topic, size, licensing, publisher, language etc. is not supported. To facilitate this, interoperable metadata about the different aspects or facets of datasets is needed, and faceted search (also called view-based search) [19,9,12] can be used to provide an alternative paradigm for string-based semantic search.

This paper presents DataFinland, a semantic portal for creating, publishing, and finding datasets based on metadata. In contrast to systems like CKAN[7], the LOD-oriented voiD[8] (Vocabulary of Interlinked Datasets) metadata schema is used to describe datasets with property values taken from a set of shared domain ontologies providing controlled vocabularies with clearly defined semantics. Content is annotated using a web-based annotation tool SAHA 3[9] connected to ONKI ontology services[10] [22,21] that publish the domain ontologies. SAHA 3 has been integrated with the lightweight multifaceted search engine HAKO[11] [16], which facilitates automatically forming a faceted search and browsing application for taking in and discerning the datasets on offer. The annotation data itself is stored in RDF format, which makes combining the metadata about different datasets from different sources simple. This means that it would be possible to have several annotation projects for different sets of datasets, which could then be combined as needed for searching purposes. As a proof of concept, the system has been applied to describing the LOD cloud datasets and datasets in the Finnish Open Data Catalogue Project[12] complementing the linked open governmental datasets on a national level. The demonstration is available online[13] and the system received the first prize in this year's "Apps4Finland–Doing Good With Open Data" competition.

[4] http://www.data.gov/ and http://data.gov.uk/
[5] http://swoogle.umbc.edu/
[6] http://watson.kmi.open.ac.uk/WatsonWUI/
[7] http://www.ckan.net/
[8] http://semanticweb.org/wiki/VoiD
[9] http://www.seco.tkk.fi/services/saha/
[10] http://www.onki.fi/
[11] http://www.seco.tkk.fi/tools/hako/
[12] http://data.suomi.fi/
[13] http://demo.seco.tkk.fi/saha3sandbox/voiD/hako.shtml

In the following we will first present the general model and tools for creating and publishing metadata about (linked) datasets, and then discuss the voiD metadata schema and ontology repository ONKI presenting a controlled vocabulary. After this, the annotation tool SAHA for distributed semantic content creation is presented along with the faceted publication engine HAKO. In conclusion, the main contributions of the paper are listed, related work discussed, directions for future research proposed.

2 Overview of the Publication Process

Our solution for the process of producing metadata and publishing the annotated datasets is depicted in Figure 1. The process begins with the publication of a dataset. Metadata for the dataset is produced either by its original publisher or by a third party, using an annotation tool, in our case SAHA 3. A metadata schema, in our case modified voiD, is used to dictate for the distributed and independent content providers the exact nature of the metadata needed. Interoperability in annotation values is achieved through shared ontologies that are used for certain property values in the schema (e.g., subject matter and publisher resources are taken from corresponding ontologies). The ontologies are provided for the annotation tool as services, in our case by the national ONKI Ontology Service (or by SAHA itself). Finally, the metadata about the datasets is published in a semantic portal capable of using the annotations to make the data more accessible to the end-user, be that a human or a computer application. For this part the faceted search engine HAKO is used.

In the figure, we have marked the tools and resources used in our proof-of-concept system in parentheses, but the process model itself is generic.

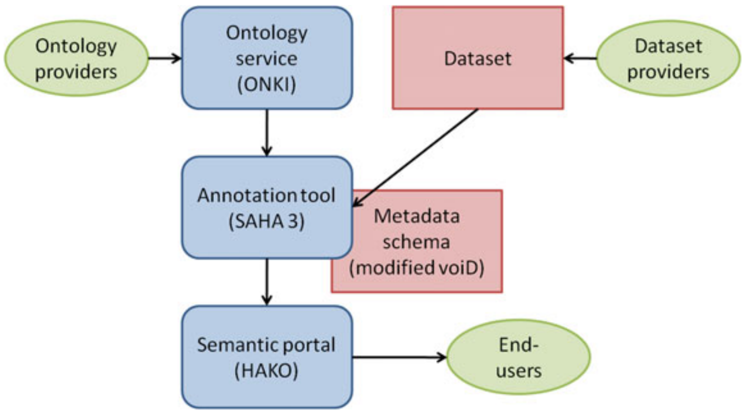


Fig. 1. The distributed process of producing and publishing metadata about (linked) datasets

3 Metadata and Ontologies

From a semantic viewpoint, the key ingredients of general model presented above are the metadata schema and domain ontologies/vocabularies used for filling in values in the schema. As for the metadata schema, the Vocabulary of Interlinked Datasets (voiD), an RDF vocabulary for describing linked datasets [1], seemed like a natural starting point because it addresses specifically problems of representing linked data. It was therefore chosen as a basis in our proof-of-concept system.

The basic component in voiD is a dataset, a collection of RDF triples that share a meaningful connection with each other in the form a shared topic, source or host. The different aspects of metadata that voiD collects could be classified into the following three categories or facets:

1. Descriptive metadata tells what the dataset is about. This includes properties such as the name of the dataset, the people and organizations responsible for it, as well as the general subject of the dataset. Here voiD reuses other, established vocabularies, such as `dcterms` and `foaf`. Additionally, voiD allows for the recording of statistics concerning the dataset.
2. Accessibility metadata tells how to access the dataset. This includes information on SPARQL endpoints, URI lookup as well as licensing information so that potential users of the dataset know the terms and conditions under which the dataset can be used.
3. Interlinking metadata tells how the dataset is linked to other datasets. This is done by defining a linkset, the concept of which is depicted in Figure 2. If dataset :DS1 includes relations to dataset :DS2, a subset of :DS1 of the type void:Linkset is made (:LS1) which collects all the triples that include links between the two datasets (that is, triples whose subject is a part of DS1 and whose object is a part of :DS2).

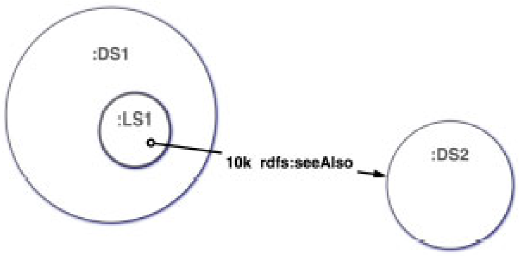


Fig. 2. Modeling interlinking of datasets in voiD [1]

3.1 Extending voiD

In order to facilitate annotating also non-linked open datasets, we made some extensions to voiD. The most important of these was a class for datasets in formats other than RDF. This `void-addon:NonRdfDataset` is similar to the `void:Dataset` but does not have the RDF-specific properties such as SPARQL endpoint while including a proprety for describing the format of the dataset, `void-addon:format`. The addition of this class also resulted in modifications to most of the voiD properties to include `void-addon:NonRdfDataset` in their domain specifications. Another addition to the basic voiD in our system was `dcterms:language` that facilitates the multi-language applications.

4 From Annotations to Faceted Search

Since the publishing of open data is not done by any central authority, annotating the data should also be collaborative and community-driven. To this end the annotation tools should be easy to use and publishing the results of the annotations should be quick and easy.

Our solution to facilitating collaborative annotation of distributed communities is based on the SAHA 3 metadata editor and the HAKO faceted search system [16]. In addition, we use the ONKI Ontology Service [21,22] for providing ontological concepts to annotations. These concepts, organized as hierarchical ontologies, also provide facets classifying the subject matter and some other aspects of the datasets in focus. Using ontologies instead of a free tagging system provides a controlled vocabulary with well defined meanings as well as support for multiple languages. Furthermore, the semantic relations can be used in further reasoning when the number of datasets gets very high.

4.1 SAHA 3 Metadata Editor

SAHA 3 is a metadata editor that allows for easy annotation of varied resources hiding the complexities of RDF and OWL from the end users. It is easily configurable to different schemas and supports distributed, simultaneous annotation in different locations, which is of paramount importance when using it in a community-driven environment such as Linked Open Data. It also functions in a normal web browser needing no special software to be installed.

The process of annotation itself is simple using SAHA 3. When a project has been configured, the annotator is shown the main view of a project, which gives a general overview of it. On the left side all the classes (as denoted by owl:Class) are listed, along with the count of how many instances of that class exist in the project. New instances can be created from this list as can be seen in Figure 3. The instances for any class can be browsed and filtered to find the desired ones.

The resource view, shown in Figure 4, is a basic overview of a resource. There is a resource view for each resource in the model. All the property values of the resource are listed, except those that are configured to be hidden. The data cannot be edited here - to do that the [edit] button must be pressed, which takes the user to the Annotation view.

When annotating an instance, the annotator is provided with a number of fields corresponding to the properties whose domain matches the class of the instance (see Figure 5). Depending on the range of a given property, the field takes in either free text or instances. In the latter case the instances can be either ones defined in the current project or chosen from linked ONKI ontologies. In both cases autocompletion[14][10] is used to aid the annotator.

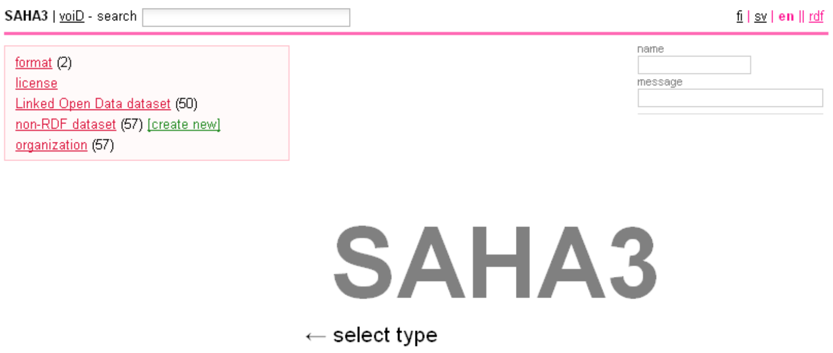
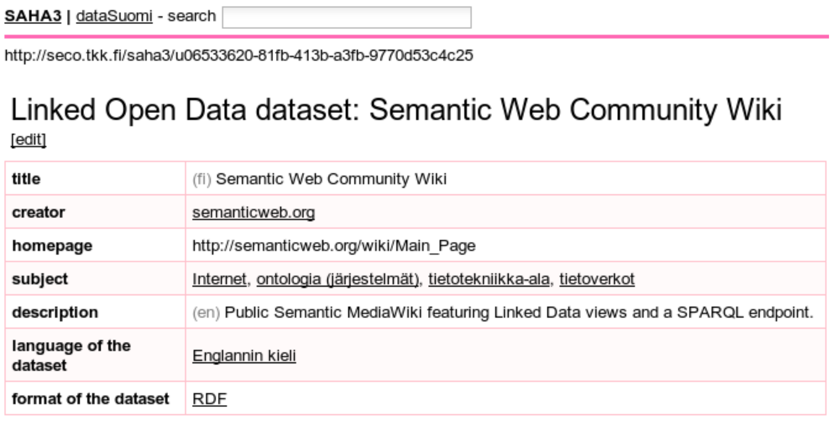


Fig. 3. Overview of a Saha project


SAHA3 | dataSuomi - search

http://seco.tkk.fi/saha3/u06533620-81fb-413b-a3fb-9770d53c4c25

Linked Open Data dataset: Semantic Web Community Wiki
[edit]

title	(fi) Semantic Web Community Wiki
creator	semanticweb.org
homepage	http://semanticweb.org/wiki/Main_Page
subject	Internet, ontologia (järjestelmät), tietotekniikka-ala, tietoverkot
description	(en) Public Semantic MediaWiki featuring Linked Data views and a SPARQL endpoint.
language of the dataset	Englannin kieli
format of the dataset	RDF

Fig. 4. Resource view of an instance in Saha

4.2 HAKO Faceted Search Engine

HAKO is a faceted search engine that can be used to publish a SAHA 3 project as a readily usable portal. The RDF data produced in SAHA 3 is exported into HAKO, which is then configured to produce a portal matching the needs of the end user. The publisher configures the classes whose instances are to be searched and whose properties form the search facets for these instances.

The end result is a semantic portal supporting both faceted search as well as free text search, which is done as a prefix search by default. For machine use, SAHA 3 also has a SPARQL endpoint[14] which can be used to access the metadata from the outside as a service instead of accessing the HAKO portal human interface. The SPARQL interface

[14] http://demo.seco.tkk.fi/saha/service/data/voiD/sparql?query={query}

can be used also internally in SAHA for providing semantic recommendation links between data objects on the human interface.

4.3 DataFinland

DataFinland is the name given for the whole solution of combining SAHA 3 and HAKO search portal with the extended voiD schema for creating, publishing, and finding datasets based on metadata.

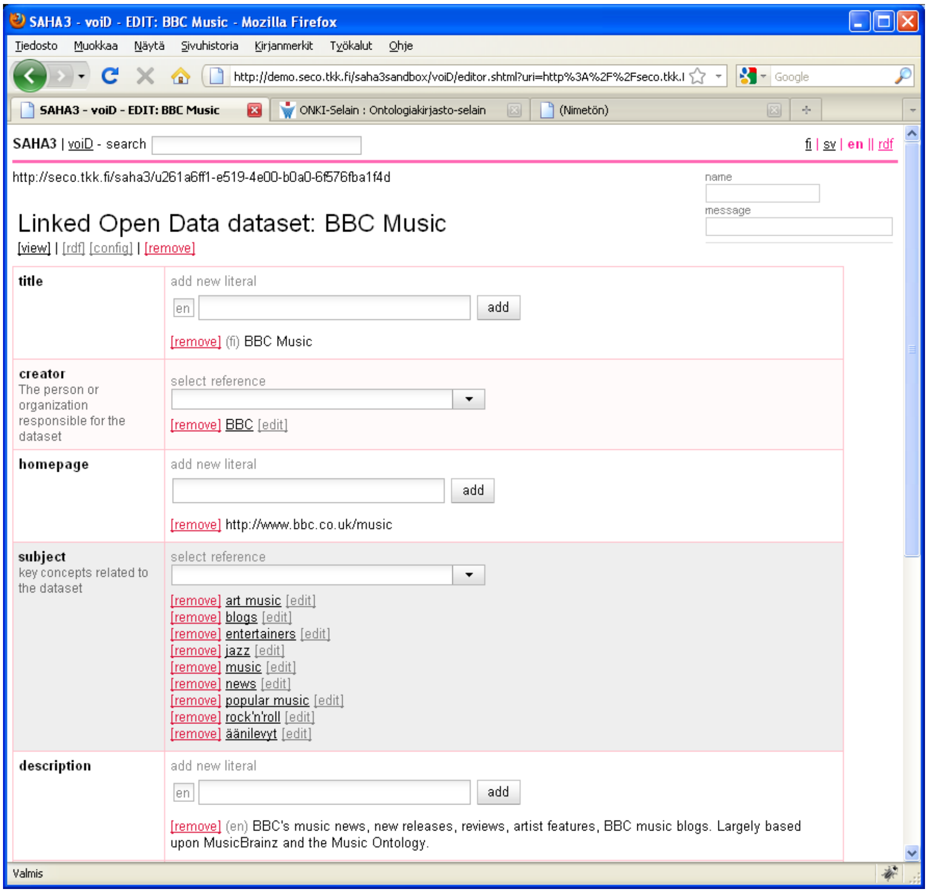
When configuring SAHA 3 for voiD, the `dcterms:subject` was connected to the ONKI instance of the General Finnish Ontology (YSO)[15] with over 20,000 concepts. The property `dcterms:license` was linked to an ONKI instance featuring six Creative Commons license types, but the system also allows for the defining of other license types as new instances of a simple license class. Its properties include of a free text description of the license as well as a possible link to a webpage describing the license further. Finally, `dcterms:language` was connected to the ONKI instance of the Lingvoj[16] vocabulary listing of the languages of the world.

The SAHA 3 annotation environment for voiD (depicted in Figure 5) allows for the annotation of both RDF and non-RDF datasets as well as licenses, formats and organizations. Licenses are additional licenses that the user may want to use aside from the ready linked Creative Commons licenses. Formats are simple resources to identify the format of the dataset, e.g. PDF, MS Word Document, etc. Finally, organizations allows for a simple way of describing an organization or a person responsible for a given dataset in the form of a title, free text description and a link to a homepage or a similar information source.

HAKO was configured to search for both RDF and non-RDF datasets and to form facets based on the license, language, format and subject properties. This way the end-user can, for example, limit his/her search to cover only Linked Open datasets by choosing the RDF format. In Figure 6 the user has selected from the facets on the left RDF datasets concerning Information technology industry in the English language. Out of the nine results provided by HAKO, the user has chosen Advogato to see its metadata.

A problem of faceted search with wide-ranging datasets is that facets tend to get very large, which makes category selection more difficult. A solution to this is to use hierarchical facets. However, using the hierarchy of a thesaurus or an ontology intended originally for annotations and reasoning may not be an optimal facet for information retrieval from the end-user's perspective [20]. For example, the top levels of large ontologies with complete hierarchies can be confusing for the end-users. Our planned solution in the future is to provide the annotators with a simple tool for building hierarchies for the facets as a part of the annotation process. Another possible solution would be to use some kind of an all-inclusive classification system as the top level of the facets. There has been some discussion of a classification schema for open datasets in the community, but no clear standard has risen. In the future we plan to explore the possibility of using the Finnish Libraries' classification system that is based on Dewey Decimal Classification.

[15] http://www.yso.fi/onki/yso/
[16] http://www.lingvoj.org/


SAHA3 - voiD - EDIT: BBC Music - Mozilla Firefox
Tiedosto Muokkaa Näytä Sivuhistoria Kirjanmerkit Työkalut Ohje
SAHA3 - voiD - EDIT: BBC Music
ONKI-Selain : Ontologiakirjasto-selain
(Nimetön)
Google
Valmis

http://demo.seco.tkk.fi/saha3sandbox/voiD/editor.shtml?uri=http%3A%2F%2Fseco.tkk.l

SAHA3 | voiD - search

fi | sv | en || rdf

http://seco.tkk.fi/saha3/u261a6ff1-e519-4e00-b0a0-6f576fba1f4d

name

message

Linked Open Data dataset: BBC Music

[view] | [rdf] [config] | [remove]

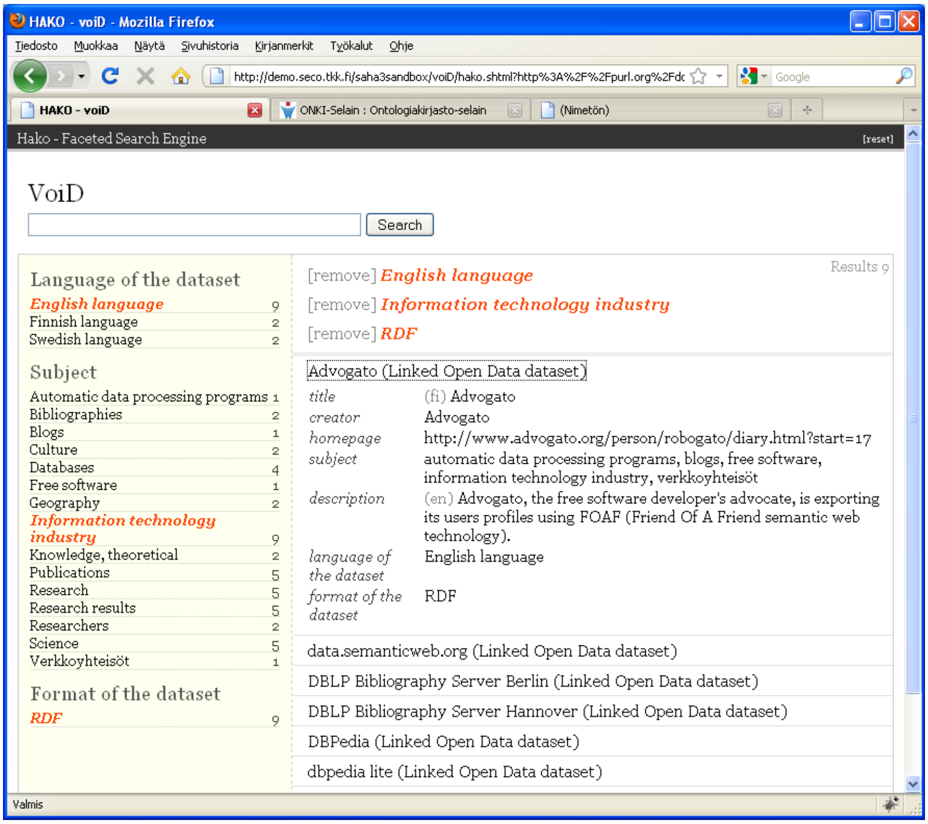
title	add new literal en add [remove] (fi) BBC Music
creator The person or organization responsible for the dataset	select reference [remove] BBC [edit]
homepage	add new literal add [remove] http://www.bbc.co.uk/music
subject key concepts related to the dataset	select reference [remove] art music [edit] [remove] blogs [edit] [remove] entertainers [edit] [remove] jazz [edit] [remove] music [edit] [remove] news [edit] [remove] popular music [edit] [remove] rock'n'roll [edit] [remove] äänilevyt [edit]
description	add new literal en add [remove] (en) BBC's music news, new releases, reviews, artist features, BBC music blogs. Largely based upon MusicBrainz and the Music Ontology.

Fig. 5. SAHA 3 annotation environment

5 Discussion

5.1 Contributions

This paper presented a distributed content creation model for metadata about datasets published on the web. The model emphasizes and supports the idea that metadata should be created in an interoperable way by the actors that publish the actual content. Making metadata interoperable afterwards is usually more difficult and costly. [13] In practice this requires support for using shared metadata schemas and domain ontologies/vocabularies, as well as a shared publication channel, a semantic portal. These facilities are provided in our model by the combination of ONKI, SAHA and HAKO tools.


HAKO - voiD - Mozilla Firefox
Tiedosto Muokkaa Näytä Sivuhistoria Kirjanmerkit Työkalut Ohje
http://demo.seco.tkk.fi/saha3sandbox/voiD/hako.shtml?http%3A%2F%2Fpurl.org%2Fdc
Google
HAKO - voiD
ONKI-Selain : Ontologiakirjasto-selain
(Nimetön)
Hako - Faceted Search Engine
[reset]
Search
Valmis

VoiD

Language of the dataset

English language	9
Finnish language	2
Swedish language	2

Subject

Automatic data processing programs	1
Bibliographies	2
Blogs	1
Culture	2
Databases	4
Free software	1
Geography	2
Information technology industry	9
Knowledge, theoretical	2
Publications	5
Research	5
Research results	5
Researchers	2
Science	5
Verkkoyhteisöt	1

Format of the dataset

RDF	9

Results 9

[remove] *English language*
[remove] *Information technology industry*
[remove] *RDF*

Advogato (Linked Open Data dataset)

title	(fi) Advogato
creator	Advogato
homepage	http://www.advogato.org/person/robogato/diary.html?start=17
subject	automatic data processing programs, blogs, free software, information technology industry, verkkoyhteisöt
description	(en) Advogato, the free software developer's advocate, is exporting its users profiles using FOAF (Friend Of A Friend semantic web technology).
language of the dataset	English language
format of the dataset	RDF

data.semanticweb.org (Linked Open Data dataset)
DBLP Bibliography Server Berlin (Linked Open Data dataset)
DBLP Bibliography Server Hannover (Linked Open Data dataset)
DBPedia (Linked Open Data dataset)
dbpedia lite (Linked Open Data dataset)

Fig. 6. HAKO faceted search portal

One of the main challenges of any model dealing with dataset metadata is to motivate dataset publishers to also publish semantically annotated metadata about their content. Our work is driven by the hope that this social challenge can be addressed by making annotating easy by online tools (such as SAHA and ONKI), and by providing the annotators with instant feedback on how their dataset is shown in the final semantic portal (HAKO).

5.2 Related Work

There is a number of tools available for creating voiD descriptions. The voiD editor ve[17] and liftSSM[18], an XSLT script that transforms a semantic sitemap in XML to voiD RDF/XML format, but these allow building only rudimentary descriptions, which should then be added to by manually editing the RDF file.

[17] http://ld2sd.deri.org/ve/
[18] http://vocab.deri.ie/void/guide#sec_4_3_Publishing_tools

As for datasets, there are a number of tools for finding Linked Open data. Semantic Web Search Engine[11] (SWSE) takes a free text approach allowing the user to enter a query string and returning entities from Linked Open datasets that match the query term. Searching for whole datasets is not supported.

Aside from search tools intended for human users, there is a number of search indexes intended for applications, including Sindice [18], Watson [5] and Swoogle [7]. These provide APIs supporting the discovery of RDF documents based on URIs or keywords. Sindice is intended for finding individual documents while Swoogle is used for finding ontologies. Watson allows the finding of all sorts of semantic data and features advanced filtering abilities intended for both machine and human users. However, none of these search engines are very good for exploring what sorts of datasets are available or for getting a whole picture of a given domain.

Governmental Open Data is widely published through CKAN[19] (Comprehensive Knowledge Archive Network), a registry for Open Data packages. CKAN provides support for publishing and versioning Open data packages and includes robust API support. However, the metadata about the data packages is recorded utlizing free tagging which does not support hierarchical, view-based search and does not contain semantic relation data between different tags.

Finally, concurrently to our work, an interoperability format for governmental data catalogues based on the dcat RDF vocabulary was proposed in [17]. There, the metadata schema was based on existing metadata used in the data catalogues as opposed to the LOD based voiD. Furthermore, this solution does not contain tools for editing metadata nor link to existing ontologies for use in dataset descriptions. A faceted search using Gridworks in combination with dcat was also proposed in [4].

The distributed semantic content creation and publishing approach, using shared metadata schemas, ontology services, and semantic portals for publication, has been originally developed in the semantic portals of the FinnONTO project [15].

5.3 Future Work

Our intention next is to propose the testing of the demonstrational system in the Finnish open data catalogue project. Another future application prospect is to apply the system for publishing metadata about scientific datasets for research. Additional distributed annotation-publishing projects can be opened with little extra work using the tools presented; proposals are solicited by the authors of this paper.

Acknowledgements

This work was conducted as a part of the National Semantic Web Ontology project in Finland[20] (FinnONTO, 2003-2012), funded mainly by the National Technology and Innovation Agency (Tekes) and a consortium of 38 public organizations and companies. Furthermore, we would like to thank Tuomas Palonen for his annotation work on the datasets for the demonstration and Petri Kola and Antti Poikola for fruitful discussions on publishing open datasets.

[19] http://www.ckan.net/

[20] http://www.seco.tkk.fi/projects/finnonto/

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: Conjunction with 18th International World Wide Web Conference (WWW 2009) Linked Data on the Web Workshop (LDOW 2009) (2009)
2. Bizer, C., Cyganiak, R., Heath, T.: How to publish linked data on the web (2007)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems, IJSWIS (2009)
4. Cyganiak, R., Maali, F., Peristeras, V.: Self-service linked government data with dcat and gridworks. In: Proceedings of the 6th International Conference on Semantic Systems, Graz, Austria. I-SEMANTICS 2010, pp. 37:1–37:3. ACM, New York (2010)
5. dÁquin, M., Motta, E.: Watson, more than a semantic web search engine (2010)
6. Dekkers, M., Polman, F., te Velde, R., de Vries, M.: Mepsir: Measuring european public sector information resources. final report of study on exploitation of public sector information. Technical report (2006)
7. Finin, T., Ding, L., Pan, R., Joshi, A., Kolari, P., Java, A., Peng, Y.: Swoogle: Searching for knowledge on the semantic web. In: AAAI 2005 (intelligent systems demo), pp. 1682–1683. The MIT Press, Cambridge (2005)
8. Hausenblas, M., Halb, W., Raimond, Y., Heath, T.: What is the size of the semantic web? In: Proceedings of I-SEMANTICS 2008, Graz, Austria (2008)
9. Hearst, M., Elliott, A., English, J., Sinha, R., Swearingen, K., Lee, K.-P.: Finding the flow in web site search. CACM 45(9), 42–49 (2002)
10. Hildebrand, M., van Ossenbruggen, J., Amin, A., Aroyo, L., Wielemaker, J., Hardman, L.: The design space of a configurable autocompletion component. Technical Report INS-E0708, Centrum voor Wiskunde en Informatica, Amsterdam (2007)
11. Hogan, A., Harth, A., Umrich, J., Decker, S.: Towards a scalable search and query engine for the web. In: WWW 2007: Proceedings of the 16th international conference on World Wide Web, pp. 1301–1302. ACM, New York (2007)
12. Hyvönen, E., Saarela, S., Viljanen, K.: Application of ontology techniques to view-based semantic search and browsing. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 92–106. Springer, Heidelberg (2004)
13. Hyvönen, E.: Preventing interoperability problems instead of solving them. In: Semantic Web Journal (2010) (accepted for pubication)
14. Hyvönen, E., Mäkelä, E.: Semantic autocompletion. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 739–751. Springer, Heidelberg (2006)
15. Hyvönen, E., Viljanen, K., Mäkelä, E., Kauppinen, T., Ruotsalo, T., Valkeapää, O., Seppälä, K., Suominen, O., Alm, O., Lindroos, R., Känsälä, T., Henriksson, R., Frosterus, M., Tuominen, J., Sinkkilä, R., Kurki, J.: Elements of a national semantic web infrastructure—case study finland on the semantic web. In: Proceedings of the First International Semantic Computing Conference (IEEE ICSC 2007). IEEE Press, Irvine (2007) (invited paper)
16. Kurki, J., Hyvönen, E.: Collaborative metadata editor integrated with ontology services and faceted portals. In: Workshop on Ontology Repositories and Editors for the Semantic Web (ORES 2010), The Extended Semantic Web Conference ESWC 2010, CEUR Workshop Proceedings, Heraklion, Greece (2010) http://ceur-ws.org
17. Maali, F., Cyganiak, R., Peristeras, V.: Enabling interoperability of government data catalogues. In: Wimmer, M.A., Chappelet, J.-L., Janssen, M., Scholl, H.J. (eds.) EGOV 2010. LNCS, vol. 6228, pp. 339–350. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-14799-9_29

18. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Tummarello, G.: Sindice.com: A document-oriented lookup index for open linked data. International Journal of Metadata, Semantics and Ontologies 3 (2008)
19. Pollitt, A.S.: The key role of classification and indexing in view-based searching. Technical report, University of Huddersfield, University of Huddersfield, UK (1998), `http://www.ifla.org/IV/ifla63/63polst.pdf`
20. Suominen, O., Viljanen, K., Hyvönen, E.: User-centric faceted search for semantic portals (2007)
21. Tuominen, J., Frosterus, M., Viljanen, K., Hyvönen, E.: ONKI SKOS server for publishing and utilizing SKOS vocabularies and ontologies as services. 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. 768–780. Springer, Heidelberg (2009)
22. Viljanen, K., Tuominen, J., Hyvönen, E.: Ontology libraries for production use: The finnish ontology library service ONKI. 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. 781–795. Springer, Heidelberg (2009)

Biological Names and Taxonomies on the Semantic Web – Managing the Change in Scientific Conception

Jouni Tuominen, Nina Laurenne, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University School of Science and the University of Helsinki
firstname.lastname@aalto.fi
http://www.seco.tkk.fi/

Abstract. Biodiversity management requires the usage of heterogeneous biological information from multiple sources. Indexing, aggregating, and finding such information is based on names and taxonomic knowledge of organisms. However, taxonomies change in time due to new scientific findings, opinions of authorities, and changes in our conception about life forms. Furthermore, organism names and their meaning change in time, different authorities use different scientific names for the same taxon in different times, and various vernacular names are in use in different languages. This makes data integration and information retrieval difficult without detailed biological information. This paper introduces a meta-ontology for managing the names and taxonomies of organisms, and presents three applications for it: 1) publishing biological species lists as ontology services (ca. 20 taxonomies including more than 80,000 names), 2) collaborative management of the vernacular names of vascular plants (ca. 26,000 taxa), and 3) management of individual scientific name changes based on research results, covering a group of beetles. The applications are based on the databases of the Finnish Museum of Natural History and are used in a living lab environment on the web.

1 Introduction

Exploitation of natural resources, urbanisation, pollution, and climate changes accelerate the extinction of organisms on Earth which has raised a common concern about maintaining biodiversity. For this purpose, management of information about plants and animals is needed, a task requiring an efficient usage of heterogeneous, dynamic biological data from distributed sources, such as observational records, literature, and natural history collections. Central resources in biodiversity management are names and ontological taxonomies of organisms [1,19,20,3,4]. Animal ontologies are stereotypical examples in the semantic web text books, but in reality semantic web technologies have hardly been applied to managing the real life taxonomies of biological organisms and biodiversity on the web. This paper tries to fill this gap.[1]

[1] We discuss the taxonomies of contemporary species, not 'phylogenetic trees' that model evolutionary development of species, where humans are successors, e.g., of dinosaurs.

Managing taxonomies of organisms provides new challenges to semantic web ontology research. Firstly, although we know that lions are carnivores, a subclass of mammals that eat other animals, the notion of 'species' in the general case is actually very hard to define precisely. For example, some authors discuss as many as 22 different definitions of the notion of species [16]. Secondly, taxonomic knowledge changes and increases due to new research results. The number of new organism names in biology increases by 25,000 every year as new taxa to science are discovered [11]. At the same time, the rate of changes in existing names has accelerated by the implementation of molecular methods suggesting new positions to organisms in taxonomies. Thirdly, biological names are not stable or reliable identifiers for organisms as they or their meaning change in time. Fourthly, the same name can be used by different authors to refer to different taxa (units of classification that commonly have a rank in the hierarchy), and a taxon can have more than one name without a consensus about the preferred one.

As a result, biological texts are written, content is indexed in databases, and information is searched for using different names and terms from different times and authorities. In biological research, scientific names are used instead of common names, but in many applications vernacular names in different languages are used instead. Data fusion is challenging and information retrieval without deep biological knowledge is difficult.

We argue that a shared system for publishing and managing the scientific and vernacular names and underlying conceptions of organisms and taxonomies is needed. From a research viewpoint, such a system is needed to index research results and to find out whether a potential new species is already known under some name. Biological information needed by environmental authorities cannot be properly indexed, found or aggregated unless the organism names and identifiers are available and can be aligned. For amateur scientists and the public, aligning vernacular names to scientific names and taxonomies is often a prerequisite for successful information retrieval.

This paper presents a meta-ontology and its applications addressing these problems. Our research hypothesis is that semantic web technologies are useful in practise in modelling change in the scientific perception of biological names and taxonomies, for creating a platform for collaboratively managing scientific knowledge about taxonomies, and for publishing taxonomies as ontology services for indexing and information retrieval purposes in legacy systems.

In the following, biological classification systems are first discussed and a meta-ontology TaxMeOn for defining such systems is presented [13]. Three use case applications of the meta-ontology are then discussed: a system for managing vascular plant names collaboratively (26,000 species) based on the SAHA metadata editor [12], application of the ONKI ontology service [25] for publishing taxonomic species lists on the semantic web (over 80,000 taxa of mammals, birds, butterflies, wasps, etc.), and a more focused application for managing the names and scientific findings of the Afro-tropical beetle family Eucnemidae. Finally, contributions of our work are summarised, related work discussed, and directions for further research are outlined.

2 Biological Names and Taxonomies

The scientific name system is based on the Linnean binomial name system where the basic unit is a species. Every species belongs to some genus and every genus belongs to a higher taxon. A scientific name often has a reference to the original publication where it was first published. For example, the scientific name of the bumblebee, *Apis mellifera* Linnaeus, 1758, means that Linnaeus published the description of the bumblebee in 1758 (in Systema Naturae 10th edition) and that bumblebee belongs to the genus *Apis*. The upper levels of the taxonomic hierarchy do not show in a scientific name. A confusing feature of scientific names is that the meaning of the name may change although the name remains the same. Taxon boundaries may vary according to different studies, and there may be multiple simultaneous views of taxon limits of the same organism group. For example, a genus may be delimited in three ways and according to each view different sets of species are included in the genus as illustrated in Fig. 1. These differing views are *taxonomic concepts*. The usage of the correct name is not enough, and Berendsohn [1] suggested that taxonomic concepts should be referred to by an abbreviation *sec* (*secundum*) after the authors name to indicate in which meaning the name is used.

The nature of a biological name system is a change, as there is no single interpretation of the evolution. Typically there is no agreement if the variation observed in an organism is taxon-specific or shared by more than one taxon, which makes the name system dynamic. For example, the fruit fly *Drosophila melanogaster* was shifted into the genus *Sophophora*, resulting in a new name combination *Sophonophora melanogaster* [7]. The most common taxonomic changes and their implications to the scientific names are the following: 1) A species has been shifted to another genus - the genus name changes. 2) One species turns out to be several species - new species are described and named, and the old name remains the same with a narrower interpretation. 3) Several species are found to be just one species - the oldest name is valid and the other names become its synonyms.

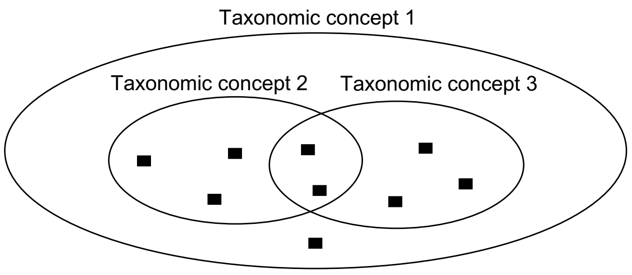


Fig. 1. A genus is delimited in three different ways according to three different studies. Black squares indicate species.

Species lists catalogue organisms occurring in a certain geographical area, which may vary from a small region to global. Often species lists contain valid taxon names with author information and synonyms of the valid names. They are snapshots of time and used especially by environmental authorities. The problem with species lists is that not all organism groups are catalogued and changes are not necessarily recorded in the species lists. Traditionally printed lists tend to be more detailed than online lists and their status is higher.

Species lists often follow different hierarchies and species may be associated with different genera according to the person who published the list. The hierarchy in a species list is a compromise that combines several studies, and the author can subjectively emphasise a view that he/she wishes. A taxon may also have different taxonomic ranks in literature, for example the same taxon can occur both as a species and a subspecies.

Common names tend to have regional variation and they do not indicate hierarchy unlike scientific names. Vernacular names have an important role in everyday language, but due to the variation and vagueness, they have little relevance in science. Vernacular names are used mainly in citizen science.

3 TaxMeOn – Meta-ontology of Biological Names

We have developed a meta-ontology for managing scientific and vernacular names. The ontology model consists of three parts that serve different purposes: 1) name collections, 2) species lists, and 3) name changes resulting from research. These parts are manageable separately, but associations between them are supported. Being a meta-ontology, TaxMeOn defines classes and properties that can be used to build ontologies. The ontologies can be used for creating semantic metadata for describing e.g. observational data or museum collections. TaxMeOn is based on RDF using some features of the OWL. The model contains 22 classes and 53 properties (61 including subproperties), of which ten classes and 15 properties are common to all the three parts of the model[2].

The core classes of TaxMeOn express a taxonomic concept, a scientific name, a taxonomic rank, a publication, an author, a vernacular name, and a status of a name. Taxonomic ranks are modelled as classes, and individual taxa are instances of them, for example the species forest fir *forrestii* (belongs to the genus *Abies*) is an instance of the class *Species*. The model contains 61 taxonomic ranks, of which 60 are obtained from TDWG Taxon Rank LSID Ontology[3]. In order to simplify the management of subspecific ranks, an additional class that combines species and taxonomic levels below it was created.

References embody publications in a broad sense including other documented sources of information, for instance minutes of meetings. Bibliographic information can be associated to the reference according to the Dublin Core metadata standard. In biology, author names are often abbreviated when attached to taxon

[2] The TaxMeOn schema is available at http://schema.onki.fi/taxmeon/

[3] http://rs.tdwg.org/ontology/voc/TaxonRank

names. The TaxMeOn model supports the referring system that is typical to biology. Some of the properties used in TaxMeOn are part-specific as the uses of the parts differ from each other. For instance, the property that refers to a vernacular name is only available in the name collection part as it is not relevant in the other parts of the model.

The most distinctive feature of the research part [14] is that a scientific name and taxonomic concepts associated to it are separated, which allows detailed management of them both. In the name collection and species list parts, a name and its taxonomic concepts are treated as a unit. Different statuses can be associated to names, such as validity (accepted/synonym), a stage of a naming process (proposed/accepted) and spelling errors.

The model has a top-level hierarchy that is based on a rough classification, such as the division of organism *classes* and *orders*. Ontologies that are generated using TaxMeOn, can be hung on the top-level classification. A hierarchy is created using the transitive *isPartOfHigherTaxon* relation, e.g. to indicate that the species *forrestii* belongs to the genus *Abies*.

Taxon names that refer to the same taxon can occur as different names in the published species lists and different types of relations (see Table 1) can be set between the taxa. Similarly, research results of phylogenetic studies can be mapped using the same relations. The relations for mapping taxa are divided on the basis of attributes of taxa (*intensional*) or being a member of a group (*ostensive*). If it is known that two taxa have an association which is not specified, a class is provided for expressing incomplete information (see the empty ellipse in Fig. 2). This allows associations of taxa without detailed taxonomic knowledge, and especially between taxa originating from different sources.

Table 1. Mapping relations used in species lists and research results. The three relations can be used as intensional and/or ostensive, using their subproperties.

Relation	**Description**
congruent with taxon	taxonomic concepts of two taxa are equal
is part of taxon	a taxonomic concept of a taxon is included in a taxonomic concept of another taxon
overlaps with taxon	taxonomic concepts of two taxa overlap

In TaxMeOn, a reference (an author name and a publication year) to the original publication can be attached to a name. A complete scientific name is atomised into units that can be combined in applications by traversing the RDF graph by utilising the *isPartOfHigherTaxon* and *publishedIn* relations.

Name collections. Scientific names and their taxonomic concepts are treated as one unit in the name collection, because the scope is in vernacular names. The model supports the usage of multiple languages and dialects of common names. There may be several common names pointing to the same taxon, and typically one of them is recommended or has an official status. Alternative names are expressed defining the status using the class *VernacularNameStatus* and references related to the changes of a name status can be added. This allows the tracking the temporal order of the statuses. The model for vernacular names is illustrated in Fig. 2.

Species lists. Species lists have a single hierarchy and they seldom include vernacular names. Species lists have more relevance in science than name collections, but they lack information about name changes and a single list does not express the parallel or contradictory views of taxonomy which are crucial for researchers. Synonyms of taxa are typically presented and the taxonomic concept is included in a name like in a name collection. Taxa occurring in different species lists can be mapped to each other or to research results using the relations in Table 1. In addition, a general association without taxonomic details can be used (see Fig. 2).

Biological research results. In biological research results a key element is a taxonomic concept that can have multiple scientific names (and vice versa). Instead of names, taxonomic concepts are used for defining the relations between taxa. The same relations are applied here as in the speies list part (see Table 1). The latest research results often redefine taxon boundaries, for example a split of taxa narrows the original taxonomic concept and the meaning of the name changes although the name itself may remain the same. The new and the old concepts are connected into a temporal chain by instantiation of a change event. In Fig. 3 the concept of the beetle genus Galba is split into the concepts of the Balgus and Pterotarsus. The taxon names are shown inside the ellipses

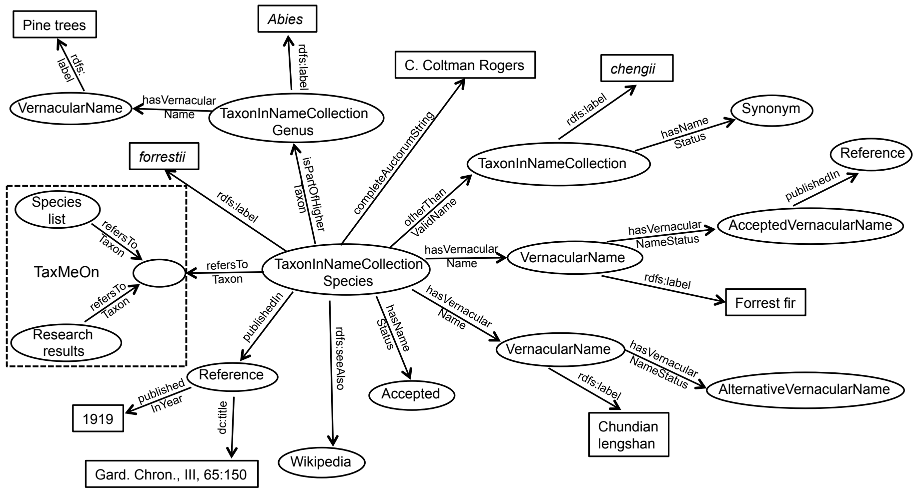


Fig. 2. An example of vernacular names in a name collection. The ellipses represent instances of TaxMeOn classes and literals are indicated as boxes. Other parts of the model are connected to the example taxon in the box with dotted line, in which the empty ellipse illustrates a general representation of a taxon.

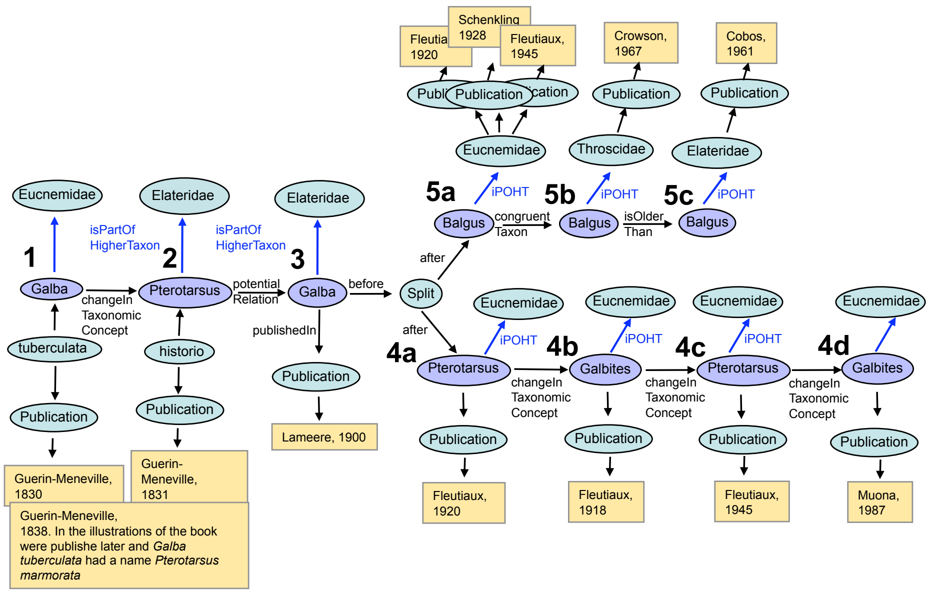


Fig. 3. An example of name changes and taxonomies of eucnemid beetles based on research results. The ellipses represent instances of TaxMeOn classes. Taxonomic hierarchies are expressed with the *isPartOfHigherTaxon* (iPOHT) relations, and the name change series of taxa are illustrated with a darker colour. The following abbreviations are used for the change types: S = Split of taxa, NC = Name change, TCC = Taxon concept change and CH = Change in hierarchy. The meaning of the numbers: 1) The species description of *Galba tuberculata* was originally published in 1830, but the illustrations of the book were published in 1838. However, in the illustrations *G. tuberculata* appeared with the name *Pterotarsus marmorata* (conflicting information). 2) Meanwhile, in 1831, the same taxon was independently described as *Pterotarsus historio* (independent events). 3) Lameere was confused by the two independently published works and changed the name to *Galba* in 1900 (uncertain relation between 2 and 3). 4a) Fleutiaux split the genus *Galba* into two genera. The name *Galba* was changed into *Pterotarsus* as there turned out to be a crustacean genus *Galba* (S, NC,TCC). 4b) Fleutiaux re-examined the genus and concluded that it is new to science and described it as *Galbites* (NC, TCC). 4c) Later Fleutiaux changed his mind and renamed the genus as *Pterotarsus* again (NC, TCC). 4d) Muona discovered that Fleutiaux was originally right and renamed the genus as *Galbites* (NC, TCC). 5a) When *Galba* was split, a part of its species were shifted into the genus *Balgus* that was described as new to science at the same time. *Balgus* was placed in the family Eucnemidae (CH). 5b) And changed into the family Throscidae (CH). This was originally published in a monthly magazine in the 1950's, but the magazines were published as a book in 1967 which is most commonly cited. 5c) *Balgus* was changed into the family Elateridae in 1961 (CH and conflict in publication years).

representing taxonomic concepts in order to simplify the presentation. Other change types are a lump of taxa, a change in taxon boundaries and a change in a hierarchy. These changes lead to the creation of a new instance of a taxonomic concept in order to maintain the traceable taxon history. An instantion of a new concept prevents evolving non-existing name combinations and artificial classifications. For instance, a species name is not associated with a genus name in which it has never been included.

The status of a scientific name may change in time as an accepted name may become a synonym. Multiple statuses can be attached to a name, but according to the nomenclatural rules only one of them is accepted at time. The temporal order of the statuses can be managed according to the same idea as in the name collections part.

4 Use Cases

We have applied the TaxMeOn ontology model to three use cases that are based on different needs. The datasets include a name collection of common names of vascular plants, several species lists of different animal groups and a collection of biological research results of Afro-tropical beetles. The use cases were selected on the basis of the active usage of the data (vernacular names), usefulness to the users (species lists), and the taxonomic challenges with available expertise (scientific names based on research results). The datasets used are depicted in Table 2.

4.1 Collaborative Management of Vascular Plants Names

The biological name collection includes 26,000 Finnish names for vascular plants that are organised into a single hierarchy. A deeply nested hierarchy is not necessary here as the classification used is robust, containing only three taxonomic ranks. The need is to maintain the collection of the common names and to manage the name acceptance process. The number of yearly updates exceeds 1,000. The typical users of the name collection are journalists, translators and other non-biologists who need to find a common name for a scientific name.

The name collection of vascular plants is managed in SAHA[4] [12]. SAHA is a simple, powerful and scalable generic metadata editor for collaborative content creation. Annotation projects can be added into SAHA by creating the metadata schema for the content and loading it into SAHA. The user interface of SAHA adapts to the schema by providing suitable forms for producing the metadata. The values of the properties of the schema can be instances of classes defined in the schema, references to ontologies or literals. The annotations created using SAHA are stored in a database, from which they can be retrieved for use in semantic applications. SAHA also provides a SPARQL endpoint for making queries to the RDF data.

[4] http://demo.seco.tkk.fi/saha/VascularPlants/index.shtml

Table 2. Datasets TaxMeOn has been applied to. Vascular plants are included in the name collection, the false click beetles are biological research results, and all other datasets are based on species lists.

Taxon group	**Region**	**Publ. years**	**# of taxa**
Vascular plants	World	constantly updated	25726
Long-horn beetles (Coleoptera: Cerambycidae)	Scandinavia, Baltic countries	1939, 1960, 1979, 1992, 2004, 2010, 2010	205, 181, 247, 269, 300, 297, 1372
Butterflies and moths (Lepidoptera)	Scandianavia, North-West Russia, Estonia	1962, 1977, 1996, 2002, 2008	313, 256, 265, 4573, 12256, 3244, 3251, 3477
Thrips (Thysanoptera)	Finland	2008	219
Lacewings and scorpionflies (Neuroptera and Mecoptera)	Finland	2008	113
True bugs (Hemiptera)	Finland	2008	2690
Flies (Diptera: Brachycera)	Finland	2008	6373
Parasitic wasps (Hymenoptera: Ichneumoidae)	Finland	1995, 1999, 1999, 2000, 2003	282, 398, 919, 786, 733
Bees and wasps (Hymenoptera: Apoidea)	Finland	2010	1048
Mammals	World	2008	6062
Birds	World	2010	12125
False click beetles (Coleoptera: Eucnemidae)	Afrotropics	–	9 genera

New scientific species names are added by creating a new instance of the *Species* class and then adding the other necessary information, such as their status. Similarly, a higher taxon can be created if it does not already exist, and the former is linked to the latter with the *isPartOfHigherTaxon* relation. SAHA has search facilities for querying the data, and a journalist writing a non-scientific article about a house plant, for example, can use the system for finding a common name for the plant.

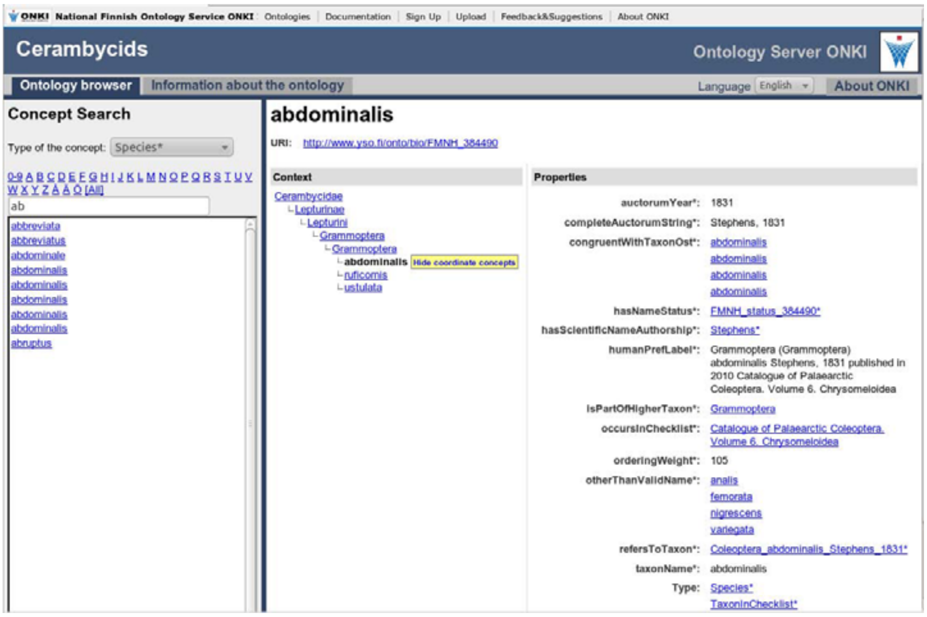
4.2 Publishing Species Lists as Ontology Services

The users of species lists are ecologists, environmental authorities and amateurs searching for the correct scientific name occurring in a certain geographical area. In this use case ca. 20 published species lists obtained from the taxonomic database of the Finnish Museum of Natural History[5] containing more than 80,000 names were converted into TaxMeOn ontologies. In addition, seven regional lists of long-horn beetles (cerambycids) with 100 species are available from the years 1936–2010. The various names meaning the same taxon were mapped by an expert. The most common differences between the lists are a shift of a genus for a species, a change in a hierarchy and/or in a name status. Similarly, ca. 150 species of butterfly names from five lists were mapped.

[5] http://taxon.luomus.fi/

Currently, the mapped beetle names are published as services for humans and machines in the ONKI Ontology Service[6] [25]. The ONKI Ontology Service is a general ontology library for publishing ontologies and providing functionalities for accessing them, using ready-to-use web widgets as well as APIs. ONKI supports content indexing, concept disambiguation, searching, and query expansion.

Fig. 4 depicts the user interface of the ONKI server [24]. The user is browsing the species lists of cerambycid beetles, and has made a query for taxon names starting with a string "ab". The selected species *abdominalis* has been described by Stephens in 1831, and it occurs in the species list Catalogue of Palaearctic Coleoptera, published in the year 2010 [15]. The species *abdominalis* belongs to the subgenus and genus *Grammoptera*. The taxonomy of the family Cerambycidae is visualised as a hierarchy tree. The same species also occurs in other species lists, which is indicated by *congruentWithTaxonOst* relation. Browsing the taxa reveals varying taxon names and classifications. For example, the *Grammoptera (Grammoptera) abdominalis* has a subgenus in this example, but the rank subgenus does not exist in the other lists of cerambycid. Also, the synonyms of the selected taxon are shown (*analis*, *femorata*, *nigrescens* and *variegata*).



Type of the concept: Species*

ab

abbreviata
abbreviatus
abdominale
abdominalis
abdominalis
abdominalis
abdominalis
abdominalis
abruptus

abdominalis

URI: http://www.yso.fi/onto/bio/FMNH_384490

Context

```
Cerambycidae
 └Lepturinae
   └Lepturini
     └Grammoptera
       └Grammoptera
         └abdominalis
         └ruficornis
         └ustulata
```

Properties

auctorumYear*:	1831
completeAuctorumString*:	Stephens, 1831
congruentWithTaxonOst*:	abdominalis abdominalis abdominalis abdominalis
hasNameStatus*:	FMNH_status_384490*
hasScientificNameAuthorship*:	Stephens*
humanPrefLabel*:	Grammoptera (Grammoptera) abdominalis Stephens, 1831 published in 2010 Catalogue of Palaearctic Coleoptera. Volume 6. Chrysomeloidea
isPartOfHigherTaxon*:	Grammoptera
occursInChecklist*:	Catalogue of Palaearctic Coleoptera. Volume 6. Chrysomeloidea
orderingWeight*:	105
otherThanValidName*:	analis femorata nigrescens variegata
refersToTaxon*:	Coleoptera_abdominalis_Stephens_1831*
taxonName*:	abdominalis
Type:	Species* TaxonInChecklist*

Fig. 4. The species of *abdominalis* shown in the ONKI Browser

The ONKI Ontology Services can be integrated into applications on the user interface level (in HTML) by utilising the ONKI Selector, a lightweight web widget providing functionalities for accessing ontologies. The ONKI API has

[6] http://demo.seco.tkk.fi/onkiskos/cerambycids/

been implemented in three ways: as an AJAX service, as a Web Service, and as a simple HTTP API.

The ONKI Ontology Service contains several ontologies covering different fields and is a part of the FinnONTO project [6] that aims to build a national ontology infrastructure. The Finnish Spatio-temporal Ontology (SAPO) [8], for example, can be used to disambiguate geographical information of observational data. Combining the usage of species ontologies and SAPO, extensive data harmonisation is avoided as both taxon names and geographical names change in time.

4.3 Management of Individual Scientific Names

The use case of scientific names is the Afro-tropical beetle family Eucnemidae, which consists of ca. nine genera that have gone through numerous taxonomic treatments. Also, mistakes and uncertain events are modelled if they are relevant to name changes. For example, the position of the species *Pterotarsus historio* in taxonomic classification has changed 22 times and at least eight taxonomic concepts are associated to the genus *Pterotarsus* [17]. Fig. 3 illustrates the problematic nature of the beetle group in a simplified example. A comparable comparable situation concerns most organism groups on Earth. Due to the numerous changes in scientific names, even researchers find it hard to remember them and this information can only be found in publications of taxonomy. The option of managing individual names is advantageous as it completes the species lists and allows the mapping of detailed taxonomic information to the species lists. For example, environmental authorities and most biologists prefer a simple representation of species lists instead of complicated change series.

5 Discussion

We have explored the applicability of the semantic web technologies for the management needs of biological names. Separating taxonomic concepts from scientific and vernacular names is justified due to the ambiguity of the names referring to taxa. This also enables relating relevant attributes separately to a concept and to a name, although it is not always clear to which of these an attribute should be linked and subjective decisions have to made. The idea of the model is simplicity and practicality in real-world use cases.

The fruitfulness lays in the possibilities to link divergent data serving divergent purposes and in linking detailed information with more general information. For example, a common name of a house plant, a taxonomic concept that appears to be a species complex (a unit formed by several closely related species) and the geographical area can be linked.

The most complex use case is the management of scientific name changes of biological research results. The main goal is to maintain the temporal control of the name changes and classifications. The instantiation of taxon names and concepts lead to a situation in which they are hard to manage when they form a long chain. Every change increases the number of instances created. Protegé[7] was used for editing the ontologies, although managing names is quite inconvenient because they are shown as an alphabetically ordered flat list, not as a taxonomic hierarchy.

As Protegé is rather complicated for a non-expert user, the metadata editor SAHA was used for maintaining the continuous changes of common names of plants. The simplicity of SAHA makes it a suitable option for ordinary users who want to concentrate on the content. However, we noticed that some useful features are missing from SAHA. The visualisation of a nested hierarchy would help users to compare differing classifications.

In many biological ontologies the 'subclass of' relation is used for expressing the taxon hierarchies. However, in the TaxMeOn model we use the *isPartHigherTaxon* relation instead. If the 'subclass of' relation was used to express the taxonomic hierarchy, a taxon would incorrectly be an instance of the higher taxon ranks, e.g., a species would be an instance of the class *Genus*. This would lead to a situation in which queries for genera also return species.

5.1 Related Work

NCBO BioPortal[8] and OBO Foundry[9] have large collections of life science ontologies mainly concentrating on biomedicine and physiology. The absence of taxonomic ontologies is distinctive which may indicate the complexity of the biological name system. The portals contain only three taxonomic ontologies (Amphibian taxonomy, Fly taxonomy and Teleost taxonomy) and one broader classification (NCBI organismal classification). The taxonomic hierarchy is defined using the *rdfs:subClassOf* relation in the existing ontologies. Taxonconcept.org[10] provides Linked Open Data identifiers for species concepts and links data about them originating from different sources. All names are expressed using literals and the following taxonomic ranks are included: a combination of a species and a genus, a class and an order. Parallel hierarchies are not supported. Geospecies[11] uses the properties *skos:broaderTransitive* and *skos:narrowerTransitive* to express the hierarchy.

Page [19] discusses the importance of persistent identifiers for organism names and presents a solution for managing names and their synonyms on the semantic web. The taxon names from different sources referring to the same taxon are mapped using the *owl:sameAs* relation which is a strong statement. Hierarchy is expressed using two different methods in order to support efficient queries.

Schulz et al. [20] presented the first ontology model of biological taxa and its application to physical individuals. Taxa organised in a hierarchy is thoroughly discussed, but the model is static and based on a single unchangeable taxonomy.

[7] http://protege.stanford.edu/
[8] http://bioportal.bioontology.org/
[9] http://www.obofoundry.org/
[10] http://www.taxonconcept.org/
[11] http://lod.geospecies.org/

Despite recognising the dynamic nature of taxonomy and the name system, the model is not applicable in the management of biological names as such.

Franz and Peet [3] enlighten the problematic nature of the topic by describing how semantics can be applied in relating taxa to each other. They introduce two essentially important terms from philosophy to taxonomy to specify the way, in which differing classifications that include different sets of taxa can be compared. An ostensive relation is specified by being a member of a group and intensional relations are based on properties uniting the group. These two fundamentally different approaches can be used simultaneously, which increases the information content of the relation.

Franz and Thau [4] developed the model of scientific names further by evaluating the limitations of applying ontologies. They concluded that ontologies should focus either on a nomenclatural point of view or on strategies for aligning multiple taxonomies.

Tuominen et al. [23] model the taxonomic hierarchy using the *skos:broader* property, and preferred scientific and common names of the taxa are represented with the property *skos:prefLabel* and alternative names with *skos:altLabel*. The property *rdf:type* is used to indicate the taxonomic rank. This is applicable to relatively simple taxonomies such as species lists, but it does not support expressing more elaborate information (changes in a concept or a name).

The Darwin Core (DwC) [2] is a metadata schema developed for observation data by the TDWG (Biodiversity Information Standards). The goal of the DwC is to standardise the form of presenting biological information in order to enhance the usage of it. However, it lacks the semantic aspect and the terms related to biological names are restricted due to the wide and general scope of the DwC.

The scope of the related work presented above differs from our approach as our focus is on practical name management and retrieval of names.

Research on ontology versioning [10] and ontology evolution [18] has focused on finding mappings between different ontology versions, performing ontology refinements and other changes in the conceptualisation [9,21], and in reasoning with multi-version ontologies [5]. There are similarities in our problem field, but our focus is to support multiple parallel ontologies interpreting the domain differently, not in versioning or evolution of a specific ontology. For example, there is no single taxonomy of all organisms, but different views of how they should be organised into hierarchies.

A similar type of an approach for managing changes and parallel views of concepts has been proposed by Tennis and Sutton [22] in the context of SKOS vocabularies. However, TaxMeOn supports richer ways of expressing information, e.g. for managing changes of taxon names and concepts separately.

5.2 Future Work

The model will be tested using different datasets to ensure its applicability. Currently, the research results part covers animal names, but will be expanded to plant names as well. The lack of user-friendly tools is obvious and the metadata editor SAHA is planned to be expanded to respond to the needs. Describing evolutionary trees and their information content is a challenging application area as phylogenetics produces name changes.

Acknowledgements. This work is part of the National Semantic Web Ontology project in Finland[12] (FinnONTO, 2003-2012), funded mainly by the National Technology and Innovation Agency (Tekes) and a consortium of 38 organizations. We thank Jyrki Muona, Hans Silfverberg, Leo Junikka and Juhana Nieminen for their collaboration.

References

1. Berendsohn, W.: The concept of ”potential taxon” in databases. Taxon 44, 207–212 (1995)
2. Darwin Core Task Group. Darwin core. Tech. rep (2009), http://www.tdwg.org/standards/450/
3. Franz, N., Peet, R.: Towards a language for mapping relationships among taxonomic concepts. Systematics and Biodiversity 7(1), 5–20 (2009)
4. Franz, N., Thau, D.: Biological taxonomy and ontology development: scope and limitations. Biodiversity Informatics 7, 45–66 (2010)
5. Huang, Z., Stuckenschmidt, H.: Reasoning with multi-version ontologies: A temporal logic approach. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 398–412. Springer, Heidelberg (2005)
6. Hyvönen, E., Viljanen, K., Tuominen, J., Seppälä, K.: Building a national semantic web ontology and ontology service infrastructure – the FinnONTO approach. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 95–109. Springer, Heidelberg (2008)
7. ICZN. Opinion 2245 (case 3407) drosophila fallén, 1823 (insecta, diptera): Drosophila funebris fabricius, 1787 is maintained as the type species. Bulletin of Zoological Nomenclature 67(1) (2010)
8. Kauppinen, T., Väätäinen, J., Hyvönen, E.: Creating and using geospatial ontology time series in a semantic cultural heritage portal. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 110–123. Springer, Heidelberg (2008)
9. Klein, M.: Change Management for Distributed Ontologies. Ph.D. thesis, Vrije Universiteit Amsterdam (August 2004)
10. Klein, M., Fensel, D.: Ontology versioning on the Semantic Web. In: Proceedings of the International Semantic Web Working Symposium (SWWS), July 30 – August 1, pp. 75–91. Stanford University, California (2001)
11. Knapp, S., Polaszek, A., Watson, M.: Spreading the word. Nature 446, 261–262 (2007)
12. Kurki, J., Hyvönen, E.: Collaborative metadata editor integrated with ontology services and faceted portals. In: Workshop on Ontology Repositories and Editors for the Semantic Web (ORES 2010), the Extended Semantic Web Conference ESWC 2010, CEUR Workshop Proceedings, Heraklion, Greece (June 2010), http://ceur-ws.org/

[12] http://www.seco.tkk.fi/projects/finnonto/

13. Laurenne, N., Tuominen, J., Koho, M., Hyvönen, E.: Modeling and publishing biological names and classifications on the semantic web. In: TDWG 2010 Annual Conference of the Taxonomic Databases Working Group (September 2010); poster abstract
14. Laurenne, N., Tuominen, J., Koho, M., Hyvönen, E.: Taxon meta-ontology TaxMeOn – towards an ontology model for managing changing scientific names in time. In: TDWG 2010 Annual Conference of the Taxonomic Databases Working Group (September 2010); contributed abstract
15. Löbl, I., Smetana, A.: Catalogue of Palearctic Coleoptera Chrysomeloidea, vol. 6. Apollo Books, Stenstrup (2010)
16. Mayden, R.L.: A hierarchy of species concepts: the denouement in the saga of the species problem. In: Claridge, M.F., Dawah, H.A., Wilson, M.R. (eds.) Species: The Units of Biodiversity Systematics Association Special, vol. 54, pp. 381–424. Chapman and Hall, London (1997)
17. Muona, J.: A revision of the indomalesian tribe galbitini new tribe (coleoptera, eucnemidae). Entomologica Scandinavica. Supplement 39, 1–67 (1991)
18. Noy, N., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems 6(4) (2004)
19. Page, R.: Taxonomic names, metadata, and the semantic web. Biodiversity Informatics 3, 1–15 (2006)
20. Schulz, S., Stenzhorn, H., Boeker, M.: The ontology of biological taxa. Bioinformatics 24(13), 313–321 (2008)
21. Stojanovic, L.: Methods and Tools for Ontology Evolution. Ph.D. thesis, University of Karlsruhe, Germany (2004)
22. Tennis, J.T., Sutton, S.A.: Extending the simple knowledge organization system for concept management in vocabulary development applications. Journal of the American Society for Information Science and Technology 59(1), 25–37 (2008)
23. Tuominen, J., Frosterus, M., Laurenne, N., Hyvönen, E.: Publishing biological classifications as SKOS vocabulary services on the semantic web. In: TDWG 2010 Annual Conference of the Taxonomic Databases Working Group (September 2010); demonstration abstract
24. Tuominen, J., Frosterus, M., Viljanen, K., Hyvönen, E.: ONKI SKOS server for publishing and utilizing SKOS vocabularies and ontologies as services. 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. 768–780. Springer, Heidelberg (2009)
25. Viljanen, K., Tuominen, J., Hyvönen, E.: Ontology libraries for production use: The finnish ontology library service ONKI. 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. 781–795. Springer, Heidelberg (2009)

An Approach for More Efficient Energy Consumption Based on Real-Time Situational Awareness

Yongchun Xu, Nenad Stojanovic, Ljiljana Stojanovic,
Darko Anicic, and Rudi Studer

FZI Research Center for Information Technology
76131 Karlsruhe, Germany
name.familyname@fzi.de

Abstract. In this paper we present a novel approach for achieving energy efficiency in public buildings (especially sensor-enabled offices) based on the application of intelligent complex event processing and semantic technologies. In the nutshell of the approach is an efficient method for realizing the real-time situational awareness that helps in recognizing the situations where a more efficient energy consumption is possible and reaction on those opportunities promptly. Semantics allows a proper contextualization of the sensor data (i.e. its abstract interpretation), whereas complex event processing enables the efficient real-time processing of sensor data and its logic-based nature supports a declarative definition of the situations of interests. The approach has been implemented in the iCEP framework for intelligent Complex Event Reasoning. The results from a preliminary evaluation study are very promising: the approach enables a very precise real-time detection of the office occupancy situations that limit the operation of the lighting system based on the actual use of the space.

Keywords: Energy Efficiency, Complex Event Processing, Semantic Technology, Office Occupancy Control.

1 Introduction

Real-time processing has become very important for sensor-based applications, since the quantity of data being generated from sensors requires on–the-fly processing and immediate reaction in order to be effective. There are many examples, starting from item-tracking in RFID-supported logistics to remote patient monitoring in eHealth. Indeed, real-time awareness enables the detection of problems (e.g. a damaged item in a delivery, or an acute health problem in a patient) as soon as they happen, so that the reaction can be successfully performed. Note that the same mechanism can be used for preventive reactions, i.e. reacting before a problem would happen.

In the nutshell of this mechanism is the ability to recognize in real-time[1] (or even ahead of time) some interesting situations, what is called "real-time situational awareness". Note that this goes beyond the traditional (static) situational awareness

[1] We consider "business real-time" as the criteria for declaring something to be processed in real-time.

(like in [1]) that is focused on the **understanding a situation** (if possible in real-time). Real-time situational awareness introduces the notion of real-time emergency: the main goal is to **recognize a situation** of interest as soon as possible in order to be able to react on it properly.

On the other hand, such a process introduces several challenges for the processing of sensor-data:

a) it should be very efficient in order to retain its "real-time" flavor and

b) it should be very flexible in order to deal with various and dynamically changing patterns (situations) of interests (that should be recognized in real-time).

Complex event processing is a technology that can resolve these challenges.

Energy efficiency is one of application areas, where real-time situational awareness can bring a substantial added value. Smart Grid is a well-known example: Smart meters[2] enable real-time publishing of information about energy consumption of a user (usually once every seven minutes), which consequently can support the real-time optimization of the energy production. Decreasing energy consumption in buildings, public spaces and households is another very promising area for applying real-time situational awareness. It has been shown that the creation of the awareness about the current (in real-time) energy consumption in a household can bring itself up to 20% savings in electricity bills. Beside these passive savings, active energy savings by switching off electrical devices in particular situations are very common methods in reducing energy consumption[3].

Although well developed, current approaches for achieving energy efficiency seem to be "inefficient": they are usually dealing with customized solutions tuned to the predefined energy consumption scenarios. The consequence is that the costs for introducing and maintaining these solutions are quite high, since e.g. each new scenario (the so-called energy consumption pattern) has to be modeled separately. On the other hand, due to a high variation in the energy consumption profile (e.g. the spatial distribution of energy consumers), energy consumption patterns are very variable and modeling new patterns is more a rule than an exception. Therefore, although oriented towards real-time recognition of the interesting situations, these approaches are suffering from the inflexibility in the detection process, e.g. by not having available a declarative description of the situations to be detected and by not performing an intelligent processing (reasoning) on the incoming data. Obviously, the application of semantic technologies can be very promising for resolving these challenges.

In this paper we present a novel approach for achieving energy efficiency that exploits real-time situational awareness based on the use of Complex Event Processing and Semantic Technologies. The main idea is to enable a semantic-based description of the situations of interests (i.e. energy consumption patterns) and perform reasoning about those situations in real-time. The approach leverages on our work in the domain of intelligent Complex Event Processing (iCEP)[4], especially complex event reasoning, that combines a very efficient in-memory processing (on the fly) of a huge amount of streaming data and the reasoning (on the fly) using available domain knowledge.

[2] http://www.smartmeters.com/faqs.html

[3] http://www.greentrac.com/

[4] See iCEP.fzi.de

The approach has been implemented using the iCEP framework and deployed in our experimental environment that supports testing novel technologies with a higher users' involvement. We have performed a case study related to the office occupancy control that limits the operation of the lighting system based on the actual use of the space. Preliminary evaluation tests have shown very promising results regarding the usability and the efficiency of the approach: the approach is able to abstract from particular patterns to be recognized into general/declarative situations to be reasoned about.

The paper is structured in the following way:

In the second section we give more details about our Energy Efficiency use case, from the real-time consumption point of view. In the third section we outline the architecture of our solution. In section four we describe some evaluation details, whereas section five elaborates briefly on the related work. In section six we give some concluding remarks.

2 Energy Efficiency in Offices: State of the Art and Requirements Analysis

There are several studies which show that up to 70% of the energy consumption in an office environment can be saved just by "creating awareness" about the current energy consumption. This is especially related to the very low electricity consumption of the equipment and lighting. For example, while according to Logicalis[5], 94% of workers surveyed turn their lights off at home, only 66% thought about doing the same at work. Turning the lights off all too often gets ignored by offices, whose lights continue to shine even after everyone has gone home. On the other hand, many of the energy saving activities can be automated, like "whenever workers leave a room, the lights should be turned off" (see more examples in the Evaluation section).

There are several approaches which are dealing with the automation of this process, which are usually taking into account the following factors for the lighting issues:

- control of the user's presence in an office, as a necessary condition to turn on the light;
- regulation of the artificial light, in relation to the natural light level;
- possibility of a manual regulation of the light, forcing the automatic regulation, in order to better meet the user's needs.

Therefore, the automation of the energy saving process is related to an efficient sensing of the current situation in an office and reacting in particular situations. The best examples are the so-called **occupancy controls** that limit the operation of the lighting system based on the actual use of the space. Unlike scheduling controls, they do not operate by a pre-established time schedule. Instead, the system senses when the space is occupied and turns the lights on. When the system senses that there has been no activity in the space, it assumes the space is unoccupied and turns the lights off. To

[5] http://www.energysavingsecrets.co.uk/HowToRunAnEnergyEfficientOffice.html

prevent the system from turning the lights off while the space is still occupied but there is very little activity, a time delay typically ranging from 1 to 15 minutes can be programmed into the controls.

However, as already mentioned in the Introduction, current approaches are based on a fix set of rules that are designed for a particular scenario. Let's illustrate this on the example depicted in Figure 1: an office with six desks, each equipped with a lamp. The task is to enable switching off the corresponding lamp when a person is leaving the room. The **most advanced systems** would model several patterns that describe situations that a person is leaving the room which implies the need for switching off the light at her/his desk. A pattern for the region F would be (cf. Figure 1):

If sequence (RegionF, RegionD, RegionB, Door) Then SwitchOff(LampF) (1)

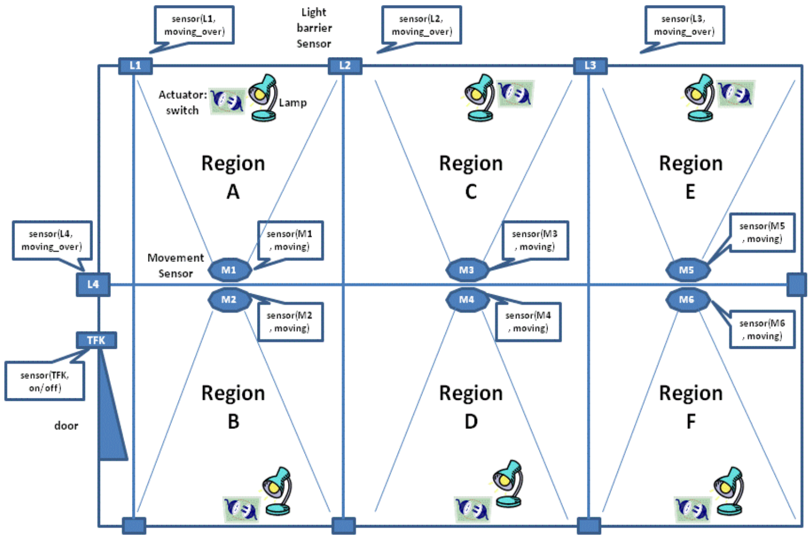


Fig. 1. The distribution of sensors in a smart office. There are three types of sensors: 1) contact sensors (TFK - attached to the door), 2) moving sensors (M1 – M6) and Light barrier sensors (L1-L4)

The main drawbacks of such an approach are:

1) the energy saving patterns are "hard-coded", which means that all patterns must be explicitly defined in order to be taken into account;
2) there is no abstraction in the pattern definition, so that any kind of generalization is excluded;
3) the patterns are "static", so that any kind of changes in the initial setting cannot be realized easily.

Consequently, the requirements for a novel approach are related to:

1) a declarative description of energy saving patterns in order to enable an abstract definition of the situations to be recognized;

2) the usage of domain knowledge in order to support the use of implicit information in the detection process;
3) the management of patterns, including their evolution.

In the rest of the paper we present such an approach.

3 iCEP Approach for the Energy Efficiency

In order to resolve the above mentioned problems, we have developed a logic-based approach for the automatic control of the energy consumption in an office scenario. In the nutshell of the approach is a very efficient complex event processing supported by deductive reasoning capabilities, implemented in the ETALIS engine [2]. The data processed by ETALIS are coming from sensors placed in the office (cf. Figure 1). Sensors are semantically described using ontologies. In the rest of this section we present the information model behind the approach and give more details about the underlying processing system.

3.1 Information Model

Figure 2 represents the information model used in the proposed approach.

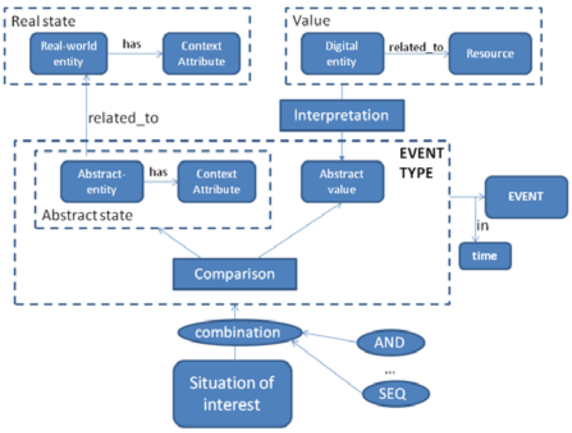


Fig. 2. Information model of the system

It consists of three layers: the *raw data* provided by sensors, the *Digital Entities* provided by sensor resources and the *context information* provided by the advanced system components or context-level resources.

The raw data consists of the value that the sensor provides, e.g., the numerical value 25. A resource may augment this information with meta-information, e.g. that the measured value is a temperature that it is in degrees Celsius that it was measured by sensor X at a certain point in time etc. We call the resulting information Digital Entity.

This information is not yet contextualized, i.e., we may know that it is a temperature, but not what this temperature describes, e.g., is this the indoor temperature of a room or the temperature within the fridge? This information is modeled by the context information. The real world is modeled as Real-world entities, which are further described by the context attributes. The Real-world entities have an identifier and an entity type. The entity type, e.g., person, place, or car, etc. defines the context attributes of an entity of interest. For example, a person can have a blood pressure; whereas a car can have a certain fuel level. The context attributes have an attribute name, e.g., "hasIndoorTemperature", an attribute type, e.g., "temperature" and a context value. The context value consists of a Digital Entity plus some quality of information parameters. This quality of information may be different from the quality information that was provided by the observation and measurement, e.g., the accuracy for a room temperature may be calculated as a function of the accuracy of the Digital Entity and the reliability that the temperature sensor provides the temperature of the room.

In order to deal with situations we introduce Abstract Entities that correspond to the abstraction of the Real-world entities in a given application context. For example, a Region in Figure 1 is an Abstract entity. Similarly to a Real-world entity, an Abstract entity is associated to some Context attribute, which has some Abstract value. Note that Abstract value is a matter of the interpretation of the raw data (see Figure 2).

Finally, a situation of interests is a combination of several Abstract states and Values that represent the so-called Event types. Combinations are based on event processing operators (AND, OR, SEQ, etc. [3].

Events represent instantiations of Event types in real time. Those events are processed by ETALIS[6] engine.

The sensor system takes advantages of using semantic technologies. We use an ontology that is specified in W3C RDF/XML standard format to manage the sensor information and the background domain knowledge (see section 3.4). Our sensor system integrates ETALIS engine, which has the ability to perform stream reasoning with the background knowledge (ontologies) to achieve a high processing performance.

3.2 ETALIS

ETALIS Language for Events [2] is a logic-based formalism for Complex Event Processing (CEP) and Stream Reasoning. It uses SWI-Prolog Semantic Web Library[7] to represent an RDF/XML ontology as a set of Prolog rules and facts. ETALIS is an open-source implementation of the language. The language and a corresponding implementation are based on a novel event processing strategy which detects complex events by maintaining the intermediate states. Every time an atomic event (relevant w.r.t. the set of monitored events) occurs, the system updates the internal state of complex events. Essentially, this internal state encodes what atomic events are still missing for the completion a certain complex event. Complex events are detected as

[6] http://code.google.com/p/etalis/

[7] http://www.swi-prolog.org/pldoc/package/semweb.html

soon as the last event required for their detection has occurred. Descriptions telling which occurrence of an event drive the detection of complex events (including the relationships between complex events and events they consist of) are given by deductive rules. Consequently, detection of complex events then amounts to an inference problem.

Event processing formalisms based on deductive or logic rules [4, 5, 6] have been attracting considerable attention as they feature formal, declarative semantics. Declarative semantics of a CEP system prescribe what the system needs to detect, i.e., a user does not need to worry how that will be detected. In this respect declarative semantics guarantees predictability and repeatability of results produced by a CEP system. Moreover, CEP systems based on deductive rules can process not only events, but also any additional background knowledge relevant with respect to the detection of complex situations in real-time. Hence a rule-based approach enables a high abstraction level and a uniform framework for realizing knowledge-based CEP applications (i.e., specification of complex event patterns, contextual knowledge, and their interaction). Such applications can be further supported by machine learning (more specifically data mining) tools, to automate the construction and refinement of event patterns (see [7]). Although the machine learning support per se is out of scope of this paper, we want to emphasize the importance of the formal, rule-based semantics which can further enable automated construction of both, event patterns and the background knowledge. These features are beyond capabilities of existing approaches [8, 9, 10], and this is one of reasons why ETALIS follows a logic rule-based approach for event processing.

In the following, we identify a number of benefits of the ETALIS event processing model, realized via deductive rules: First, a rule-based formalism (like the one we present in this paper) is expressive enough and convenient to represent diverse complex event patterns. Second, a formal deductive procedure guarantees the correctness of the entire event processing. Unlike reactive rules (production rules and ECA rules), declarative rules are free of side-effects; the order in which rules are evaluated is irrelevant. Third, although it is outside the scope of this paper, a deductive rule representation of complex events may further help in the verification of complex event patterns defined by a user (e.g., by discovering patterns that can never be detected due to inconsistency problems). Further on, ETALIS can also express responses on complex events, and reason about them in the same formalism [11]. Fourth, by maintaining the state of changes, the ETALIS event model is also capable of handling queries over the entire space (i.e. answering queries that span over multiple ongoing detections of complex events). Ultimately, the proposed event model allows for reasoning over events, their relationships, entire state, and possible contextual knowledge available for a particular domain (application). Reasoning in the ETALIS event model can be further exploited to find ways to reach a given aim, which is a task that requires some intelligence. For example, an application or a service needs to reach a stable or known (desired) state. To achieve this, the system has to have a capability to reason about, or to asses states (in a changing environment). Another example is to just "track and trace" the state of any entity at any time (in order to be able to "sense and respond" in a proactive way).

Technically, ETALIS approach is based on the decomposition of complex event patterns into intermediate patterns (i.e. goals). The status of achieved goals is materialized as first class citizens of a fact base. These materialized goals show the progress toward completion of one or more complete event patterns. Such goals are automatically asserted by rules as relevant events occur. They can persist over a period of time "waiting" in order to support detection of a more complex goal or complete pattern. Important characteristics of these goals are that they are asserted only if they are used later on (to support a more complex goal or an event pattern), that goals are all unique and persist as long as they remain relevant (after that they can be deleted). Goals are asserted by rules which are executed in the backward chaining mode. The notable property of these rules is that they are event-driven. Hence, although the rules are executed backwards, overall they exhibit a forward chaining behavior. For more information, an interested reader is referred to [2].

3.3 Example

As already mentioned, one of the main advantages of our approach is the possibility to define the situations of interests in a declarative way and reason about them based on the incoming sensor data.

In order to illustrate the abstractions introduced by ETALIS, we present here a very illustrative example for the occupancy control based on the office context presented in Figure 1.

In the traditional approaches, the situation of interests:

"*a person left the room and her/his desk lamp should be switched off within 5 sec*" must be described by using one rule for each possible situation. An example is illustrated in Figure 3: a person left the room by traversing from Region F, through Region D and B till the door.

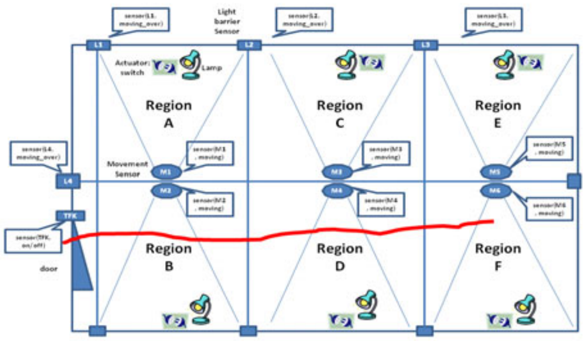


Fig. 3. A possible path from the office desk to the door: a situation that can lead to switching off the lamp at the desk in the Region F

Therefore, traditional approaches must cover use all possible "evacuation" paths which is a tedious and error prone process. The situation is even worse when we consider that the distribution of objects in the office can be changed – the whole set of rules must be rewritten.

On the other hand, in our approach there is only one logic-based statement that covers all requested situations, by describing them declaratively:

```
Namespace: cep: http://www.icep.fzi.de/cepsensor.owl#
Pattern.event:

door_open <- status('cep:door', 'cep:door_opened').

status(A, B) <- sensor(X, Y)
        WHERE
        (rdfs_individual_of(Sensor, 'cep:Sensor'),
        rdf(Sensor, 'cep:hasName', X),
        rdf(State, 'cep:hasValue', Y),
        rdfs_individual_of(State, 'cep:State'),
        rdf(B, 'cep:detectedWithState', State),
        rdfs_individual_of(B, 'cep:Status'),
        rdf(Sensor, 'cep:locatedIn', A)).

movement(Loc1,Loc2) <- status(Loc1, 'cep:movementInRegion') SEQ status(Bord,
'cep:moveover') SEQ status(Loc2, 'cep:movementInRegion')
        WHERE
        (rdfs_individual_of(Loc1, 'cep:Region'),
        rdfs_individual_of(Loc2, 'cep:Region'),
        rdfs_individual_of(Bord, 'cep:Borderline'),
        rdf(Loc1, 'cep:hasNeighbor', Loc2),
        rdf(Loc1, 'cep:hasBorderline', Bord),
        rdf(Loc2, 'cep:hasBorderline', Bord))2sec.
```

comment: this statement detects the situation that a person has changed the region, if within 2 sec the movement sensor and light barrier sensor for a Region has been activated

```
movement(Loc1,Loc3) <- movement(Loc1,Loc2) SEQ movement(Loc2,Loc3) .
movement(Loc, 'cep:door') <- (movement(Loc, 'cep:regionB') SEQ status(Bord,
'cep:moveover')) 2sec.
```

comment: this statement is the most crucial one: by introducing recursive rules we are able to describe all possible paths which are containing succeeding regions

```
SwitchOff(Loc) <- (movement(Loc, 'cep:door') SEQ door_open)5sec.
```

comment: this statement detects the situation that a person has left the room (after a sequence of traversing between regions) and that after 5 sec the light at the starting location should be switched off

Note that the particular state of the world (like this presented in Figure 2) is represented in the domain ontology and the CEP engine (ETALIS) is accessing that knowledge in the real time.

rdf(***?Subject, ?Predicate, ?Object***)
is a function in SWI-Prolog Semantic Web Library . It is an Elementary query for triples. Subject and Predicate are atoms representing the fully qualified URL of the resource. Object is either an atom representing a resource or literal (Value) if the object is a literal value.

rdfs_individual_of(*?Resource, ?Class*)
is a function in SWI-Prolog Semantic Web Library. It tests whether the *Resource* is an individual of *class*. It returns true if *Resource* is an individual of *Class*. This implies *Resource* has an rdf:type property that refers to *Class* or a sub-class thereof. It can be used to test, to generate classes *Resource* belongs to or to generate individuals described by *Class*.

3.4 Domain Ontology

The structure of the ontology used in our scenario is shown in Figure 4.

The S*ensor* class hierarchy on the top left models the sensors of the scenario such as *contact sensor*, *movement sensor* and *light barrier sensor*. Each sensor type has one or several states, which describe the physical sensing information of sensors, e.g. the contact sensor has two states: *on* or *off.* The states of sensors are modeled by using the **class** S*tate.*

The *Actuator* class hierarchy on the top right of the Figure 4, similar to the *Sensor* class hierarchy, models all actuators. Each Actuator has some processes, modeled in the **class** *Process* and defining the functions of the actuators. For example, the S*witch* has two processes corresponding to the switch on and switch off functions.

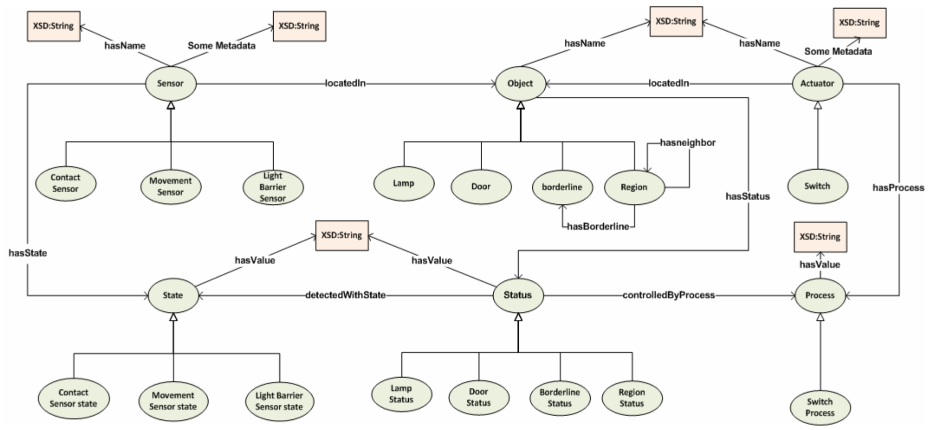


Fig. 4. Illustration of the main concepts in the domain ontology

The O*bject* hierarchy describes the real world entities such as *Lamp*, *Door* and *Region,* which are connected to a sensor or an actuator. The object property *locatedIn* describes the connection between the *Objects* and S*ensors* or A*ctuators.* Each object has several statuses e.g. D*oor* has two statuses: *open* and *closed.* Some of these statuses can be detected by *Sensor* with the special *State;* the others are controlled by *Actuator* by using related *Process.*

This ontology is used as background knowledge by ETALIS engine. Indeed, ETALIS allows using background knowledge in the detection process - any constraint can be easily associated to each situation, which enables a very easy generation of new occupancy situations that should be detected. For example, it is very easy to introduce a new property of a region in an office, like to treat regions that have a window separately from other regions.

4 Evaluation

In order to evaluate the performance of the proposed system we have implemented a test case concerning efficient energy consumption in an office. We have used the FZI Living Lab[8] environment for the testing.

The use case is based on simulating occupancy control situations that limit the operation of the lighting system based on the actual use of the space. In other words, if there is a situation, that leads to possibly saving energy, being recognized in a way specified in Section 3.3, the corresponding lighting source should be either dimmed or switched off. In order to make the test realistic we have implemented the set of energy consumption patterns developed for a Building Energy Challenge[9]. Table 1 represents some of those patterns. Note that in order to be realistic we assume that there are negative and positive situations from the energy efficiency point of view, depicted as Penalties and Bonus in Table 1 (the setting has been completely taken from the Building Energy Challenge).

Table 1. Examples of the consumption patterns from the Building Energy Challenge

Penalties	Bonus
Having a window open while the heating system is on	Switch off the light each time when leaving the office
Leaving the office at the end of the day with the computer switched on	Switch off the heating each time when none is in the office
Switch on the artificial light while day light is sufficient	Switch off the computer when leaving the office for more than one hour
Having a temperature lower than 26 °C with the air conditioning on [12]	Switch off the artificial light while day light is sufficient

We have modeled all these patterns using ETALIS language and ontologies as discussed in Section 3. The setting of the sensors was very similar to that presented in Figure 1 (additional sensors for measuring temperature, light intensity and actuators for electric devices have been introduced).

We performed an experiment in order to measure savings in the energy consumption. We have measured the power saving time in the period of one month in an office with five people. We find this setting as a very common one. As already explained, our declarative approach doesn't depend on the number of sensors and the

[8] Living Labs is a practical approach to realize open innovation with a regional dimension. By definition, it is "research methodology for sensing, validating and refining complex solutions in real life contexts". It is conceptualized as an innovation platform that brings together and involves a multitude of actors, such as end-users, researchers, industrialists and policy makers. They crucial characteristic is that they are user-centered with an active participation of users within the entire development process. Usually, Living Labs build upon or create a technology platform geared to answer the needs of users in a particular situation.

[9] A contest regarding energy consumption between several office buildings within a company, see: http://www.artist-embedded.org/docs/Events/2010/GREEMBED/0_GREEMBED_Papers/IntUBE 20- 20GREEMBED.pdf

size of the room. We performed several changes in the layout of the room (position of sensors) but without the need to change the complex event patterns. Therefore, the abstraction provided by our language is correct: interesting situations are defined on the level of objects, independently from the current position of sensors.

Table 2 presents the results from this experiment. In the last column we present the average value of measurements and in the rest of the columns the values for four particular days (1st, 10th, 20th and 30th) in order to illustrate how theses consumption values varied.

Power saving time represents the time when some electric devices were switched off because of the situation that the corresponding person (related to that device) had left the room.

We are quite satisfied with the general result of the experiment: the proposed approach leads to significant reductions in the energy consumption. We didn't encounter any example of the false positive.

Table 2. The results from the experiment

	1st day	10th day	20th day	30th day	**average**
Power saving time:	33089s	19548s	58133s	42152s	**38255s**
Total time:	109199s	103665s	124676s	117021s	**111354s**
Proportion:	30.3%	18.9%	46.6%	36%	**34.3%**
Error:	12	13	18	10	**13**
Total switch:	67	56	59	56	**62**
Error rate	17.9%	23.2%	30.5%	17.9%	**18.3%**

The only problem we have faced is the rather huge error rate, whereas an error represents the number of situations that couldn't be detected by using currently deployed patterns (out of scope of the experiment). In the following we give an explanation (i.e. interfere factors) of these situations.

The first interfere factor is the precision of the sensors. In the evaluation we used ELV FS20 sensor systems including FS20 PIRI-2 motion sensor, FS20 IR light barrier sensor, FS20 TFK contact sensor and FS20 ST-3 radio electrical socket. The motion sensor and the light barrier sensor have a minimal send time interval of 8 seconds, which means they can only send a single value every 8 seconds. In the case of a high activity frequency, the sensors can't detect all activities. Furthermore, the sensors can't detect some situations such as two people come into the office together. In this situation the sensors are not able to recognize the number of the people and only one lamp will be switched on. To overcome this, we can use more sensors and the better sensors to increase the precision of the event detection.

The second interfere factor is unanticipated activity in the office. For example, a user forgets to close the door after coming into the office. Then when another user leaves the office, he doesn't need to open the door, which is a necessary event according to the pattern. In this situation, the lamp will also not be switched off. Similarly, a visitor has visited the office, when he leaves the office, one lamp in the office will be falsely switched off. This problem can be overcome by installing automatic door closing device and using new sensor technologies (such as RFID) to recognize the identity of the user.

The third interfere factor results from the fact that the pattern definition doesn't match the character of a user. In the pattern we have defined that the movement event and door open event must happen within 5 seconds to trigger the switch off event. If a user is accustomed to do something else costing more than 5 seconds before he opens the door, then his lamp will not be switched off. The problem can be solved by doing some study on the characters of the users before defining the patterns.

Modeling the above mentioned situations will be one of the subjects of the further work.

5 Related Work

In this section we only present the related work related to the current lighting control systems. Related work to our approach for complex event processing can be found in [2].

Current lighting and climate control systems often rely on building regulations that define maximum occupancy numbers for maintaining proper lighting and temperatures. However, in many situations, there are rooms that are used infrequently, and may be lighted, heated or cooled needlessly. Having knowledge regarding occupancy and being able to accurately predict usage patterns may allow significant energy-savings.

In [13], the authors reported on the deployment of a wireless camera sensor network for collecting data regarding occupancy in a large multi-function building. They constructed multivariate Gaussian and agent based models for predicting user mobility patterns in buildings.

In [14], the authors identified that the majority of this energy waste occurs during the weekdays, not during the weeknights or over the weekends. They showed that this pattern of energy waste is particularly suited to be controlled by occupancy sensors, which not only prevent runaway operation after typical business hours, but also capture savings during the business day.

An analysis of the impact of the new trends in energy efficient lighting design practices on human comfort and productivity in the modern IT offices is given in [14].

In [15], the authors presented the design and implementation of a presence sensor platform that can be used for accurate occupancy detection at the level of individual offices. The presence sensor is low-cost, wireless, and incrementally deployable within existing buildings.

An examination of different types of buildings and their energy use is given in [16]. The authors discussed opportunities available to improve energy efficient operation through various strategies from lighting to computing.

As a conclusion, there are many approaches for the lighting control, but none of them is using a more declarative approach that would enable an efficient real-time situation detection.

6 Conclusions

In this paper we presented a novel approach for achieving energy efficiency in public buildings (especially sensor-enabled offices) based on the application of intelligent complex event processing and semantic technologies. In the nutshell of the approach is an efficient method for realizing real-time situational awareness that helps in recognizing the situations where a more efficient energy consumption is possible and reaction on those opportunities promptly. Semantics allows a proper contextualization of the sensor data (its abstract interpretation). Complex event processing enables the efficient real-time processing of sensor data and its logic-based nature supports a declarative definition of the situations of interests.

The approach has been implemented using iCEP framework and deployed in the FZI Living Lab environment that supports testing novel technologies with a higher users' involvement. We have performed a case study related to the office occupancy control, that limits the operation of the lighting system based on the actual use of the space. Preliminary evaluation tests have shown very promising results regarding the usability and the efficiency of the approach: the approach is able to abstract from particular patterns to be recognized into general/declarative situations to be reasoned about.

Future work will be related to modeling a more comprehensive set of patterns for representing more complex situations as described in the Evaluation section. Additionally, new tests in the Living Lab have been planned.

Acknowledgments

Research for this paper was partially financed by EU in the following FP7 projects: ALERT (ICT-258098), PLAY (ICT-258659) and ARtSENSE (ICT-270318).

References

1. Thirunarayan, K., Henson, C., Sheth, A.: Situation Awareness via Abductive Reasoning for Semantic Sensor Data: A Preliminary Report. In: Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009), Baltimore, MD, May 18-22 (2009)
2. Anicic, D., Fodor, P., Rudolph, S., Stuehmer, R., Stojanovic, N., Studer, R.: A rule-based language for complex event processing and reasoning. In: Proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR 2010), pp. 42–57 (2010)
3. Deepak, M.: SNOOP: An Event Specification Language For Active Database Systems. Master Thesis, University of Florida (1991)
4. Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.: Monitoring streams: a new class of data management applications. In: VLDB 2002: Proceedings of the 28th international conference on Very Large Data Bases. VLDB Endowment (2002)
5. Ray, O.: Nonmonotonic abductive inductive learning. Journal of Applied Logic (2008)

6. Gutierrez, C., Hurtado, C.A., Vaisman, A.A.: Introducing time into rdf. IEEE Transactions on Knowledge and Data Engineering 19(2), 207–218 (2007)
7. Ryvkina, E., Maskey, A.S., Cherniack, M., Zdonik, S.: Revision processing in a stream processing engine: A high-level design. In: Proc. Int. Conf. on Data Eng (ICDE), Atlanta, GA, USA (2006)
8. Agrawal, J., Diao, Y., Gyllstrom, D., Immerman, N.: Efficient pattern matching over event streams. In: Proceedings of the 28th ACM SIGMOD Conference, pp. 147–160 (2008)
9. Barga, R.S., Goldstein, J., Ali, M.H., Hong, M.: Consistent streaming through time: A vision for event stream processing. In: Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR 2007), pp. 363–374 (2007)
10. Arasu, A., Babu, S., Widom, J.: The cql continuous query language: semantic foundations and query execution. VLDB Journal 15(2), 121–142 (2006)
11. Anicic, D., Stojanovic, N.: Expressive logical framework for reasoning about complex events and situations. In: Intelligent Event Processing - AAAI Spring Symposium 2009. Stanford University, California (2009)
12. Recommendation from the French Construction code for construction and housing, http://www.legifrance.gouv.fr/affichCodeArticle.do;jsessionid=87AE72FAE86DC9CF56B8673C1B88F9AD.tpdjo08v_2?cidTexte=LEGITEXT000006074096&idArticle=LEGIARTI000006896264&dateTexte=20090619&categorieLien=id
13. Erickson, V.L., et al.: Energy Efficient Building Environment Control Strategies Using Real-time Occupancy Measurements. In: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pp. 19–24 (2009)
14. von Neida, B., et al.: An analysis of the energy and cost savings potential of occupancy sensors for commercial lighting systems, http://www.lrc.rpi.edu/resources/pdf/dorene1.pdf
15. Walawalkar, R., et al.: Effect of Efficient Lighting on Ergonomic Aspects in Modern IT Offices, http://www.walawalkar.com/info/Publications/Papers/EE&Ergonomics.pdf
16. Agarwal, Y., et al.: Occupancy-driven energy management for smart building automation. In: Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building (2010)

Ontology-Driven Complex Event Processing in Heterogeneous Sensor Networks

Kerry Taylor[1] and Lucas Leidinger[2]

[1] CSIRO ICT Centre
GPO Box 664, Canberra, ACT, 2602, Australia
`kerry.taylor@csiro.au`
[2] University of Applied Sciences of the Saarland
`lucas.leidinger@gmx.de`

Abstract. Modern scientific applications of sensor networks are driving the development of technologies to make heterogeneous sensor networks easier to deploy, program and use in multiple application contexts. One key requirement, addressed by this work, is the need for methods to detect events in real time that arise from complex correlations of measurements made by independent sensing devices. Because the mapping of such complex events to direct sensor measurements may be poorly understood, such methods must support experimental and frequent specification of the events of interest. This means that the event specification method must be embedded in the problem domain of the end-user, must support the user to discover observable properties of interest, and must provide automatic and efficient enaction of the specification.

This paper proposes the use of ontologies to specify and recognise complex events that arise as selections and correlations (including temporal correlations) of structured digital messages, typically streamed from multiple sensor networks. Ontologies are used as a basis for the definition of contextualised complex events of interest which are translated to selections and temporal combinations of streamed messages. Supported by description logic reasoning, the event descriptions are translated to the native language of a commercial Complex Event Processor (CEP), and executed under the control of the CEP.

The software is currently deployed for micro-climate monitoring of experimental food crop plants, where precise knowledge and control of growing conditions is needed to map phenotypical traits to the plant genome.

1 Introduction

Sensor networks, especially low-cost wireless sensor networks (WSNs), are rapidly gaining popularity for use in developing scientific knowledge. Scientists are performing dense monitoring of natural environment parameters to learn about matters including faunal distribution and behaviour, biodiversity and biological interactions, air and water quality, micro-climatic conditions, and human impact. In some cases a regular collection of homogenous sensor data is sufficient to support subsequent intensive data analysis over a data archive. In other cases, there is a need for real-time data collection, analysis, and active response. This will arise when scientists are unsure about the how to detect the phenomena being investigated, when a human response is necessary to an observed event, or when the recognised occurrence of an event should cause a change in the monitoring behavior or equipment configuration. For example, when turtle eggs begin to hatch on the beach in stormy weather, a scientist should be alerted and cameras should be activated. When an endangered plant begins to flower and pollinators are detected in the locality, a protecting cloche should be removed to enable pollination. When nitrate concentrations at several points in the water course exceed a threshold, the downstream oyster farmers should be warned and additional water quality parameters should be monitored.

In these cases the events may be detected only by recognising a complex correlation of observations made over multiple sensors attached to multiple WSN nodes, and possibly distributed over multiple networks. The sensors may be heterogenous, as may be the sensing nodes that control them. Although in principle is possible to use in-network processing techniques by explicitly programming each device to make observations and to coordinate and correlate those observations with neighbouring nodes, this is very difficult in general and certainly requires advanced programming skills and dedication with current technology. If we add the typical experimental challenge to the mix—that it is not well known *how* to define the desired event in terms of measurable properties, nor even *what* all the interesting events might be over the life of a sensor network deployment, then we can conclude that better tools are needed to offer this capability.

A better tool should enable an experimental scientist to

– discover sensors that could be used to make relevant measurements;
– develop a specification for the event of interest in terms of the available sensing capability;
– reuse measurements that are already being made if possible or to
– program the sensor devices to make the necessary measurements otherwise;
– describe an action to be taken if the interesting event is detected; and to
– easily deploy the specification for efficient runtime processing.

In this paper we propose the use of ontologies as an important part of such a tool. In earlier work we have shown how to effectively program sensor networks and devices by modelling the sensing capability and programming language in an OWL ontology, using the ontology to add contextual concepts for use in the programming task, and using the ontology reasoning capability to validate commands [12]. In this work, we propose that an ontology can capture valuable domain context for sensor capability description and discovery, for description of an interesting event in terms of potential sensor measurements, and for optimising the execution strategy for run-time event detection.

We do not address sensor discovery, but discovery methods based on ontological descriptions abound, for example [11] and [6]. In our work, we rely on the newly developed SSN ontology from the W3C Semantic Sensor Network Incubator Group [13], to take advantage of an emerging community terminology.

For enactment of event detection, we employ one of the many commercial or open source *complex event processors* (CEP) or distributed stream management systems (DSMS). These software systems are capable of binding to input *streams* of real-time structured messages, such as those arising as observation values reported from sensor networks. They can efficiently evaluate queries over those streams that check message content for particular values and for values related to previous messages in the same stream or to messages in other streams. They support temporal and aggregation operators to enable the detection of events that arise as complex correlations of data values in the messages. These systems, too, are notoriously difficult to configure [5] although they commonly offer an SQL-like query language with which to express the desired events. There is no standard language (although they are usually similar to CQL[2]) nor even a widely-used algebra or well-defined semantic model for events.

In our work, we develop an ontology of domain, event, observation and sensor network, and use that ontology to drive a user interface. When the user specifies an event of interest through the interface, our middleware processes the specification and generates configuration commands for a CEP system. The CEP system monitors the selected data streams and generates alerts to be delivered to specified clients when the event occurs. Our contribution is the general method for a *better tool* for scientists as described above, although this paper focuses on the parts that offer development and optimised deployment of the specification of a complex event.

Outline. We begin our description of the method for ontology-driven complex event detection by presenting the overall architecture and the role of the ontology and CEP in section 2. In section 3 we describe the ontology driven user interface, and how it is used to specify complex events over sensor measurements. In section 4 we describe how our semantic middleware processes the specification together with the initial ontology to generate optimised configuration instructions for the CEP middleware. In order to apply the system each sensor network needs an interface developed as described in section 5. After an analysis of related work in section 6 we conclude with a summary and presentation of future work.

2 Architecture - Event Framework

Our *event framework* is software that implements our method for ontology driven complex event processing in heterogeneous sensor networks.

The event framework includes all software to define, process and recognize complex events in sensor networks. The framework is composed of five separate parts, each of which may be replaceable for different applications, although the semantic event middleware is the core component:

User interface to build complex event definitions;
Semantic event middleware to process complex event definitions;
Management module interfaces to program and access different kinds of sensor data sources;

Event ontology containing the model of sensor networks, sensor programs, complex events and the event environment; and

CEP server to perform the complex event detection over multiple heterogeneous sensor network data streams (*Coral8*[1] in our case).

Fig. 1 shows the event framework and illustrates the communication between components. The basic process can be described as follows:

1. The user defines a complex event definition composed from several atomic sensed events within the user interface.
2. The complex event definition is processed and stored in the event ontology.
3. The ontology data is used to program the required sensors.
4. The event detection part for the current event definition is separately saved in an *ExportOntology* and sent to the semantic event middleware.
5. The semantic event middleware transforms the received ontology into CEP streams and queries, written in the CEP-dependent Event Processing Language (EPL).
6. The semantic event middleware sets up the CEP server with created streams and queries, and initiates the event detection process.
7. The CEP server performs the event detection and sends an alert if the specified event has been recognized.

The *EventOntology* is the central part of the Event Framework. It allows the use of reasoning over event information to obtain additional knowledge and to perform semantic optimisations. A clear formalization of the event and measurement environment is necessary to exploit these advantages. Our OWL 2.0 event ontology extends an early form of the SSN ontology of the W3C SSN-XG [13]. It is reported to be in ALCIQ(D) by the ontology editor, Protégé. Along with the sensors, it models the domain of application and the concept and model of the entire event framework. Events, alerts, triggers, streams, locations, instruments, phenomena, sensors and sensor programs are all part of this description. Additional classes and instance data are included to describe relations between single events, time intervals, and to define different kinds of sensor programs. The high-level structure is apparent through its reflection in the user interface given in section 3. The interested reader may find the ontology at `http://research.ict.csiro.au/conferences/ssn/EventOntology_no_imports.owl`.

The CEP server application performs the actual complex event detection. For this, the server receives a stream with configuration information for the event data sources and a query for the complex event detection, both expressed in the CEP's proprietary EPL. The CEP server is also responsible for sending the user defined alert message if an event has been detected.

We now provide more detail on our user interface, our semantic event middleware and our management module interfaces for each source.

[1] The Coral8 CEP-platform. `http://www.aleri.com/products/aleri-cep/coral8-engine`

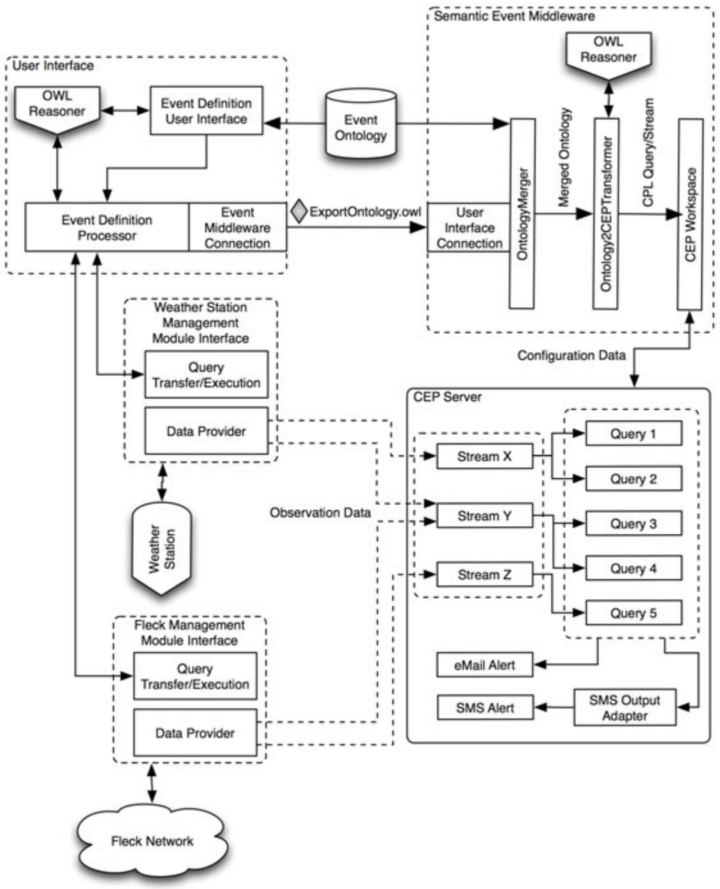
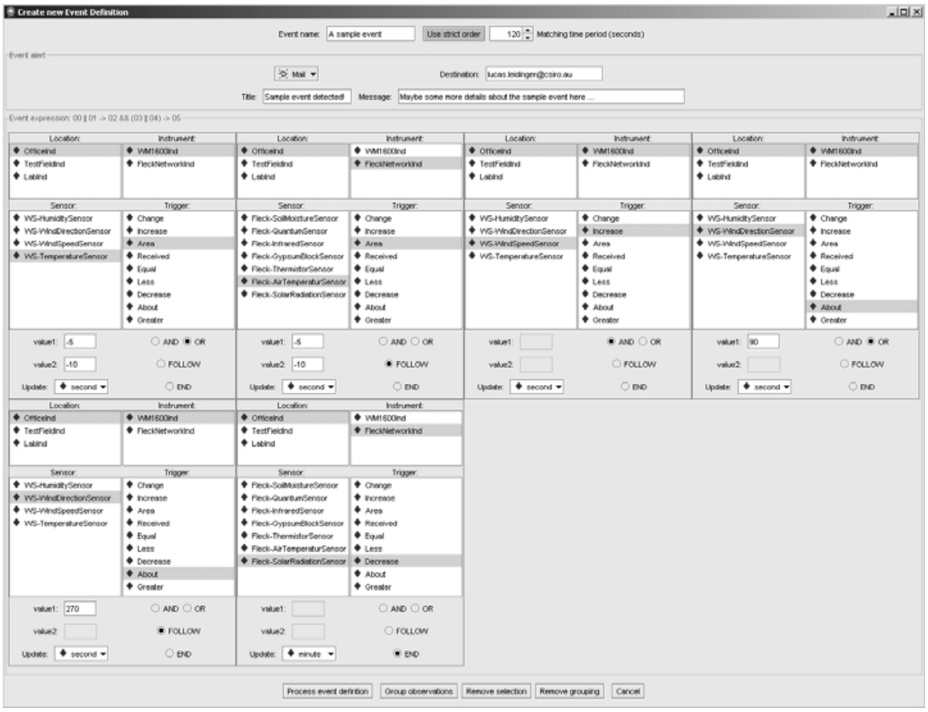


Fig. 1. Event Framework

3 Ontology-Driven User Interface

The user interface is implemented as a plug-in to the popular OWL 2.0 ontology editor *Protégé*. It contains windows to manage new locations and instruments, and to create complex event definitions as shown in see Fig 2. It has the functionality to store entered information as ontology instance data, the capability to perform semantic optimisations, and to program the sensor devices before complex event descriptions are sent for the further processing to the semantic event middleware. The sensor programming capability [12] is not further described.

The ability to provide a user interface which allows one to define events in a logical and expressive way and also to store this definition in an ontology requires splitting the entire complex event definition into parts: *Events*, *Alerts*, *Observations*, and *Triggers*. Each part describes exactly one event property that is able to be stored within an ontology.



Event name: A sample event
120 Matching time period (seconds)
Mail
Destination: lucas.leidinger@csiro.au
Title: Sample event detected!
Message: Maybe some more details about the sample event here ...
Event expression: 00 ‖ 01 -> 02 && (03 ‖ 04) -> 05

Location:	Instrument:	Sensor:	Trigger:	value1:	value2:	Update:	
OfficeInd	WM1600Ind	WS-TemperatureSensor	Area	-5	-10	second	AND [✓ OR] FOLLOW END
OfficeInd	FleckNetworkInd	Fleck-AirTemperaturSensor	Area	-5	-10	second	AND OR [✓ FOLLOW] END
OfficeInd	WM1600Ind	WS-WindSpeedSensor	Increase			second	[✓ AND] OR FOLLOW END
OfficeInd	WM1600Ind	WS-WindDirectionSensor	About	90		second	AND [✓ OR] FOLLOW END
OfficeInd	WM1600Ind	WS-WindDirectionSensor	About	270		second	AND OR [✓ FOLLOW] END
OfficeInd	FleckNetworkInd	Fleck-SolarRadiationSensor	Decrease			minute	AND OR FOLLOW [✓ END]

Location: OfficeInd, TestFieldInd, LabInd
Instrument: WM1600Ind, FleckNetworkInd
Sensor: WS-HumiditySensor, WS-WindDirectionSensor, WS-WindSpeedSensor, WS-TemperatureSensor
Sensor: Fleck-SoilMoistureSensor, Fleck-QuantumSensor, Fleck-InfraredSensor, Fleck-GypsumBlockSensor, Fleck-ThermistorSensor, Fleck-AirTemperaturSensor, Fleck-SolarRadiationSensor
Trigger: Change, Increase, Area, Received, Equal, Less, Decrease, About, Greater

Fig. 2. User Interface

3.1 Events

Every event description consists of two major parts: an alert which will be activated if an event has been recognized, and the definition of the event itself. The definition can be expressive and must be able to represent user-defined complex structures. To achieve this goal, every complex event definition, called an *observation*, is composed of several atomic observations. Each atomic observation can be individually configured in four main steps.

1. Select a sensor from an instrument at a specific location.
2. Choose the update interval for this sensor.
3. Set up a trigger including trigger values.
4. Define the relationship to the following observation.

Defining the relationship to the following observation together with functionality to group sequential observations enables the formation of complex event descriptions. The design of instruments and locations declares that every instrument is located at a specific location. This allows the grouping of all instruments from one location and the selection of instruments based on the location name. The location itself has no finer definition in this implementation, although it could be useful to add extra information like physical values or the kind of location and an environment description. Both location and instrument descriptions are stored within the ontology. The user interface loads this information dynamically and only displays valid entries within the event definition window.

3.2 Alerts

The user can choose to receive an email or a text message on a mobile phone if an event has been detected. Every email includes standard passages about the event processing supplemented with event specific information such as the latest sensor data and a description of the trigger configuration. The SMS alert is shorter. Alerts are defined within the ontology, together with the relevant EPL code fragments, and dynamically loaded and integrated into the user interface. This easily allows the integration of additional alert types. In future work we will add an alert type to send a control message to external software systems.

3.3 Observations

Complex events are designed within the ontology, so that every complex event contains an observation. An observation itself is used as a generic term for five different kinds that are used to realize the complexity of definitions. Observation operators are used to define the logical *AND*, *OR* and *FOLLOWED BY* relationships between atomic or compound observations. Bracketed grouping can also be represented in observation groups.

Atomic Observation is the description of an atomic event within the entire complex event definition. It contains the information to program the selected sensor and the trigger definition for the event.

Observation Intersection is used to create a logical *AND* (&&) relationship between two observations. Each of these observations can be a single observation, an observation union, an observation sequence or another observation intersection in turn.

Observation Union is the counterpart to the observation intersection. It links two observations by a logical *OR* (||) relationship. Here again, each of observations can be a single observation, an observation intersection, an observation sequence or another observation union.

Observation Sequence is used to create a *FOLLOWED BY* (,) relationship between two observations. *FOLLOWED BY* specifies that the next event has to be recognized chronologically after the previous one. Each of these observations can be a single observation, an observation intersection, an observation union or an observation sequence. This is only available if strict order is used, as described in the next paragraph.

Observation Groups are used to combine multiple single observations. Each group belongs to a certain event and contains all consecutive observations summarized by the user within one parenthetical group.

It is not possible to define *AND* and *OR* relations using the corresponding primitives of the underlying ontology language itself because we also need to use the custom *FOLLOWED BY* operator to express temporal relationships between atomic observations.

The *FOLLOWED BY* relationship is used to define temporal sequences of atomic events meaning that the next event within the event expression must arrive after the previous one. A time period specifies the duration within which individual observations can be triggered in order to be valid and taken into account for the entire complex event detection. Using a strict temporal order allows the defintion of event definitions such as: Send an alert if the temperature at location A or at location B falls below 0 degrees and then humidity becomes less than 15 percent at location A followed by wind speed with more than 130 km/h in combination with a north-easterly wind direction at location B within a period of 30 seconds.

An important part of the observation design is that the ontology includes code fragments to generate CEP-platform-specific statements which are used for the run-time event detection.

3.4 Triggers

To create expressive and practical event descriptions, it is necessary to interpret, analyse and filter environmental observations to be able to define which occurrences are interesting for an event. The data source is already described by the sensor program within an atomic observation. In order to be in a position to recognize complex events, it is not enough to simply compare incoming values. It is much more revealing to observe time dependent behaviour and value patterns. For this reason, nine different kinds of *Triggers* were designed.

About examines if the received data is equal to a user defined value. Values within 10 percent tolerance are detected.
Area recognizes all readings in the interval between two given values.
Change monitors if the current reading changes with respect to the average value of the previous two readings.
Decrease is used to detect if the latest value is lower than the average of two preceding readings.
Equal simply checks if the current value equals a value defined by the user.
Greater triggers if the received value is greater than a user defined value.
Increase is the opposite of Decrease and observes if the latest value is greater than the average of two preceding readings.
Less triggers if the received reading is smaller than a user defined value.
Received recognizes if data from a sensor has been received, without further qualification.

Like the observation concept, the trigger specification within the ontology contains CEP-platform specific code fragments which are used to generate queries for the event detection. The trigger implementation focuses on demonstration purposes and is quite elementary. The calculations of limiting values as well as the limiting values itself were chosen arbitrarily to obtain a satisfactory behaviour in the targeted environment. This design is sufficient to demonstrate the technical feasibility of defining complex triggers in an ontology and to transform them into CEP statements.

4 Semantic Event Middleware

All complex event descriptions in the form of an ontology are forwarded from the user interface to the semantic event middleware. The semantic event middleware uses the complex event descriptions to generate streams and queries for the CEP server application to enact the event detection. Subsequently, the CEP server is connected, streams and queries are registered, and the event detection process started.

4.1 Communication with the User Interface

Within the semantic event middleware, the same ontology as in the user interface*EventOntology*, is used. New complex event descriptions generated in the user interface are received by a network connection, so that the event definition and the event transformation can be performed independently. Only ontology fragments for new complex event descriptions are exchanged; they are merged with the local *EventOntology* on receipt.

4.2 Transforming Ontology Data into EPL Statements

Through the decomposition of the complex event definitions into individual parts, describing exactly one property, it is possible to extract the entire event description from the ontology and make it usable to generate CEP program code. These CEP programs are composed of streams and queries. *Streams* are used to create connections to event data sources and to filter the required data for the event detection. *Queries* describe the actual event detection including all observed sensors, triggers and relationships between atomic observations, groups and the temporal order of observations.

At this point, code fragments, which are stored as datatype property values in the event ontology are used to form EPL statements. Furthermore, class and instance names from the ontology are used to generate names for variables in the CEP code. Together, this supports the generation of large parts of the CEP code automatically and independently of the particular EPL. Storing code fragments not only for one CEP implementation but for multiple CEP-platforms in the ontology would additionally allow the choice of different CEP implementations. However, as the grammar of the EPL needs to be reflected in the translation of the *Ontology2CEPTransformer* (fig. 1), alternative implementations of this module may be necessary for CEP platforms employing a very different language structure.

Fig. 3 illustrates this usage of ontology data to create Coral8 EPL program code. The shown ontology data and the code snippet correspond to a complex event definition example which recognizes temperature between -20 and -10 degrees, amongst other observations. All shaded statements are generated from the description of sensor programs from the ontology. For example, the sensor program individual "program_0_1283418466826Ind" has the object properties "WS-TemperatureSensor" and "WM1600". Both are used as variable names to describe input and output streams as well as CEP windows that include the definition of which data has to be filtered. The black highlighted expression within the "where" clause is also created automatically. The template clause, "trigCmd", for the *Area* trigger, "column > value1 AND column < value2", is used as basis for the "where" clause. This information is stored in the ontology as a part of the definition of the *Area* trigger class, and so, through a reasoner, is also a dataproperty value of the "trigger_0_1283418466826Ind" individual of that class. Other data property values instantiated through the user interface provide the user defined thresholds "hasValue1" and "hasValue2" to replace the terms "value1" and "value2" inside the template clause. The template string "column" is replaced by the corresponding variable name generated by using the description of sensor program "program_0_1283418466826Ind".

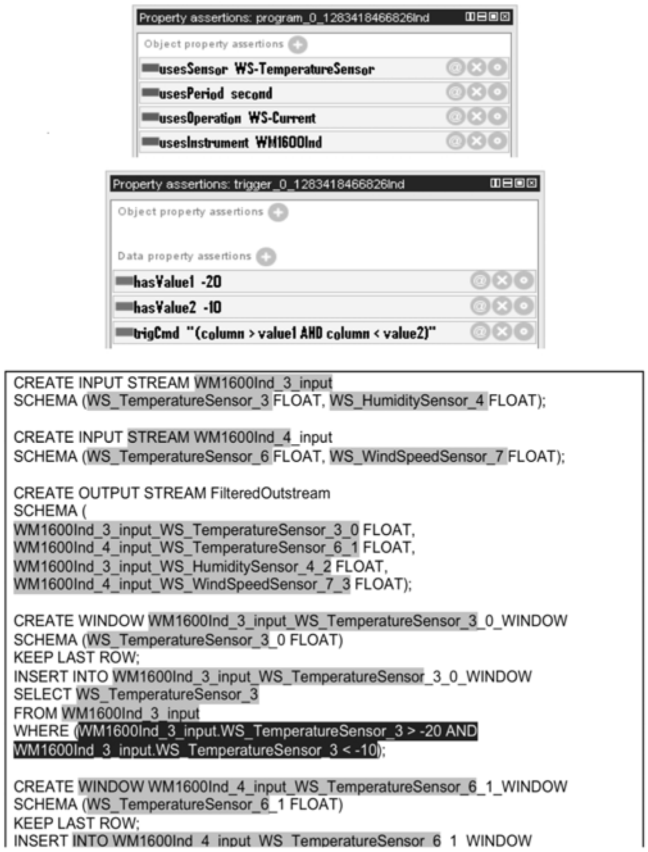
4.3 Semantic Optimisation of Streams

Reasoning is used to improve the efficiency of streams within the semantic event middleware. The goal is to reuse existing information to save run-time resources and to reduce the amount of data which must be transferred between the data sources and the event processing application. This may be especially important when multiple users are being served.

Since CEP platforms and the *Semantic Event Middleware* are able to monitor several complex events at the same time, it is possible to re-use some streams already configured in the CEP for previously specified complex events. When a stream definition is created through this software through commands to the CEP, it is also defined in the *EventOntology* as an instance of the stream class with associated property instances. For example, if streams which provide temperature and wind speed from location *plot21north* and wind direction from location *plot23central* have previously been established, then a new query which only requires wind speed from *plot21north* and wind direction from location *plot23central* can reuse the two existing streams. The usable pre-existing streams are retrievable by a reasoner, being the individual members of the class described as a stream which measures wind speed at *plot21north* and the individual members of the class described as a stream which measures wind direction from location *plot23central*.

5 Management Module Interfaces

In order that different sources can be used for the event processing, the module for specifying and providing event data streams is abstracted. The usage of a



Property assertions: program_0_1283418466826Ind

Object property assertions

usesSensor WS-TemperatureSensor

usesPeriod second

usesOperation WS-Current

usesInstrument WM1600Ind

Property assertions: trigger_0_1283418466826Ind

Object property assertions

Data property assertions

hasValue1 -20

hasValue2 -10

trigCmd "(column > value1 AND column < value2)"

```
CREATE INPUT STREAM WM1600Ind_3_input
SCHEMA (WS_TemperatureSensor_3 FLOAT, WS_HumiditySensor_4 FLOAT);

CREATE INPUT STREAM WM1600Ind_4_input
SCHEMA (WS_TemperatureSensor_6 FLOAT, WS_WindSpeedSensor_7 FLOAT);

CREATE OUTPUT STREAM FilteredOutstream
SCHEMA (
WM1600Ind_3_input_WS_TemperatureSensor_3_0 FLOAT,
WM1600Ind_4_input_WS_TemperatureSensor_6_1 FLOAT,
WM1600Ind_3_input_WS_HumiditySensor_4_2 FLOAT,
WM1600Ind_4_input_WS_WindSpeedSensor_7_3 FLOAT);

CREATE WINDOW WM1600Ind_3_input_WS_TemperatureSensor_3_0_WINDOW
SCHEMA (WS_TemperatureSensor_3_0 FLOAT)
KEEP LAST ROW;
INSERT INTO WM1600Ind_3_input_WS_TemperatureSensor_3_0_WINDOW
SELECT WS_TemperatureSensor_3
FROM WM1600Ind_3_input
WHERE (WM1600Ind_3_input.WS_TemperatureSensor_3 > -20 AND
WM1600Ind_3_input.WS_TemperatureSensor_3 < -10);

CREATE WINDOW WM1600Ind_4_input_WS_TemperatureSensor_6_1_WINDOW
SCHEMA (WS_TemperatureSensor_6_1 FLOAT)
KEEP LAST ROW;
INSERT INTO WM1600Ind_4_input_WS_TemperatureSensor_6_1_WINDOW
```

Fig. 3. Using ontology information to generate Coral8 CEP statements

management module interface allows the implementation of an independent solution for each specific kind of instrument, sensor network or event source. for our test deployment have developed two instances of the management module interface, one to connect a *WeatherMaster1600*[2] weather station instrument and the other to interface with a relational database archive. Each additional management module interface would be accessed, integrated and implemented in the same manner. It would be necessary to update the user interface source code to use a new management module interface as well as to extend the model of the *EventOntology* with new instrument specific capabilities.

Most parts of the weather station management module interface are reused from the related project [12]. The software allows high-level programming for the weather station instrument and access to measured data.

[2] Environdata WeatherMaster1600.
`http://www.environdata.com.au/weathermaster1600`

6 Related Work

Research over the past twenty years has produced a large number of research prototypes and commercial products to continuously monitor data streams for detection of events arising as complex correlations of source data values. These may be known as distributed stream management systems, complex event processors, or sensor data management systems. Some are directed at scientific users and incorporate grid computing elements; others are optimised for high message throughput rates; and others again are closely integrated with the sensor networks that generate the data and can instigate collection of data from the sensors on demand. Some of the best known are Aurora/Borealis[1], TinyDB [10], Caldera[9], and Esper [4].

In concurrent work, [3] extends the standard RDF query language SPARQL to provide a SPARQL interface to querying streaming data sources, together with a temporally extended RDF response. One aim of that work was to make streaming sensor data available within the linked open data framework, hence the choice of an RDF/SPARQL model. However, like in our work, the queries are mapped to to a native stream processing language for run-time execution. In that case a more traditional query translation approach is used with an intermediate algebraic representation, and reverse translation of query answers.

Our event framework does not offer a query language interface directly but we offer an exploratory GUI that can be understood as defining an interesting event rather than querying for instances of the event. However, because an event specification in our approach is simply an ontology fragment, other query-language like interfaces or API could be readily designed. A query language based on description logic conjunctive queries over the classes and properties of an event definition would be a more natural match for our work than extended SPARQL.

Although we do not offer a linked open data solution, we rely on the more expressive OWL ontology language to provide design-time contextualisation of the sensor data and optimisation, but with no run-time processing overhead. Our approach provides a more direct translation path to the underlying EPL, so is likely to allow more expressive queries (where modelled in the ontology) and possibly also more compact and efficient EPL code. For example, our system straightforwardly offers complex events defined by integrating multiple sensor data streams, whereas that capability is set down for future work in [3].

In [6], a SensorMashup platform is described that offers a visual composer for sensor data streams. Data sources and intermediate analytical tools are described by reference to an ontology, enabling an integrated discovery mechanism for such sources. A SPARQL endpoint is offered to query both sensor descriptions (in the conventional manner) and also individual sensor data streams. Like our event framework, a third-party DSMS is used to manage the raw sensor data streams. A SPARQL query is translated to an SQL-like continuous query over the streams to be handled by the DSMS, but the usual essential windowing and aggregation functions of a DSMS (such as "average") cannot be used as there is no SPARQL counterpart. The most important difference to our framework is in the approach to queries requiring correlations over multiple independent streams. In SensorMashup such queries are written, mapped and processed separately for each input stream. Although the visual composer supports the explicit combination of streams, it appears that this means feeding the streams as inputs to subsequent processing steps embedded in arbitrary software services, so correlation over the streams to recognise combined events must be done outside the DSMS. In our approach a single declarative query over multiple streams is mapped directly to a declarative query in the language of the CEP, therefore enabling the CEP to perform run-time optimisation of the processing for which it is designed. On the other hand, other than the basic operators embedded in our CEP, our current approach does not support integration of analytical tools.

7 Future Work and Conclusion

We have shown that it is possible to offer a sophisticated, domain-contextualised service for complex event processing. The ontology can be used to specify events that comprise complex correlations of observations over multiple sensor data streams, and to specify an action to be taken when events occur. The ontology is used only for event definition and optimisation and is compiled out of the specification for run time processing, where a third party high-throughput complex event processing engine is used. This approach offers the advantages of semantic descriptions but without the cost of semantic interpretation at run time, permitting choice of the best CEP or DSMS tool for stream processing.

Most such processors can be used, provided they offer a query language interface. Because the ontology itself holds command language fragments, and most of the command languages have a similar syntax, the adaptation to a different event processor may require no more than replacement of the fragments in the ontology. Some alteration to the code managing the CEP interface might also be required. For broader sensor network applications, it will be necessary to extend the complexity of the aggregation and relationship operators over stream values beyond those that are available natively in the CEPs. We plan to incorporate a statistical analysis component, and possibly also a textual analysis component to support this. We expect this will slow the run-time processing but it will offer added functionality in relatively low throughput applications.

We have described our work in terms of the Protégé plug-in user interface to construct event specifications. Although our user interface is ontology-driven, we expect that the capability it offers would, in some cases, be better built into other tools more specifically customised to the domain of application. Recalling that the task of the user interface is only to supply an ontology fragment that drives the downstream processing, this should not be difficult.

We have integrated the work reported in this paper with our earlier work for ontology-driven sensor network programming[12]. Within one ontology-driven interface a user can both program the sensor networks and specify actions to be taken when complex events arising from the sensed data are detected. Furthermore, a user can specify a complex event of interest, within the capability of the available sensor networks, and if the necessary phenomena are not currently being monitored, the system will automatically and transparently program those various sensor networks to monitor the necessary phenomena.

Our work is currently deployed on our Phenonet network for agricultural monitoring installed near Leeton, NSW, Australia. Although a fairly small deployment, the architecture is highly heterogeneous. The network includes several Fleck devices[3] of a WSN with sensors for soil moisture, leaf temperature, and soil moisture. There is a separate Fleck WSN with the node directly connected to a Vaisala Automatic Weather Station [4], a solar radiation sensor, and a photosynthetically active radiation sensor. Another independent wireless network of Hussat data loggers[5] with soil temperature profile and soil moisture profile sensors is also present. Finally there is an independent IP-connected Environdata automatic weather station. Currently, stream data arising from Fleck and Hussat nodes is retrieved by polling a database archive and automated programming of the nodes through the event framework is not possible. The programming capability and live stream feed for these sources will be available shortly, taking advantage of code optimisation work [7] in a service architecture [8].

One event that is particularly important in this domain is the occurrence of frost. We will investigate including frequent weather reports as a source of streaming data together with our sensors in the field. We will use the system to develop an adequate recognition of a frost occurrence and connect the event notification to the control system for infrastructure that can protect the experimental crop in the field.

The strength and novelty of this work lies in its use of ontologies and reasoning. We have shown how a scientist can develop a specification for an event of interest in terms of the available sensing capability, reusing measurements that are already being made. We have shown how a scientist can describe an action to be taken if the interesting event is detected, and can easily deploy the specification for efficient runtime processing. Because of the ontological component, the work can also be used together with semantic discovery techniques and also semantic sensor network programming techniques to offer a complete solution for user-driven scientific and experimental reuse of heterogenous sensor networks.

Acknowledgement. The authors thank the members of the CSIRO semantic sensor networks team, Peter Lamb, Doug Palmer, Laurent Lefort, Leakha Henry and Michael Compton, and the CSIRO Phenomics scientists, Xavier Sirault and

[3] Powercom Fleck. See `http://www.powercomgroup.com/Latest_News_Stories/Fleck_long_range_wireless_sensing_and_control.shtml`

[4] Vaisala WM30. See `http://www.vaisala.com/files/WM30_Brochure_in_English.pdf` for the data sheet

[5] Hussat wireless microloggers. See `http://hussat.com.au/`

David Deery. This work was conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine.

References

1. Abadi, D.J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.B.: Aurora: a new model and architecture for data stream management. VLDB Journal 12(2), 120–139 (2003)
2. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: Semantic foundations and query execution. Very Large Database (VLDB) Journal 14 (2005)
3. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
4. Esper–Complex Event Processing. Espertech event stream intelligence (December 2010), http://esper.codehaus.org/
5. Hinze, A., Sachs, K., Buchmann, A.: Event-based applications and enabling technologies. In: DEBS 2009: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, pp. 1–15. ACM, New York (2009)
6. Le-Phuoc, D., Hauswirth, M.: Linked open data in sensor data mashups. In: Taylor, K., Ayyagari, A., De Roure, D. (eds.) Proceedings of the 2nd International Workshop on Semantic Sensor Networks, SSN 2009, Washington DC, USA, October 2009, vol. 522, pp. 1–16 (2009) CEUR workshop proceedings
7. Li, L., Taylor, k.: Generating an efficient sensor network program by partial deduction. In: Zhang, B.-T., Orgun, M.A. (eds.) PRICAI 2010. LNCS, vol. 6230, pp. 134–145. Springer, Heidelberg (2010)
8. Li, L., Taylor, K.: A framework for semantic sensor network services. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 347–361. Springer, Heidelberg (2008)
9. Liu, Y., Vijayakumar, N., Plale, B.: Stream processing in data-driven computational science. In: Proc. 7th IEEE/ACM International Conference on Grid Computing, GRID 2006, pp. 160–167. IEEE, Washington, DC, USA (2006)
10. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TinyDB: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. 30, 122–173 (2005)
11. Sirin, E., Parsia, B., Hendler, J.: Filtering and selecting semantic web services with interactive composition techniques. IEEE Intelligent Systems 19, 42–49 (2004)
12. Taylor, K., Penkala, P.: Using explicit semantic representations for user programming of sensor devices. In: Advances in Ontologies: Proceedings of the Australasian Ontology Workshop Conferences in Research and Practice in Information Technology, Melbourne, Australia, December 1. CRPIT, vol. 112, Australasian Computer Society (2009)
13. W3C SSN-XG members. SSN: Semantic sensor network ontology (December 2010), http://purl.oclc.org/NET/ssnx/ssn

A Semantically Enabled Service Architecture for Mashups over Streaming and Stored Data

Alasdair J.G. Gray[1], Raúl García-Castro[2], Kostis Kyzirakos[3], Manos Karpathiotakis[3], Jean-Paul Calbimonte[2], Kevin Page[4], Jason Sadler[4], Alex Frazer[4], Ixent Galpin[1], Alvaro A.A. Fernandes[1], Norman W. Paton[1], Oscar Corcho[2], Manolis Koubarakis[3], David De Roure[4], Kirk Martinez[4], and Asunción Gómez-Pérez[2]

[1] University of Manchester, United Kingdom
[2] Universidad Politécnica de Madrid, Spain
[3] National and Kapodistrian University of Athens, Greece
[4] University of Southampton, United Kingdom
http://www.semsorgrid4env.eu

Abstract. Sensing devices are increasingly being deployed to monitor the physical world around us. One class of application for which sensor data is pertinent is environmental decision support systems, e.g. flood emergency response. However, in order to interpret the readings from the sensors, the data needs to be put in context through correlation with other sensor readings, sensor data histories, and stored data, as well as juxtaposing with maps and forecast models. In this paper we use a flood emergency response planning application to identify requirements for a semantic sensor web. We propose a generic service architecture to satisfy the requirements that uses semantic annotations to support well-informed interactions between the services. We present the SemSorGrid4Env realisation of the architecture and illustrate its capabilities in the context of the example application.

Keywords: Sensor web architecture, semantic sensor networks, sensor network ontology, mashups, use case.

1 Introduction

Sensor networks promise to bridge the gap that, for too long, has separated computing applications from the physical world that they model and in which they are ultimately embedded. Many scientific and technological challenges need to be tackled before sensor networks can be exploited to their full capacity for aiding decision support applications. Additionally, as more and more sensor networks are independently developed and deployed, it becomes increasingly important to support their reuse in applications that were not foreseen or that transcend their original purpose. This will facilitate the use of sensor network technology to support decision-making that requires on-the-fly integration of data of differing modalities, e.g. sensed data with data stored in databases, as well as the *ad hoc* generation of mashups over data stemming from computations that combine real-time and legacy historical data. This, in turn, will enable the enacting of decisions based on such real-time sensed data.

One area that has seen a massive increase in the deployment of sensing devices to continuously gather data is environmental monitoring [9]. For example, metocean data, i.e. wave, tide, and meteorology data, for the south coast of England is measured by two independently deployed sensor networks—the Channel Coastal Observatory (CCO)[1] and WaveNet[2]—as well as meteorological data from the MetOffice[3]. More broadly, data from sensors is being used worldwide to improve predictions of, and plan responses to, environmental disasters, e.g. coastal and estuarine flooding, forest fires, and tsunamis. The models that make these predictions can be improved by combining data from a wide variety of sources. For example, in a coastal flooding scenario, details of sea defences, combined with current wave information, can be used to identify potential overtopping events—when waves go over the top of a sea defence. When responding to a flooding event additional data sources such as live traffic feeds[4], and details of public transport infrastructure can help inform decisions.

Enabling the rapid development of flexible and user-centric decision support systems that use data from multiple autonomous independently deployed sensor networks and other applications raises several technical challenges, including: (i) Discovering relevant sources of data based on their content, e.g. features of interest and the region covered by the dataset; (ii) Reconciling heterogeneity in the data sources, e.g. the modality, data model, or interface of the data source, and enabling users to retrieve data using domain concepts; and (iii) Integrating and/or mashing up data from multiple sources to enable more knowledge about a situation to become available. OGC-SWE [3] and GSN [1] are previous proposals that share some of our aims in this paper. However, both require data sources to expose their respective data model, thus limiting the reuse of existing sources from a multitude of domains. Additionally, neither supports integrating data from heterogeneous data sources. Our proposed approach makes extensive use of semantic technologies to reconcile the heterogeneity of data sources whilst offering services for correlating data from independent sources. This enables user-level applications to generate queries over ontologies which are then translated into queries to be executed over the data sources. Data is returned expressed in terms of the user-level ontology and can be correlated and juxtaposed with other data in a meaningful and controlled manner.

The rest of the paper is structured as follows. In Section 2 we provide a detailed description of the flood emergency planning scenario for the south coast of England and identify the set of requirements. Section 3 provides an overview of the ontology network that we have developed to represent the information

[1] `http://www.channelcoast.org/` (21 October 2010).

[2] `http://www.cefas.co.uk/our-science/observing-and-modelling/monitoring-programmes/wavenet.aspx` (21 October 2010).

[3] `http://www.metoffice.gov.uk/` (4 November 2010).

[4] `http://www.highways.gov.uk/rssfeed/rss.xml` (4 November 2010).

needed in this scenario. We then present our architecture for a semantic sensor web in Section 4, and describe the role of ontologies and semantic annotations in the architecture. Section 5 describes a prototype deployment of our architecture for the flood emergency response use case. We discuss related work in Section 6 and present our conclusions in Section 7.

2 Motivating Scenario and Requirements

This section presents a flood emergency planning scenario for the south coast of England, drawn from the SemSorGrid4Env project[5], that illustrates the use of data coming from sensor networks together with traditional stored data sources, e.g. relational databases, and map images. Requirements for a software infrastructure aimed to support the use case are identified. Although the requirements have been identified for a specific use case, they are representative of general requirements for a software infrastructure to enable web applications to generate data mashups over heterogeneous data sources, including sensor data.

2.1 Flood Emergency Planning Scenario

A flood emergency response planner would like to receive an alert, by email or SMS text message, when a flooding event is predicted to occur, or has been detected to have occurred, for their area of responsibility. Such a prediction may stem from a model that forecasts the future tidal patterns based on the current sea state, as measured by sensor networks such as the CCO[1] and WaveNet[2], and predicted weather from a web feed such as that provided by the UK Met Office[3]. Alternatively, it could arise from a sea defence, the details of which are stored in a database such as the UK Environment Agency's National Flood and Coastal Defence Database (NFCDD)[6], being overtopped, as measured by some sensor. This requires characterising the overtopping event that, in turn, requires stored and sensed data to be correlated.

When the emergency response planner receives an alert, they need to plan a suitable response based on the likely severity of the flood, its location, and the likely impact on the public, the environment, and industry. To enable them to respond appropriately, they need to dynamically find relevant data sources based on the thematic and spatiotemporal coverage of the data. Once relevant sources have been identified, suitable mechanisms for retrieving, correlating, and displaying the data in terms of the flooding domain are required that hide the complexities of the heterogeneity of the sources, data modalities, and terminology used. For example, the manager responsible for the Portsmouth region would need details of the shipping due to arrive (available from web feeds), as well as details of the ships and their cargo (available from databases). They would also need the weather forecast (available from web feeds), details of the current sea

[5] http://www.semsorgrid4env.eu/ (24 November 2010).

[6] http://www.scisys.co.uk/casestudies/environment/nfcdd.aspx (21 October 2010).

state (available from sensors), and a forecast of how the latter are likely to evolve in the region (available from predictive models). This would enable them to make more accurate decisions about the likely effects of the sea-state on shipping. Similarly, they must assess the risk to the public. To aid this, they need details of the transportation infrastructure (available from stored data), populated areas (available as maps), and the predicted effects of the flood (available from models based on the sensor data).

2.2 Requirements

The following general requirements can be drawn from the scenario.

R1 *Accurate characterisation of conditions that define an event.* It should be possible to describe the data of interest and allow the system to discover how to retrieve the data. This is shown in the example scenario by the need to automatically identify when the conditions characterising an overtopping event are met so that an alert can be generated.

R2 *Correlation of data of differing modalities.* It should be possible to correlate streaming data from sensors with stored data such as stream archives and databases, and visual data such as maps. This is shown in the scenario by the need to detect overtopping events that combines sensor data with data stored in a database and display it on a map.

R3 *Integrating data coming from heterogeneous data models.* It should be possible to mediate data coming from autonomous sources regardless of the source data model, i.e. the conceptualisation and terminology used in the data, and representations (e.g. relational, RDF, and XML). This is shown in the scenario by the need to use a wide variety of independent sources.

R4 *Discovery of relevant data sources.* It should be possible to identify potentially useful sources of data based on the spatiotemporal and thematic coverage of the data provided. This is shown in the scenario by the need to find sources of data when planning the response to a flooding event.

R5 *Presentation and control of information.* It should be possible for users to discover, relate, juxtapose, and display information from a variety of sources without needing to understand the sources, and in response to an evolving situation. This is shown throughout the scenario.

3 Modelling Semantic Sensor Web Information

The requirements presented in the previous section indicate that there is a clear need to support the *ad hoc* responsive evolving use of an information space. The data resources of the information space will contain various forms of heterogeneity, including data modality (i.e. sensed, stored, and graphical), data model (i.e. terminology), and data representation (e.g. relational, RDF, and XML), which need to be reconciled into a single coherent conceptualisation. We use ontologies to represent the common data model for the information space since they facilitate: (i) Describing the different infrastructure services and data sources as

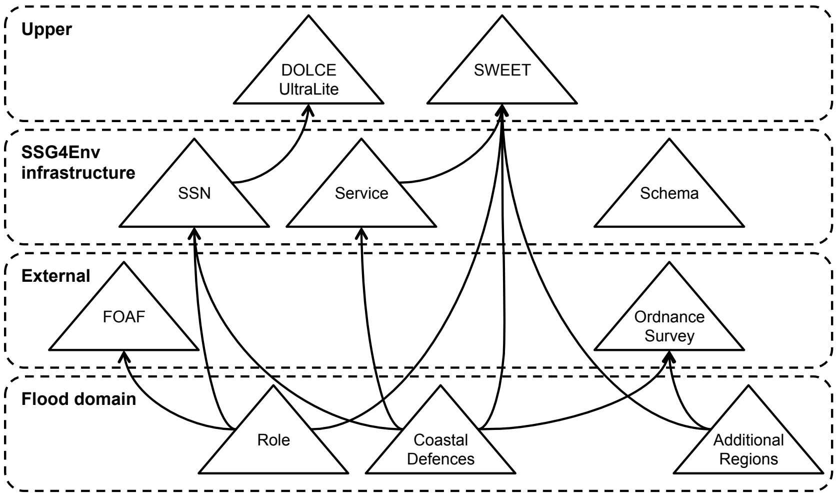


Fig. 1. The SemSorGrid4Env ontology network in the flood emergency planning scenario. The arrows indicate ontology reuse.

well as any domain-dependent information; (ii) Having a shared vocabulary to interoperate both across the internal infrastructure services, and between that infrastructure and external sources that adopt alternative approaches, e.g. OGC-SWE based ones [3]; and (iii) Discovering, accessing, and integrating information that is shared within the infrastructure.

Fig. 1 illustrates how the ontology network used in the flood emergency planning scenario is composed of different ontologies that can be classified in different layers according to whether the ontology represents: domain-specific information required for the scenario, information required for the infrastructure, or upper-level information used to facilitate interoperability among the other ontologies. These ontologies satisfy different knowledge representation requirements extracted during the development of the architecture and of the scenario prototype. (i) To represent sensor networks and their observed information about properties of certain features of interest. This is covered by the SSN ontology, developed by the W3C Semantic Sensor Network Incubator Group[7]. The SSN reuses the DOLCE+DnS UltraLite upper ontology[8]. (ii) To represent the services provided by the infrastructure and the datasets they provide access to. This is covered by the Service module that reuses the SWEET upper ontologies [15] and includes concepts from the ISO19119 standard on geographic information services [11]. (iii) To represent schema metadata about relations and relational streams. This is covered by the Schema module that extends, and corrects, an ontology for relational data and schema components [14]. (iv) To represent the geographic and administrative regions of the south coast of England. This is covered by the Ordnance Survey ontologies[9], which include the regions from

[7] http://www.w3.org/2005/Incubator/ssn/ (11 November 2010).

[8] http://www.loa-cnr.it/ontologies/DUL.owl accessed 11 November 2010.

[9] http://www.ordnancesurvey.co.uk/oswebsite/ontology/ (11 November 2010).

Great Britain, and by the Additional Regions ontology, which includes other regions needed in our scenario. (v) To represent those features of interest and their properties that are specific to the flood emergency planning scenario. This is covered by the Coastal Defences ontology. (vi) To represent the different roles involved in a flood emergency planning scenario. This is covered by the Roles ontology.

All the ontologies[10] have been implemented using OWL. While some of the ontologies presented here are specific to the flood warning scenario, e.g. Role, the architecture proposed in the next section is generic. Thus, it can be adapted to other situations by replacing the flood domain ontologies.

4 Semantic Sensor Web Architecture

The proposed service architecture gives rise to a semantic sensor web for environmental management that aims to meet the requirements identified in Section 2. The architecture (Fig. 2) comprises a set of services that can be composed into orchestrations to deliver the desired functionality. The architecture can interact with existing frameworks such as OGC-SWE [3], either as applications that exploit our services or as concrete resources that are wrapped to provide additional functionality, e.g. query-based access or semantic integration. A sample orchestration from the flood scenario is given in Fig. 5 and described in Section 5. The architecture is structured into three tiers, although a service may call any other service regardless of which tier they appear in. The *data tier* enables the publication and querying of data in its native format, i.e. a relational database can be queried using SQL and sensor data through a continuous query language such as SNEEql [7]. The *middleware tier* supports the discovery of relevant data as well as the reconciliation of data models and querying over these reconciled models. The *application tier* provides domain specific services and supports the transition from service-oriented web services to REST services [6]. The interfaces[11] offered by the web services in our architecture are given in Table 1. A full specification of the services, and the operations they support, can be found in [8]. The service architecture uses, where possible, existing web service standards.

The specification of the service interfaces, i.e. the WSDL definition of the service-oriented services, supports *well-formed* interactions with the services. However, it is anticipated that there will be multiple services which, at a functional level, satisfy a user's needs. To enable the user to make a *well-informed* choice between these services, the architecture uses *semantically annotated property documents* which describe the non-functional properties of a service, e.g. the datasets and their features, that are available from the service. The property document can be retrieved through the service interface operations, provided by all services, and their content is specified using the ontology network (Fig. 1).

In the following sections, we describe the services of the architecture and the role of the property document in supporting their activities. We illustrate these interactions with the property document for the CCO sensor data service,

[10] http://www.semsorgrid4env.eu/ontologies/ (7 December 2010)

[11] We use the term *interface* to mean a logical group of operations.

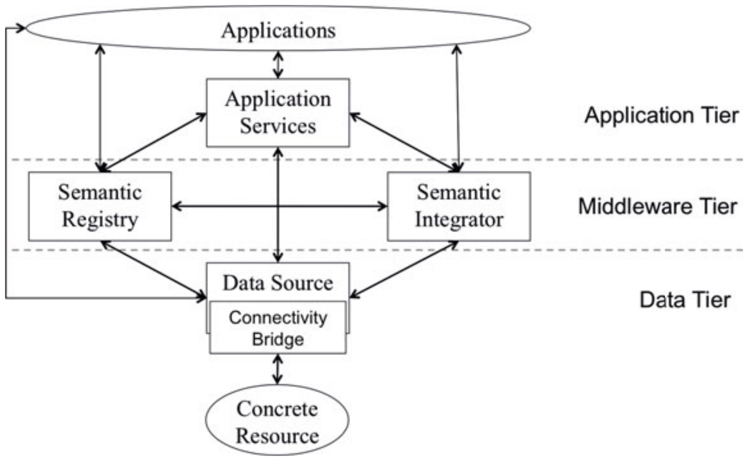


Fig. 2. Conceptual view of the service architecture. Boxes denote SemSorGrid4Env services, ovals denote external services/entities, arrows denote caller-callee relationships.

Table 1. Services and their interfaces. Interfaces shown in italics are optional.

Service Name	Interfaces Offered	Notes
Data Source Service	Service, *Integration*, *Query*, *Data Access*, *Subscription*	Must provide at least one of query, data access, or subscription interfaces.
Semantic Registry Service	Service, Registration, *Discovery*, *Query*, *Data Access*, *Subscription*	Either the discovery or the query interface must be provided.
Semantic Integrator Service	Service, Integration, Query, *Data Access*, *Subscription*	One of the data access or subscription interface must be offered to support queries over streams.
Application Services	REST	Interface design focuses on identifying and structuring web resources.

excerpts of which are shown in Fig. 3. Note, these declarations make use of stRDF [12], a spatiotemporal extensions for RDF that defines URIs of the form `&term`.

4.1 Data Source Services

Data source services provide the mechanism to publish data: either coming from a sensor network or some other data source, e.g. a database or another data service. Depending on the interfaces supported by the data service, operations are provided for querying, retrieving, and subscribing to data. A distributed query processing service can be offered, using the integration interface, which consumes data from other services that may only support the data access or subscription interfaces.

Data source services publish a property document about the data that they provide, and the mechanisms by which it may be accessed. The first part of the property document in Fig. 3 describes the interaction mechanisms provided,

```
 1  <service:WebService rdf:about="#cco-ws">
 2      <rdfs:label>Channel coastal observatory streaming data service</rdfs:label>
 3      <service:hasInterface rdf:resource="service:ssg4ePullStream"/>
 4      <service:hasDataset rdf:resource="#envdata_SandownPier_Tide"/>
 5      <service:hasDataset rdf:resource="#envdata_SandownPier_Met"/>
 6      ...
 7  </service:WebService>
 8  <sweet:Dataset rdf:about="#envdata_SandownPier_Tide">
 9      <rdfs:label>envdata_SandownPier_Tide</rdfs:label>
10      <service:coversRegion rdf:resource="&AdditionalRegions;SandownPierLocation"/>
11      <time:hasTemporalExtent rdf:datatype="&registry;TemporalInterval">
12          [2005, NOW]</time:hasTemporalExtent>;
13      <service:includesFeatureType rdf:resource="&CoastalDefences;Sea"/>
14      <service:includesPropertyType rdf:resource="&CoastalDefences;TideHeight"/>
15      <service:includesPropertyType rdf:resource="&CoastalDefences;WaveHeight"/>
16      <service:hasSchema rdf:resource="#envdata_SandownPier_Tide_Schema"/>
17      ...
18  </sweet:Dataset>
19  <schema:Stream rdf:about="#envdata_SandownPier_Tide_Schema">
20      <schema:extent-name>envdata_SandownPier_Tide</schema:extent-name>
21      <schema:hasAttribute rdf:resource="#HMax"/>
22      <schema:hasAttribute rdf:resource="#Tp"/>
23      ...
24  </schema:Stream>
25  <schema:Attribute rdf:about="#HMax">
26      <schema:attribute-name>HMax</schema:attribute-name>
27      ...
28  </schema:Attribute>
29  ...
```

Fig. 3. Snippets from the CCO sensor data web service semantic property document expressed in stRDF [12] using XML notation. We assume appropriate namespace declarations and represent omitted parts of the document with '. . . '.

i.e. the interfaces (line 3) and operations supported by the data service. The rest of the property document describes the data that is available through the service, which is not covered in the WSDL definition of the service.

A data source may publish one or more datasets, as per the WS-DAI standard [2]. Lines 4 and 5 show that the CCO sensor data service publishes multiple datasets including the two identified as `#envdata_SandownPier_Tide` and `#envdata_SandownPier_Met`. Each dataset is described in terms of its spatiotemporal and thematic coverage, and (where appropriate) its schema. Lines 8 to 18 provide details of the `#envdata_SandownPier_Tide` dataset. Specifically, lines 10 to 12 describe the spatiotemporal range of the dataset as providing data for the 'Sandown Pier' location and that the time range is from 2005 until the current time, represented with the distinguished literal 'NOW'. The types of features covered by the dataset are declared by the statements in line 13, which state that the `#envdata_SandownPier_Tide` dataset contains information about the CoastalDefences concept Sea. Lines 14 and 15 give the property types covered as the CoastalDefences concepts of TideHeight and WaveHeight. Where appropriate, e.g. for relational data sources, the property document also includes an ontological description of the schema of the dataset using the Schema ontology. Line 16 declares that the `#envdata_SandownPier_Tide` dataset has a schema described by the resource `#envdata_SandownPier_Tide_Schema`. Lines 19 to 28 describe the relational schema of the `#envdata_SandownPier_Tide` data stream:

its name, attributes, types of the attributes, primary key, and timestamp attribute. It is this information that enables a distributed query service, which itself can be seen as a data service, to support queries over external data sources.

4.2 Semantic Registry Service

The registry service supports the discovery of relevant services. The registration interface enables the registration of service descriptions, viz. the information contained in the service property document. Since a registry service can be seen as a data service, it provides the same interfaces, with the data access and subscription interfaces providing support for long-lived queries. In the SemSorGrid4Env implementation, the registry service data store is populated with the content of the service property documents. It supports query-based access through the spatiotemporal SPARQL extension stSPARQL [12]. External services, such as those defined by the OGC [3,5], can be manually entered.

The information contained in the property document enables applications to discover relevant data sources using application domain specific terms. For example, a flood response manager responsible for the Portsmouth region would like to discover sensor data sources for that region `&AdditionalRegions;Solent`. From Fig. 3, we see that the CCO sensor data web service `#CCO-WS` exposes a stream interface service:ssg4ePullStream (line 3) taken from the Service ontology, and that the `#envdata_SandownPier_Tide` dataset has the spatial coverage `&AdditionalRegions;SandownPierLocation` (line 10) using the Additional Regions ontology. Due to the use of spatial constraints in the definitions of the regions in the ontology, the registry is able to deduce that the `#envdata_SandownPier_Tide` dataset is relevant for the application. This is because the location `&AdditionalRegions;SandownPierLocation` is contained in the region defined for the Solent `&AdditionalRegions;Solent`. Thus, `#CCO-WS` is a relevant source for the flood response manager.

4.3 Semantic Integration Service

The integrator supports the creation and querying of an information space over independent heterogeneous data sources. The integration interface enables the creation of an integrated data model by supplying a mapping document relating the data sources to the global model. The query interface enables ontology-based access to the data sources [4]. That is, a user or application can express a query in terms of ontological concepts and the integration service translates it into a set of queries over the relevant sources, retrieves the answers, and translates them into ontological instances. Note, the integration service is a data source, thus it provides the same interfaces, with either the data access or subscription interface being offered to support long-lived continuous queries over sensor data.

The integrator uses the property documents of the data sources involved in an integration to select the appropriate interaction mechanism. That is, for each data source, the property document informs the integrator of the interfaces and query language supported. From the example document given in Fig. 3, the integrator can infer that the `#CCO-WS` only supports data retrieval through the pull-stream interface (line 3), i.e. it does not support queries, and that the schema of the `#envdata_SandownPier_Tide` is as described (lines 19-28). The integrator can also query the registry to discover suitable distributed query processing services to invoke in answering queries over the integrated data resource. The property documents of the data sources also aid the integrator in the creation of the property document that describes the integrated data model. In particular, its spatiotemporal and thematic content. Note that the semantic representation of a source schema, as provided in lines 19-28 of Fig. 3, can help mapping tools in understanding the schema of a data source and, therefore, in the creation of mappings between source schemas and an ontology for the domain of interest.

4.4 Application Services

The services in the application tier of the architecture provide support for web-based applications and mashups to interact with the services and data sources in the architecture. On the whole, web-based applications and mashups interact through a resource-oriented approach [6], i.e. they interact through HTTP calls, viz. GET, POST, PUT, and DELETE. As such, the application services bridge the gap between a resource-oriented viewpoint that prevails for user fronting applications and the service-oriented viewpoint that is preferred in middleware and back-end system components. By exposing a REST interface, the application services enable web-based applications to request content based on formats that they can process and display to the user, e.g. GML, GeoJSON, and HTML [13].

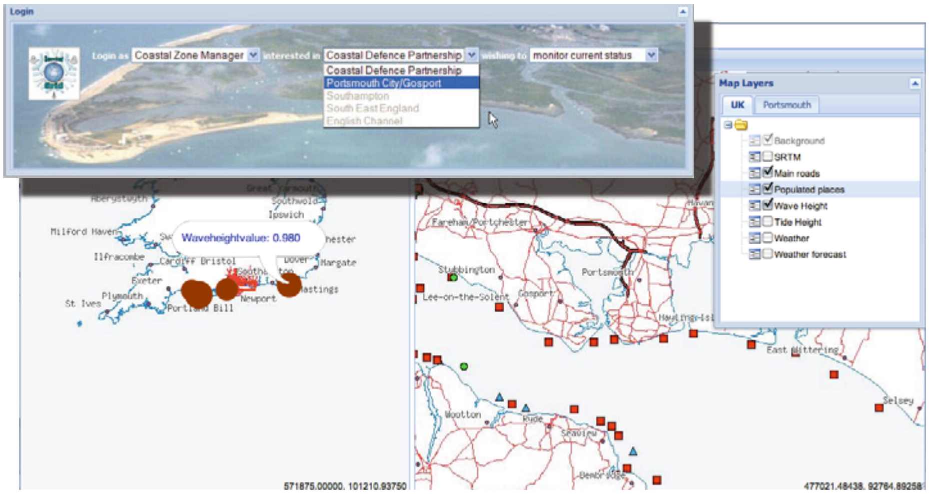
The application services, and ultimately the applications that rely on them, exploit information contained in the property documents associated with services. Property documents enable application services to locate relevant services through the registry and gain insight into how to interact with them. They also enable applications to integrate data sources through integration services.

4.5 Summary

The property document enables well-informed interactions between the services in the architecture, and is instrumental in all aspects of the functionality offered. It is not a requirement to provide the semantic property document, and no parts of it are mandatory. As such, external services, e.g. those defined by OGC [3,5], can be incorporated into the architecture. However, by describing the non-functional properties, particularly the spatiotemporal and thematic coverage of its data, in a property document a service can be discovered through the registry, and used by the integrator and application services in a seamless manner.

5 The Flood Scenario Deployment

We now show how the architecture enables the flood emergency planning scenario described in Section 2. We show how a flood emergency planner (the user) can



Login

Login as: Coastal Zone Manager; interested in: Coastal Defence Partnership; wishing to: monitor current status

Fig. 4. Screenshots from the flood emergency response Web application available from http://webgis1.geodata.soton.ac.uk/flood.html

add a source to identify when an overtopping event is detected. The scenario assumes the existence of: (i) a semantic registry service at some well known location; (ii) several data services which have registered their semantic property documents with the registry service; and (iii) a distributed query service, an integration service, and application services to support the web application.

The user accesses the web application through the login screen shown in the top left of Fig. 4. When logging into the application, the user selects their role, the region they are responsible for, and the task that they wish to conduct. The options in the selection boxes are populated with terms from the ontology network (Fig. 1), e.g. the choice of role comes from the concepts in the Role ontology. This provides an initial characterisation of the data that is relevant for the interaction, i.e. the values provided parameterise the queries sent to the registry in order to discover relevant data sources. For the login selection shown, the registry query is parameterised to discover data sources for the Portsmouth area for a Coastal Zone Manager who wishes to monitor the current status.

The result of the login process is shown in the main screenshot in Fig. 4. The user is presented with two map views based on the region selected, viz. Portsmouth. The left pane shows a zoomed-out map providing context while the right pane shows a zoomed-in map on the region selected. Both maps have been superimposed with layers presenting data from a variety of sources that satisfied the queries sent to the registry. The available layers are shown in the Map Layers pop-up window, from which the user can select the layers they wish to be displayed. In the example shown, three layers have been selected for display. Two of these—showing the main roads and the populated areas—have been retrieved from OGC-WMS services [5] that have been manually registered, i.e. details of the service and the dataset have been entered into the registry through a web form. The current wave height values retrieved

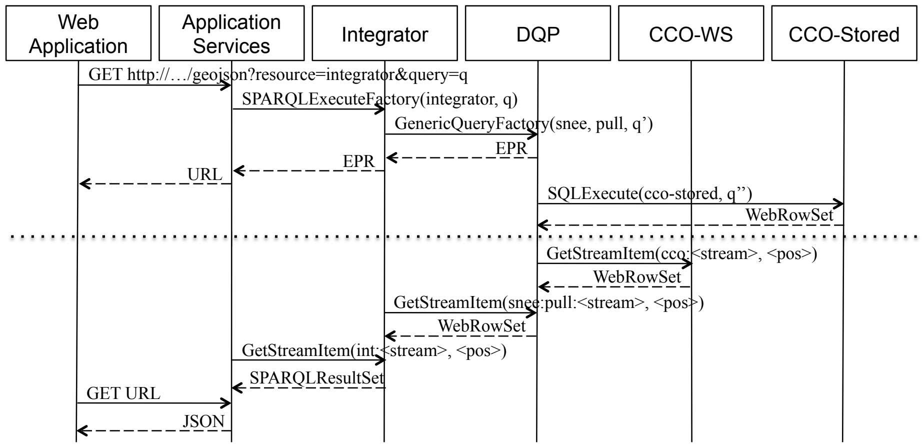

Web Application
Application Services
Integrator
DQP
CCO-WS
CCO-Stored
GET http://…/geojson?resource=integrator&query=q
SPARQLExecuteFactory(integrator, q)
GenericQueryFactory(snee, pull, q')
EPR
EPR
URL
SQLExecute(cco-stored, q")
WebRowSet
GetStreamItem(cco:<stream>, <pos>)
WebRowSet
GetStreamItem(snee:pull:<stream>, <pos>)
WebRowSet
GetStreamItem(int:<stream>, <pos>)
SPARQLResultSet
GET URL
JSON

Fig. 5. Interaction diagram showing how data is integrated across heterogeneous data sources. Operations below the dotted line are repeated periodically.

from the available sensor networks for the region are juxtaposed on top. The values are displayed as red circles: the larger the circle, the higher the wave value measured.

To support identifying an overtopping event requires data from heterogenous sources with different schemas and data modalities, viz. stored and sensed, to be integrated. The required orchestration is depicted in Fig. 5 which shows a web application retrieving data through an integrator that exposes an ontological view of the source data. Note that the orchestration assumes that the integrated resource has already been created, i.e. the mapping document relating the data sources to the ontological view has already been passed to the integrator. The web application supports the user in discovering potential data services for detecting overtopping events based on the contents of the property documents stored by the registry service (not shown in the orchestration in Fig. 5). The web application then supports the user in characterising an overtopping event as a query over the ontological view, hiding all the complexities of the required orchestration. The web application uses a RESTful interface offered by an application service to pass the query as a service call to the integrator. The integrator translates the query over the ontological view into a query expressed in terms of the source schemas. The integrator instantiates a distributed query processing service (DQP) to evaluate the query over the sources. As the query is evaluated, answers are periodically retrieved through the interactions shown below the dotted line in Fig. 5. The rate at which the DQP service polls its sources is controlled by the rate declared in the property document of each source. Similarly, the rates at which the integrator and the application poll their respective source is controlled by the property document declarations.

6 Related Work

We describe related work in its ability to satisfy the requirements identified in Section 2.

The Open Geospatial Consortium Sensor Web Enablement (OGC-SWE) [3] defines a set of XML data models for describing sensors and their data, as well as a set of web services for publishing sensor data, locating data, and receiving alerts about the data. A reference implementation of the framework was developed in the SANY project [16]. The framework can be seen to satisfy R4, and provides support for satisfying R5. However, data access patterns are limited by the service interfaces and there is no support for declarative query languages. As such, it does not fully satisfy R1. Data is published according to their XML data models, which is not always possible with autonomous data sources. Thus, they do not satisfy R3. OGC-SWE does not fully meet R2: there is support for unmediated merging of sensor and stored data but not for correlating it. We note that the GetCapabilities operation provided by the services provide support for the functional properties in our property documents but not the spatiotemporal or thematic properties. Henson *et al.* [10] have extended the sensor observation service by semantically annotating the data. Our proposal goes beyond this by using semantics to support the discovery, integration, and mashup of data stemming from autonomous heterogeneous data sources.

Global Sensor Network (GSN) [1] is a middleware platform for the deployment and programming of sensor networks. It allows for the abstraction of sensor nodes as data sources irrespective of the underlying hardware and provides query processing capabilities within the middleware. It enables a data-oriented view of sensor networks and the processing of queries over that data. GSN satisfies R1 and R2 provided that the data is all published in the same data model. It does not satisfy the other requirements.

Collaborative Oceanography [17] used semantic annotations to support the reuse of oceanographic data. Their approach relied on a centralised triple store containing the annotations and the manual mashup of data based on these annotations. Our approach provides support for semantic integration and mashup of heterogeneous data sources.

7 Conclusions

We have presented a service architecture for providing support to semantic sensor web applications. The architecture provides a semantically integrated information space for sensed and stored data drawn from heterogeneous autonomous data sources. The architecture enables rapid development of thin applications (mashups) over this information space through the use of (i) declarative queries to describe the data need, both for locating data based on its spatiotemporal and thematic coverage, and for integrating and accessing data, and (ii) semantically annotated property documents which support well-informed interactions between the architecture services.

Five high-level requirements were identified, from the application use case presented, that are considered to be relevant for a broad range of applications.

R1 *Accurate characterisation of conditions that define an event.* This is satisfied by the use of declarative queries to retrieve data. Underlying data sources can be queried through their native query language, e.g. SQL for relational databases and SNEEql for relational streams coming from sensor networks, while the integrator provides for ontology-based queries expressed in a SPARQL extension for streaming data, viz. $\text{SPARQL}_{\text{Stream}}$. Data sources that do not support query-based data retrieval can be queried through the use of a distributed query service.

R2 *Correlation of data of differing modalities.* This is satisfied through the use of query languages that provide support for correlating sensed and streaming data, e.g. SNEEql and $\text{SPARQL}_{\text{Stream}}$, as well as the use of application services to create layers for juxtaposing data.

R3 *Integrating data coming from heterogeneous data models.* This is satisfied by the use of ontologies to provide a common vocabulary and data model. Data sources publish semantically annotated property documents that use concepts available in ontologies to describe their datasets and interfaces. The registry, integrator, and application services can automatically exploit the content of the property documents due to the use of ontological terms.

R4 *Discovery of relevant data sources.* This is satisfied by the semantic registry service and the publication of semantically annotated property documents by the data services. The registry service uses the statements contained in the property documents to identify relevant data services by answering declarative stSPARQL queries which characterise the spatiotemporal and thematic data needs of applications and users.

R5 *Presentation and control of information.* This is satisfied by the interaction between web-based applications and mashups with the application services. By providing REST style interfaces, offering data in a variety of formats including mapping layers, and using domain specific concepts drawn from (where possible) standard domain ontologies, the application services enable the rapid development of thin web-based applications without sacrificing the functionality and control offered to the user.

The next steps for the implementation of our architecture are to provide services which can push data from the sources through the architecture, and to provide mechanisms for interacting with existing infrastructures such as OGC-SWE. For future work we intend to investigate offering configurable mechanisms for supporting REST interfaces to integrated information spaces. We will also perform a user evaluation with coastal managers from the Solent region.

Acknowledgments. This work has been supported by the European Commission project SemSorGrid4Env (FP7-223913).

References

1. Aberer, K., Hauswirth, M., Salehi, A.: Infrastructure for data processing in large-scale interconnected sensor networks. In: 8th International Conference on Mobile Data Management (MDM 2007), pp. 198–205 (2007)

2. Antonioletti, M., Krause, A., Paton, N.W., Eisenbrg, A., Laws, S., Malaika, S., Melton, J., Pearson, D.: The WS-DAI family of specifications for web service data access and integration. SIGMOD Record 35(1), 48–55 (2006)
3. Botts, M., Percivall, G., Reed, C., Davidson, J.: OGC® sensor web enablement: Overview and high level architecture. In: Nittel, S., Labrinidis, A., Stefanidis, A. (eds.) GSN 2006. LNCS, vol. 4540, pp. 175–190. Springer, Heidelberg (2008)
4. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
5. de la Beaujardiere, J.: OpenGIS® web map server implementation specification. Standard Specification 06-042, Open Geospatial Consortium Inc. (2006)
6. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, Information and Computer Science, University of California, Irvine, California, USA (2000)
7. Galpin, I., Brenninkmeijer, C.Y.A., Gray, A.J.G., Jabeen, F., Fernandes, A.A.A., Paton, N.W.: SNEE: a query processor for wireless sensor networks. Distributed and Parallel Databases 29(1-2), 31–85 (2010)
8. Gray, A.J.G., Galpin, I., Fernandes, A.A.A., Paton, N.W., Page, K., Sadler, J., Kyzirakos, K., Koubarakis, M., Calbimonte, J.P., Garcia, R., Corcho, O., Gabaldón, J.E., Aparicio, J.J.: SemSorGrid4Env architecture – phase II. Deliverable D1.3v2, SemSorGrid4Env (December 2010), http://www.semsorgrid4env.eu/files/deliverables/wp1/D1.3v2.pdf
9. Hart, J.K., Martinez, K.: Environmental sensor networks: A revolution in earth system science? Earth Science Reviews 78, 177–191 (2006)
10. Henson, C., Pschorr, J., Sheth, A.P., Thirunarayan, K.: SemSOS: Semantic sensor observation service. In: International Symposium on Collaborative Technologies and Systems, CTS 2009 (2009)
11. Geographic information – services. International Standard ISO19119:2005, ISO (2005)
12. Koubarakis, M., Kyzirakos, K.: Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 425–439. Springer, Heidelberg (2010)
13. Page, K., De Roure, D.C., Martinez, K., Sadler, J., Kit, O.: Linked sensor data: RESTfully serving RDF and GML. In: International Workshop on Semantic Sensor Networks, pp. 49–63 (2009)
14. Pérez de Laborda, C., Conrad, S.: Relational.OWL: a data and schema representation format based on OWL. In: 2nd Asia-Pacific Conference on Conceptual Modelling (APCCM 2005), Newcastle, Australia, pp. 89–96 (2005)
15. Raskin, R.G., Pan, M.J.: Knowledge representation in the semantic web for earth and environmental terminology (SWEET). Computers and Geosciences 31(9), 1119–1125 (2005)
16. Schimak, G., Havlik, D.: Sensors anywhere – sensor web enablement in risk management applications. ERCIM News 76, 40–41 (2009)
17. Tao, F., Campbell, J., Pagnani, M., Griffiths, G.: Collaborative ocean resource interoperability: Multi-use of ocean data on the semantic web. 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. 753–767. Springer, Heidelberg (2009)

Zhi# – OWL Aware Compilation

Alexander Paar[1] and Denny Vrandečić[2]

[1] University of Pretoria, Hillcrest, Pretoria 0002, South Africa
alexpaar@acm.org
[2] KIT Karlsruhe Institute of Technology, Karlsruhe, Germany, and
ISI Information Sciences Institute, University of Southern California, USA
denny.vrandecic@kit.edu

Abstract. The usefulness of the Web Ontology Language to describe domains of discourse and to facilitate automatic reasoning services has been widely acknowledged. However, the programmability of ontological knowledge bases is severely impaired by the different conceptual bases of statically typed object-oriented programming languages such as Java and C# and ontology languages such as the Web Ontology Language (OWL). In this work, a novel programming language is presented that integrates OWL and XSD data types with C#. The Zhi# programming language is the first solution of its kind to make XSD data types and OWL class descriptions first-class citizens of a widely-used programming language. The Zhi# programming language eases the development of Semantic Web applications and facilitates the use and reuse of knowledge in form of ontologies. The presented approach was successfully validated to reduce the number of possible runtime errors compared to the use of XML and OWL APIs.

Keywords: OWL DL, C#, Zhi#, ontologies, programmability.

1 Introduction

A typical OWL DL [12] knowledge base comprises two components: a *TBox* defining the formal relations between the classes and properties of the ontology; and an *ABox* containing assertional knowledge about the individuals of the ontology. The TBox is often regarded to be the more stable part of the ontology, whereas the ABox may be subject to occasional or even constant change. In particular, modifications may lead to an ABox that violates constraints given by the TBox, such as cardinality constraints or value space restrictions of OWL datatype properties. Up to now, ontological knowledge bases are modified using APIs, which are provided by a variety of different ontology management systems [7,21]. From a software developer's perspective, there is no support for statically detecting illegal operations based on given terminologies (e.g., undefined classes, invalid datatype property values) and conveniently integrating ontological classes, properties, and individuals with the program text of a general purpose programming language.

A common difficulty of widely used OWL APIs and the usage of wrapper classes to represent entities of an ontology are the different conceptual bases of types and instances in a programming language and classes, properties, individuals, and XML Schema Definition [4] data type values in OWL DL. In particular, the Web Ontology Language reveals the following major differences to object-oriented programming languages and database management systems:

– In contrast to object-oriented programming languages, OWL provides a rich set of class constructors. For example, classes can be described via cardinality and value restrictions on properties (e.g., a small meeting is a meeting with at most three participants).
– OWL class descriptions can be automatically classified in a subsumption hierarchy. Imitating this inherent behavior of ontological knowledge bases using a hierarchy of programming language wrapper classes would result in reimplementing a complete OWL DL reasoner.
– Unlike object-oriented programming languages or database management systems, OWL makes the *open world assumption* (OWA), which codifies the informal notion that in general no single observer has complete knowledge. The open world assumption limits the deductions a reasoner can make. In particular, it is not possible to infer that a statement is false just because it is not stated explicitly. The OWA is closely related to the monotonic nature of first-order logic (i.e. adding information never falsifies previous conclusions).
– The Web Ontology Language does not make the *unique name assumption* (UNA). In contrast to logics with the unique name assumption, different ontological individuals do not necessarily refer to different entities in the described world. In fact, two individuals can be inferred to be identical (e.g., values of functional object properties). In OWL, it is also possible to explicitly declare that two given named individuals refer to the same entity or to different entities.
– Unlike object-oriented programming languages, ontological properties in OWL DL are not defined as part of class definitions but form a hierarchy of their own (i.e. property centric modeling).
– In OWL, property domain and range declarations are not constraining. Instead, the declared domain and range of an OWL property is used to infer the types of the subjects and objects of assertions, respectively. Thus, OWL properties facilitate *ad hoc relationships* [13] between entities that may not have been foreseen when a class was defined.

The Zhi#[1] programming language is a superset of conventional C# version 1.0 boasting programming language inherent support for XML Schema Definition and the Web Ontology Language. Zhi#'s *OWL aware compilation* includes static typing and type inference for XSD data types and a combination of static typing and dynamic checking for OWL DL ontologies. XSD constraining facets and ontological reasoning were integrated with host language features such as method overriding, user-defined operators, and runtime type checks. For the lack of space,

[1] Zhi (Chinese): Knowledge, information, wisdom.

only elementary examples of an integrated use of XSD data types and ontological class descriptions in Zhi# are presented. The Zhi# programming language is implemented by a compiler framework [16] that is – by means of plug-ins – extensible with external type systems[2]. Detailed descriptions of the compiler framework and the XSD and OWL plug-ins can be found in [17]. Zhi# programs are compiled into conventional C# and are interoperable with .NET assemblies. The Zhi# approach is distinguished by a combination of features that is targeted to make ontologies available in an object-oriented programming language using conventional object-oriented notation.

In contrast to naïve approaches that are based on the generation of wrapper classes for XSD and OWL types, no code generation in form of an additional class hierarchy is required in Zhi#. Instead, ontologies are integrated into the programming language, which facilitates OWL aware compilation including type checking on the ontology level. At runtime, the results of ontological reasoning influence the execution of Zhi# programs: Zhi# programs don't just execute, they reason. The underlying ontology management system can be substituted without recompilation of Zhi# programs. The Zhi# programming language provides full support for XSD data types. Thus, Zhi# can compensate for datatype agnostic OWL APIs. Zhi# programs can be used concurrently with API-based knowledge base clients to allow for a smooth migration of an existing code-base.

2 The Zhi# Programming Language

The type system of the C# programming language implements *nominal* subtyping. In nominative type systems type compatibility is determined by explicit declarations. A type is a subtype of another if and only if it is explicitly declared to be so in its definition. The XML Schema Definition type system extends nominal subtyping with *value space-based* subtyping. An atomic data type is a subtype of another if it is explicitly declared to be so in its definition or if its value space (i.e. the set of values for a given data type) is a subset of the value space of the other type. The subset relation of the types' value spaces is sufficient. The two types do not need to be in an explicitly declared derivation path. In the Web Ontology Language, nominal subtyping is augmented by *ontological reasoning*. An inferred class hierarchy can include additional subsumption relations between class descriptions. Ontological individuals can be explicitly declared to be of a given type and they can be inferred to be in the extension of further class descriptions. Some object-oriented programming languages provide a limited set of isomorphic mappings from XSD data types to programmatic types. In general, however, compilers for programming languages such as Java or C# are unaware of the subtyping mechanisms that are used for XSD and OWL.

The Zhi# programming language is a proper superset of ECMA 334 standard C# version 1.0 [6]. The only syntactical extensions, which are entailed by Zhi#'s

[2] Given the general extensibility of the Zhi# programming language with external type systems and for the sake of brevity, in this work, XSD data types and OWL class descriptions are subsumed under the term *external types*.

extensibility with respect to external type systems, are the following: External types (i.e. XSD data types and OWL class descriptions) can be included using the keyword *import*, which works analogously for external types like the C# *using* keyword for .NET programming language type definitions. It permits the use of external types in a Zhi# namespace such that, one does not have to qualify the use of a type in that namespace. An *import* directive can be used in all places where a *using* directive is permissible. As shown below, the *import* keyword is followed by a *type system evidence*, which specifies the external type system (i.e. XSD or OWL). Like *using* directives, *import* directives do not provide access to any nested namespaces.

import *type_ system_ evidence alias* = *external_ namespace*;

In Zhi# program text that follows an arbitrary number of *import* directives, external type and property references must be fully qualified using an alias that is bound to the namespace in which the external type is defined. Type and property references have the syntactic form *#alias#local_ name* (both the namespace alias and the local name must be preceded by a '#'-symbol).

External types can be used in Zhi# programs in all places where .NET types are admissible except for type declarations (i.e. external types can only be imported but not declared in Zhi# programs). For example, methods can be overridden using external types, user defined operators can have external input and output parameters, and arithmetic and logical expressions can be built up using external objects. Because Zhi#'s support for external types is a language feature and not (yet) a feature of the runtime, similar restrictions to the usage of external types apply as for generic type definitions in the Java programming language (e.g., methods cannot be overloaded based on external types from the same type system at the same position in the method signature).

In Zhi# programs, types of different type systems can cooperatively be used in one single statement. As shown in line 5 in the following code snippet, the .NET *System.Int32* variable *age* can be assigned the XSD data type value of the OWL datatype property *hasAge* of the ontological individual Alice.

```
1 import OWL chil = http://chil.server.de;
2 class C {
3   public static void Main() {
4     #chil#Person alice = new #chil#Person("#chil#Alice");
5     int age = alice.#chil#hasAge;
6   }
7 }
```

2.1 Static Typing

C# is a statically typed programming language. Type checking is performed during compile time as opposed to runtime. As a consequence, many errors can be caught early at compile time (i.e. fail-fast), which allows for efficient execution at runtime. This section describes the static type checks that can be performed on ontological expressions in Zhi# programs.

Syntax checks. The most fundamental compile-time feature that Zhi# provides for OWL is checking the existence of referenced ontology elements in the imported terminology. The C# statements below declare the ontological individuals A and B. Individual B is added as a property value for property R of individual A. For the sake of brevity, in this work, the URI fragment identifier "#" may be used to indicate ontology elements in Zhi# programs instead of using fully-qualified names. The object o shall be an instance of an arbitrary OWL API. The given code is a well-typed C# program. It may, however, fail at runtime if in the TBox of the referenced ontology classes A and B and property R do not exist.

```
1  IOWLAPI o = [...];
2  o.addIndividual("#A", "#A");
3  o.addIndividual("#B", "#B");
4  o.addObjectPropertyValue("#A", "#R", "#B");
```

In Zhi#, the same declarations can be rewritten as shown below, turning the ontological properties into first-class citizens of the programming language. As a result, the Zhi# compiler statically checks if class descriptions A and B and property R exist in the imported ontology and raises an error if they are undefined. Note how in line 4 the RDF triple [A R B] is created in the shared ontological knowledge base using object-oriented member access.

```
1  import OWL alias = ontology namespace;
2  #A a = new #A("#A");                    // variable a refers to individual A
3  #B b = new #B("#B");                    // variable b refers to individual B
4  a.#R = b;
```

Creation of individuals. In C#, the *new*-operator can be used to create objects on the heap and to invoke constructors. In Zhi#, the *new*-operator can also be used to return ontological individuals in a knowledge base as follows.

```
1  #Event e = new #Meeting("#Briefing");  // Because Meeting ⊑ Event...
2  #Meeting m = new #Event("#Briefing");  // ...this assignment is rejected
```

Zhi# provides a constructor for OWL class instances that takes the URI of the individual. As in conventional C#, the *new*-operator cannot be overloaded. In contrast to .NET objects, ontological individuals are not created on the heap but in the shared ontological knowledge base, and as such they are subject to ontological reasoning. This is also in contrast to naïve approaches where wrapper classes for ontological classes are instantiated as plain .NET objects. Zhi# programs use handles to the actual individuals in the shared ontological knowledge base. Also note that an existing individual in the ontology with the same URI is reused, following Semantic Web standards. As for assignments of .NET object creation expressions to variables or fields, the type of the individual creation expression must be subsumed by the type of the lvalue based on the class hierarchy (see line 2 in the code snippet above). Zhi# supports covariant coercions for ontological individuals and arrays of ontological individuals.

Disjoint classes. In OWL DL, classes can be stated to be disjoint from each other using the *owl:disjointWith* constructor. It guarantees that an individual that is a member of one class cannot simultaneously be a member of the other class. In the following code snippet, the Zhi# compiler reports an error in line 2 for the disjoint classes *MeetingRoom* and *LargeRoom*.

```
1  #LargeRoom l = [...];                        // LargeRoom ⊑ ¬MeetingRoom
2  #MeetingRoom m = (#MeetingRoom) l;    // ⇝ InvalidCastException
```

If a property relates an individual to another individual, and the property has a class as its range, then the other individual must belong to the range class. Assuming the OWL object property *takesPlaceInAuditorium* relates *Lecture*s with *LargeRoom*s, line 2 in the following code snippet results in a compile-time error due to the disjointness of *MeetingRoom* and *LargeRoom*. Property domain declarations are treated analogously.

```
1  #Lecture l = [...];
2  l.#takesPlaceInAuditorium = new #MeetingRoom([...]);
```

Disjoint XSD data types. In Zhi#, a "frame-like" view on OWL object properties is provided by the *checked*-operator used in conjunction with assignments to OWL object properties (see Section 2.2). For assignments to OWL datatype properties in Zhi# programs, the "frame-like" composite view is the default behavior. The data type of the property value must be a subtype of the datatype property range restriction. The following assignment in line 2 fails to type-check for an OWL datatype property *hasCapacity* with domain *MeetingRoom* and range *xsd#byte* because in Zhi# programs the literal *23.5* is interpreted as a .NET floating point value (i.e. *xsd#double*), which is disjoint with the primitive base type of *xsd#byte* (i.e. *xsd#decimal*).

```
1  #MeetingRoom r = [...];
2  r.#hasCapacity = 23.5;
```

The XSD compiler plug-in allows for downcasting objects to compatible XSD data types (i.e. XSD types that are derived from the same primitive base type). The assignment in line 3 in the following Zhi# program is validated by a downcast. In general, this may lead to an *InvalidCastException* at runtime, which prevents OWL datatype properties from being assigned invalid property values.

```
1  int i = [...];
2  #MeetingRoom r = [...];
3  r.#hasCapacity = (xsd#byte) i;
```

Properties. Erik Meijer and Peter Drayton note that "at the moment that you define a [programming language] class *Person* you have to have the divine insight to define all possible relationships that a person can have with any other possible object or keep type open" [13]. Ontology engineers do not need to make early commitments about all possible relationships that instances of a class may have. In Zhi# programs, ontological individuals can be related to other individuals and XSD data type values using an object-oriented notation. In contrast to authoritative type declarations of class members in statically typed object-oriented programming languages, domain and range declarations of OWL object properties are used to infer the types of the subject (i.e. host object) and object (i.e. property value). Hence, the types of the related individuals do not necessarily need to be subsumed by the domain and range declarations of the used object property *before* the statement. The only requirement here is that the related individuals are not declared to be instances of classes disjoint to the declared domain and range. In the following Zhi# program, the ontological individuals referred to by *e* and *l* are inferred to be not only an *Event* and a *Location* but also a *Lecture* and a *LargeRoom*, respectively.

```
#Event e = [...];                        // e refers to E, E:Event
#Location l = [...];                     // l refers to L, L:Location
e.#takesPlaceInAuditorium = l;  // E:Lecture, L:LargeRoom
```

Both for OWL object and non-functional OWL datatype properties the property assignment semantics in Zhi# are *additive*. The following assignment statement *adds* the individual referred to by *b* as a value for property *R* of the individual referred to by *a*; it does not remove existing triples in the ontology.

```
a.#R = b;
```

Correspondingly, property access expressions yield arrays of individuals and arrays of XSD data type values for OWL object properties and non-functional OWL datatype properties, respectively, since an individual may be related to more than one property value. Accordingly, the type of OWL object property and non-functional OWL datatype property access expressions in Zhi# is always an array type, where the base type is the range declaration of the property.

The type of an assignment to an OWL object property and a non-functional OWL datatype property is always an array type, too. This behavior is slightly different from the typical typing assumptions in programming languages. Because the assignment operator (=) cannot be overloaded in .NET, after an assignment of the form $x = y = z$ all three objects can be considered equal based on the applicable kind of equivalence (i.e. reference and value equality). The same is not always true for assignments to OWL properties considering the array ranks of the types of the involved objects. In the following cascaded assignment expression, the static type of the expression *b.#R = c* is `Array Range`(R) because individual B may be related by property *R* to more individuals than only C. As a result, with the following assignment in Zhi#, individual A is related by property *R* to *all* individuals that are related to individual B by property *R*.

```
a.#R = b.#R = c;  // a, b, and c refer to individuals A, B, and C, respectively
```

Ontological equality. In Zhi#, the equality operator (==) can be used to determine if two ontological individuals are identical (i.e. refer to the same entity in the described world). The inequality operator (!=) returns true if two individuals are known (either explicitly or implicitly) to be *not* identical. Note that the inequality operator is thus *not* implemented as the logical negation of == as individuals can be unknown to be identical or different.

Auxiliary properties and methods. The Zhi# compiler framework supports a full-fledged object-oriented notation for external types. In particular, compiler plug-ins can provide methods, properties, and indexers for static references and instances of external types. The OWL compiler plug-in implements a number of auxiliary properties and methods for ontological classes, properties, and individuals in Zhi# programs. For example, in order to remove property value assertions from the ontology, the OWL compiler plug-in provides the auxiliary methods *Remove* and *Clear* for OWL properties to remove one particular value and all values for an OWL property of the specified individual, respectively. In line 2 in the following code snippet, the ontological individual B is removed as a property value for property *R* of individual A. In line 3, all property values for property *R* of individual A are removed.

```
1  #A a = new #A("#A");              // Note: Other clients of the ontological know-
2  a.#R.Remove(new #B("#B"));        // ledge base could have added B as a property
3  a.#R.Clear();                     // value between the execution of line 1 and 2.
```

As a second example, for static references of OWL classes the auxiliary properties *Exists*, *Count*, and *Individuals* are defined. The *Exists* property yields true if individuals of the given type exist in the ontology, otherwise false. *Count* returns the number of individuals in the extension of the specified class description. *Individuals* yields an array of defined individuals of the given type. The *Individuals* property is generic in respect of the static type reference on which it is invoked. In the following array definition, the type of the property access expression *#Person.Individuals* is `Array` *Person* (and not `Array` *Thing*). Accordingly, it can be assigned to variable *persons* of type `Array` *Person.*

```
#Person[] persons = #Person.Individuals;
```

Note that all described functionality is provided in a "pay-as-you-go" manner: in Zhi#, there is no runtime performance or code size overhead for conventional C# code and Zhi# programs that do not use external type definitions.

2.2 Dynamic Checking

In a statically typed programming language such as C# the possible types of an object are known at compile time. Unfortunately, the non-contextual property centric data modeling features of the Web Ontology Language render static type checking only a partial test on Zhi# programs. As a result, the OWL plug-in for the Zhi# compiler framework and the Zhi# runtime library facilitate dynamic checking of ontological knowledge bases.

Ontological individuals can be in the extensions of a number of different class descriptions. In the same way, explicitly made RDF type assertions may be inconsistent with particular property values or the number of values for a particular property of an individual. More severely, ontological knowledge bases are subject to concurrent modifications via interfaces of different levels of abstraction (e.g., RDF triples, logical concept view). In Zhi#, before each single usage of an individual 1) the individual is dynamically checked to be in the extension of the declared class and 2) the knowledge base is checked to be consistent; an exception is thrown if either is not the case.

Runtime type checks. Reasoning is used to infer the classes an individual belongs to. This corresponds to the use of the *instanceof* and *is*-operator in Java and C#, respectively. In Zhi#, the *is*-operator is used to determine whether an individual is in the extension of a particular class description. The use of the *is*-operator is completely statically type-checked both on the programming language and the ontology level. For example, the Zhi# compiler will detect if an individual will never be included by a class description that is disjoint with its asserted type. See the Zhi# program in Section 3 for an exemplary use of the *is*-operator.

Checked property assignments. In general, neither domain nor range declarations of OWL properties are constraints. This is in contrast to frame languages and object-oriented programming languages. In statically typed object-oriented programming languages such as C#, properties are declared as class members. The domain of a property corresponds to the type of the containing host object. Only instances of the domain type can have the declared property. The range of a property (i.e. class attribute) is also given by an explicit type declaration. This type declaration is constraining, too. All objects that are declared to be values of a property must be instances of the declared type at the time of the assignment. Many ontology engineers favor a rather frame-like composite view of classes and their associated properties, too. Indeed, the advantage of using property domain and range descriptions to constrain the set of conforming RDF triples is a more succinct structuring of an ontology or schema. In Zhi#, the *checked*-keyword, which can be used as an operator or a statement, supports the frame-like notion of OWL object properties. The following example demonstrates the *checked*-operator on an OWL object property assignment expression. For an OWL object property *takesPlaceInAuditorium*, which relates *Lecture*s with *LargeRoom*s, an exception is thrown at runtime if the individuals referred to by e and l are not in the extensions of the named class descriptions *Lecture* and *LargeRoom*, respectively.

```
#Event e = [...];                                // Lecture ⊑ Event
#Location l = [...];                             // LargeRoom ⊑ Location
e.#takesPlaceInAuditorium = checked(l);          // ⇝ InvalidCastException
```

XSD and OWL were integrated with the Zhi# programming language similarly like generic types in Java. XSD data types and OWL DL class descriptions in Zhi# programs are subject to type substitution where references of external types are translated into 1) a constant set of proxy classes and 2) function calls on the Zhi# runtime library and its extensions for XSD and OWL. Detailed explanations how Zhi# programs are compiled into conventional C# are given in [17].

3 Validation

The Zhi# compiler was regression tested with 12 KLOC of Zhi# test code, which was inductively constructed based on the Zhi# language grammar, and 9 KLOC of typing information to regression test the semantic analysis of Zhi# programs.

The ease of use of ontological class descriptions and property declarations in Zhi# is illustrated in [17] by contradistinction with C#-based implementations of "ontological" behavior for .NET objects. The Zhi# approach facilitates the use of readily available ontology management systems compared to handcrafted reasoning code.

Examples for the advantage of *OWL aware compilation* in the Zhi# programming language over an API-based use of ontology management systems can be shown based on the following programming tasks, which are all frequent for ontology-based applications. Assume the following TBox.

$$A,\ B \sqsubseteq \neg C,\ \geq 1R \sqsubseteq A,\ \top \sqsubseteq \forall R.B,\ \top \sqsubseteq\ \ \leq 1R,\ \top \sqsubseteq \forall U.xsd\#positiveInteger$$

Task 1: Make an ontology available in a computer program.
Task 2: Create individual instances A, O, B, and C of classes $\top$, B, and C.
Task 3: Add individual O as a value for property R of individual A.
Task 4a): List the RDF types of individual O.
Task 4b): Check whether individual O is included by class description B.
Task 4c): List all individuals in the extension of class description B.
Task 5: Add individual C as a value for property R of individual A, which causes an inconsistent ABox since class descriptions B (range of property R) and C (which includes C) are disjoint.
Task 6: Add individual B as a value for property R of individual A and test if individuals O and B are equal (i.e. is there an inferred $sameAs(\text{O}, \text{B})$ statement in the ontology?).
Task 7: Add literals 23, −23, and string literal "NaN" as values for property U of individual O, where −23 and "NaN" are invalid values for the given TBox.

In line 1 in the Zhi# program shown below, the given TBox, which is defined in the *http://www.zhimantic.com/eval* namespace, is imported into the Zhi# compile unit. In line 2, XML Schema Definition's built-in data types are imported. In lines 4–9, ontological individuals are created. In line 9, the fully qualified name of individual C is inferred from the containing namespace of the named class description *C*. In line 10, the RDF triple [A *R* O] is declared in the ontology. Note how Zhi# facilitates the declaration of ad hoc relationships (instead of enforcing a frame-like view, where *Thing* A would not have a slot *R*). In line 11, a *foreach*-loop is used to iterate over all values of the auxiliary *Types* property of individual O. Note, in line 12, how ontological individuals are implicitly convertible to .NET *string*s. In line 15, the *is*-operator is used to dynamically check the RDF type of individual O. Be aware that the type check is performed on the ontology level. In line 16, a *foreach*-loop iterates over all values of the auxiliary *Individuals* property of the static class reference *B*. Note that the *Individuals* property is generic with respect to the static class reference on which it is invoked. The assignment statement in line 19 causes a compile-time error since individual C cannot simultaneously be in the extension of class C (its declared type) and B (the range restriction of property R) for consistent ontologies. In line 22, individuals O and B are compared for equality. Note that the equality operator (==) evaluates on the ontology level (i.e. $o == b$ evaluates to *true* since O and B were both used as values for functional property R). In lines 23–25, XSD data type variables are defined. In lines 26–28, property values are declared for datatype property U of individual O. Lines 27–28 cause compile-time errors since the XSD data types *xsd#integer* and *xsd#string* are not subsumed by the range restriction of datatype property U (i.e. *xsd#positiveInteger*).

```
 1  import OWL ont = http://www.zhimantic.com/eval;
 2  import XML xsd = http://www.w3.org/2001/XMLSchema;
 3  namespace N { class Program { static void Main () {
 4   #owl#Thing a =                                                    ↪
 5    new #owl#Thing("http://www.w3.org/2002/07/owl#A");
 6   #owl#Thing o =                                                    ↪
 7    new #owl#Thing("http://www.w3.org/2002/07/owl#O");
 8   #ont#B b = new #ont#B("http://www.zhimantic.com/eval#B");
 9   #ont#C c = new #ont#C("C");
10   a.#ont#R = o;
11   foreach (string T in o.Types) {
12    Console.WriteLine(T);
13   }
14   Console.WriteLine(o + " is" +                                     ↪
15                    (o is #ont#B ? "" : " not") + " a 'B'!");
16   foreach(#ont#B v in #ont#B.Individuals) {
17    Console.WriteLine(v);
18   }
19   a.#ont#R = c;   // Compile-time error in Zhi#!
20   a.#ont#R = b;
21   Console.WriteLine("Individuals 'o' and 'b' are" +                 ↪
22                    (o == b ? "" : " not") + " identical!");
23   #xsd#positiveInteger xpi = 23;
24   #xsd#integer xi = -23;
25   #xsd#string xs = "NaN";
26   o.#ont#U = xpi;
27   o.#ont#U = xi; // Compile-time error in Zhi#!
28   o.#ont#U = xs; // Compile-time error in Zhi#!
29  }}}
```

Java code using the Jena Semantic Web Framework for the same given programming tasks is available online[3]. It can be seen that the Zhi# code shown above treats the OWL terminology as first-class citizens of the program code, and is thus not only inherently less error-prone but can also be checked at compile time. Zhi#'s inherent support for ontologies facilitates type checking on the ontology level, which is completely unavailable if OWL APIs are used.

Finally, we mapped the CHIL OWL API [8,16,17] on auxiliary properties and methods of ontology entities in Zhi# programs. The functionality of 50 of the

[3] `http://sourceforge.net/p/zhisharp/wiki/Examples of Use/`

91 formally specified CHIL OWL API methods could be implemented as Zhi# programming language features, which facilitates the static checking of related method preconditions. Thus, more than half of the possible exceptions that may occur at runtime with API-based access could be eliminated.

The author leaves it to the reader to assess hybrid approaches that propose methodological means of integrating OWL models, which are managed by frameworks such as Protégé and Jena, with computer programs (see Puleston et al. [19] for an OWL-Java combination). Experience shows that integration shortcomings of hybrid approaches can barely be compensated by methodologies, which usually put the burden to behave compliantly to the ontology on the programmer.

4 Related Work

A major disadvantage of using an OWL API compared to, for example, Java-based domain models is the lack of type checking for ontological individuals. This lack of compile-time support has lead to the development of code generation tools such as the Ontology Bean Generator [18] for the Java Agent Development Framework [22], which generates proxy classes in order to represent elements of an ontology. Similarly, Kalyanpur et al. [9] devised an automatic mapping of particular elements of an OWL ontology to Java code. Although carefully engineered the main shortcomings of this implementation are the blown up Java class hierarchy and the lack of a concurrently accessible ontological knowledge base at runtime (i.e. the "knowledge base" is only available in one particular Java virtual machine in the form of instances of automatically generated Java classes). This separation of the ontology definition from the reasoning engine results in a lack of available ABox reasoning (e.g., type inference based on nominals). The two latter problems were circumvented by the RDFReactor approach [25] where a Java API for processing RDF data is automatically generated from an RDF schema. However, RDFReactor only provides a frame-like view of OWL ontologies whereas Zhi# allows for full-fledged ontological reasoning.

In stark contrast to these systems, the Zhi# programming language syntactically integrates OWL classes and properties with the C# programming language using conventional object-oriented notation. Also, Zhi# provides static type checking for atomic XSD data types, which may be the range of OWL datatype properties, while many ontology management systems – not to mention the above approaches – simply discard range restrictions of OWL datatype properties. A combination of static typing and dynamic checking is used for ontological class descriptions. In contrast to static type checking that is based on generated proxy classes, Zhi#'s OWL compiler plug-in adheres to disjoint class descriptions and copes well with multiple inheritance.

Koide and Takeda [11] implemented an OWL reasoner for the $\mathcal{FL}_0$ Description Logic on top of the Common Lisp Object System [5] by means of the Meta-Object Protocol [10]. Their implementation of the used structural subsumption algorithm [2] is described, however, to yield only incomplete results. The integration of OWL with the Python programming language was suggested by Vrandečić and implemented by Babik and Hluchy [3] who used metaclass-programming to embed OWL class and property descriptions with Python. Their approach, however, offers mainly a syntactic integration in form of LISP-like macros. Also, their prototypical implementation does not support namespaces and open world semantics.

The representation and the type checking of ontological individuals in Zhi# is similar to the type *Dynamic*, which was introduced by Abadi et al. [1]. Values of type *Dynamic* are pairs of a value v and a type tag T, where v has the type denoted by T. The result of evaluating the expression *dynamic e:T* is a pair of a value v and a type tag T, where v is the result of evaluating e. The expression *dynamic e:T* has type *Dynamic* if e has type T. Zhi#'s dynamic type checking of ontological individuals corresponds to the *typecase* construct as proposed by Abadi et al. in order to inspect the type tag of a given *Dynamic*. In Zhi# source programs, the use of OWL class names corresponds to explicit *dynamic* constructs. In compiled Zhi# code, invocations of the *AssertKindOf* method of the Zhi# runtime correspond to explicit *typecase* constructs.

Thatte described a "quasi-static" type system [23], where explicit *dynamic* and *typecase* constructs are replaced by implicit coercions and runtime checks. As in Thatte's work, Zhi#'s dynamic typing for OWL detects errors as early as possible to make it easy to find the programming error that led to the type error. Abadi et al. and Thatte's dynamic types were only embedded with a simple λ-calculus. The same is true for recent *gradual typing* proposals [20]. Tobin-Hochstadt and Felleisen developed the notion of *occurrence typing* and implemented a Typed Scheme [24]. Occurrence typing assigns distinct subtypes of a parameter to distinct occurrences, depending on the control flow of the program. Such distinctions are not made by Zhi#'s OWL compiler plug-in since it is hard to imagine that appropriate subtypes can be computed considering complex OWL class descriptions.

5 Conclusion

The Zhi# programming language makes the property-centric modeling features of the Web Ontology Language available via C#'s object-oriented notation (i.e. normal member access). The power of the "." can be used to declare ad hoc relationships between ontological individuals on a per instance basis. Zhi#'s *OWL aware compilation* integrates value space-based subtyping of XML Schema Definition and ontological classification with features of the programming language such as method overriding, user-defined operators, and runtime type checks. The Zhi# programming language is implemented by an extensible compiler framework, which is tailored to facilitate the integration of external classifier and reasoner components with the type checking of Zhi# programs. The compiler was written in C# 3.0 and integrated with the MSBuild build system for Microsoft Visual Studio. An Eclipse-based frontend was developed including an editor with syntax highlighting and autocompletion. The complete Zhi# tool suite totals 110 C# KLOC and 35 Java KLOC. Zhi# is available online[4].

[4] `http://zhisharp.sourceforge.net`

Zhi# offers a combination of static typing and dynamic checking for ontological class descriptions. Ontological reasoning directly influences the execution of programs: Zhi# programs don't just execute, they reason. Thus, the development of intelligent applications is facilitated. In contrast to many OWL APIs, Zhi# contains extensive support for XSD data types. Zhi# code that uses elements of an ontology is compiled into conventional C#. All functionality related to the use of ontologies is provided in a "pay-as-you-go" manner. The underlying ontology management system can be substituted in the Zhi# runtime library without recompilation of Zhi# programs.

Future work will include the transformation of Ontology Definition Metamodels [15] into Zhi# programs. With ontological class descriptions being first-class citizens the complete MOF [14] modeling space can be translated into the Zhi# programming language. We further plan to investigate the interplay of closed world semantics in an ontology with autoepistemic features (e.g., the epistemological $\mathcal{K}$-operator) with the static typing in Zhi#.

The Zhi# solution to provide programming language inherent support for ontologies is the first of its kind. Earlier attempts either lack ABox reasoning, concurrent access to a shared ontological knowledge base, or fall short in fully supporting OWL DL's modeling features. In recent years, numerous publications described the – apparently relevant – OWL-OO integration problem. However, the plethora of naïve code generation approaches and contrived hybrid methodologies all turned out to not solve the problem in its entirety. This work demonstrates that OWL DL ontologies can be natively integrated into a general-purpose programming language. The Zhi# compiler infrastructure has shown to be a viable approach to solving the OWL-OO integration problem.

References

1. Abadi, M., Cardelli, L., Pierce, B., Plotkin, G.: Dynamic typing in a statically-typed language. ACM Transactions on Programming Languages and Systems (TOPLAS) 13(2), 237–268 (1991)
2. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
3. Babik, M., Hluchy, L.: Deep integration of Python with Web Ontology Language. In: Bizer, C., Auer, S., Miller, L. (eds.) 2nd Workshop on Scripting for the Semantic Web (SFSW) CEUR Workshop Proceedings, vol. 183 (June 2006)
4. Biron, P.V., Malhotra, A.: XML Schema Part 2: Datatypes Second Edition. Technical report, World Wide Web Consortium (W3C) (October 2004)
5. Demichiel, L.G., Gabriel, R.P.: The Common Lisp Object System: An Overview. In: Bézivin, J., Hullot, J.-M., Lieberman, H., Cointe, P. (eds.) ECOOP 1987. LNCS, vol. 276, pp. 151–170. Springer, Heidelberg (1987)
6. Hejlsberg, A., Wiltamuth, S., Golde, P.: C# language specification version 1.0. Technical report, ECMA International (2002)
7. Labs, H.P.: Jena Semantic Web Framework (2004)
8. Information Society Technology integrated project 506909. Computers in the Human Interaction Loop, CHIL (2004)

9. Kalyanpur, A., Pastor, D.J., Battle, S., Padget, J.A.: Automatic mapping of OWL ontologies into Java. In: Maurer, F., Ruhe, G. (eds.) 16th International Conference on Software Engineering and Knowledge Engineering (SEKE), pp. 98–103 (June 2004)
10. Kiczales, G., de Rivières, J., Bobrow, D.G.: The Art of the Metaobject Protocol. MIT Press, Cambridge (1991)
11. Koide, S., Takeda, H.: OWL Full reasoning from an object-oriented perspective. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 263–277. Springer, Heidelberg (2006)
12. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C) (February 2004)
13. Meijer, E., Drayton, P.: Static typing where possible, dynamic typing when needed. In: OOPSLA Workshop on Revival of Dynamic Languages (2004)
14. Object Management Group (OMG). MetaObject Facility (August 2005)
15. Object Management Group (OMG). Ontology Definition Metamodel (2005)
16. Paar, A.: Zhi# – programming language inherent support for ontologies. In: Favre, J.-M., Gasevic, D., Lämmel, R., Winter, A. (eds.) ateM 2007: Proceedings of the 4th International Workshop on Software Language Engineering, Mainzer Informatik-Berichte,Mainz, Germany, pp. 165–181. Johannes Gutenberg Universität Mainz, Nashville (2007)
17. Paar, A.: Zhi# – Programming Language Inherent Support for Ontologies. PhD thesis, Universität Karlsruhe (TH), Am Fasanengarten 5, 76137 Karlsruhe, Germany (July 2009), http://digbib.ubka.uni-karlsruhe.de/volltexte/1000019039
18. Protégé Wiki: Ontology Bean Generator (2007)
19. Puleston, C., Parsia, B., Cunningham, J., Rector, A.L.: Integrating object-oriented and ontological representations: A case study in Java and OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 130–145. Springer, Heidelberg (2008)
20. Siek, J.G., Taha, W.: Gradual typing for functional languages. In: Bailey, M.W. (ed.) 7th Workshop on Scheme and Functional Programming (Scheme) ACM SIGPLAN Notices, vol. 41. ACM Press, New York (2006)
21. Stanford University School of Medicine. Protégé knowledge acquisition system (2006)
22. Telecom Italia. Java Agent Development Framework, JADE (2007)
23. Thatte, S.R.: Quasi-static typing. In: 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), pp. 367–381. ACM Press, New York (1990)
24. Tobin-Hochstadt, S., Felleisen, M.: The design and implementation of typed Scheme. ACM SIGPLAN Notices 43(1), 395–406 (2008)
25. Völkel, M.: RDFReactor – from ontologies to programmatic data access. In: 1st Jena User Conference (JUC). HP Bristol (May 2006)

Lightweight Semantic Annotation of Geospatial RESTful Services

Víctor Saquicela, Luis M. Vilches-Blazquez, and Oscar Corcho

Ontology Engineering Group, Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{vsaquicela,lmvilches,ocorcho}@fi.upm.es

Abstract. RESTful services are increasingly gaining traction over WS-* ones. As with WS-* services, their semantic annotation can provide benefits in tasks related to their discovery, composition and mediation. In this paper we present an approach to automate the semantic annotation of RESTful services using a cross-domain ontology like DBpedia, domain ontologies like GeoNames, and additional external resources (suggestion and synonym services). We also present a preliminary evaluation in the geospatial domain that proves the feasibility of our approach in a domain where RESTful services are increasingly appearing and highlights that it is possible to carry out this semantic annotation with satisfactory results.

Keywords: REST, semantic annotation, geospatial RESTful services.

1 Introduction

In recent years, since the advent of Web 2.0 applications and given some of the limitations of "classical" Web services based on SOAP and WSDL, Representational State Transfer (REST) services have become an increasing phenomenon. Machine-oriented Web applications and APIs that are conformant to the REST architectural style [23], normally referred to as RESTful Web services, have started appearing mainly due to their relative simplicity and their natural suitability for the Web.

However, using RESTful services still requires much human intervention since the majority of their descriptions are given in the form of unstructured text in a Web page (HTML), which contains a list of the available operations, their URIs and parameters (also called attributes), expected output, error messages, and a set of examples of their execution. This hampers the automatic discovery, interpretation and invocation of these services, which may be required in the development of applications, without extensive user involvement.

Traditionally, semantic annotation approaches for services have focused on defining formalisms to describe them, and have been normally applied to WS-* service description formalisms and middleware. More recently, these (usually heavyweight) approaches have started to be adapted into a more lightweight manner for the semantic description of RESTful services [1, 5, 8]. However, most of the processes related to the annotation of RESTful services (e.g., [2, 11]) still require a large amount of human intervention. First, humans have to understand the informal descriptions provided in the RESTful service description pages, and then the semantic annotation of RESTful services is done manually, with or without assistance.

In this paper, we address the challenge of automating the semantic annotation of RESTful services by: (1) obtaining and formalising their syntactic descriptions, which allows their registration and invocation, and (2) interpreting, and semantically enriching their parameters.

The main contribution of our work is the partial automation of the process of RESTful semantic annotation services using diverse types of resources: a cross-domain ontology, DBpedia (combined with GeoNames in the specific case of geospatial services), and diverse external services, such as suggestion and synonym services.

The remainder of this paper is structured as follows: Section 2 presents related work in the context of semantic annotation of WS-* and RESTful services. Section 3 introduces our approach for automating the annotation of RESTful services, including explanations on how we derive their syntactic description and semantic annotation. Section 4 presents the evaluation of our system in the context of these services from the geospatial domain. Finally, Section 5 presents some conclusions of this paper and identifies future lines of work.

2 Related Work

Most research in the semantic annotation of RESTful services has focused on the definition of formal description languages for creating semantic annotations. The main proposed formalisms for describing these services are: the Web Application Description Language[1] (WADL), which describes syntactically RESTful services and the resources that they access; its semantic annotation extension [19]; MicroWSMO [3], which uses hREST (HTML for RESTful services) [3, 5]; and SA-REST [2, 8], which uses SAWSDL [1] and RDFa[2] to describe service properties.

From a broader point of view, the work done in the state of the art on Semantic Web Services (SWS) has mainly focused on WS-*services. OWL-S and WSMO are approaches that use ontologies to describe services.

Some authors propose the adaptation of heavyweight WS-* approaches to describe RESTful services. An example is proposed in [10], which makes use of OWL-S as the base ontology for services, whereas WADL is used for syntactically describing them. Then, the HTTP protocol is used for transferring messages, defining the action to be executed, and also defining the execution scope. Finally, URI identifiers are responsible for specifying the service interface.

Other approaches are more lightweight (e.g., [1, 2]). The authors advocate an integrated lightweight approach for describing semantically RESTful services. This approach is based on use of the hREST and MicroWSMO microformats to facilitate the annotation process. The SWEET tool [2] supports users in creating semantic descriptions of RESTful services based on the aforementioned technologies. Unlike

[1] http://www.w3.org/Submission/wadl/

[2] http://www.w3.org/TR/xhtml-rdfa-primer/

this work, our approach is focused on automating this process, and could be well integrated into this tool. Once the semantics of the RESTful service is obtained, this could be represented in any of the existing semantic description approaches, such as hREST, MicroWSMO, etc.

Finally, another approach for service description that focuses on automation, and hence can be considered closer to our work, is presented in [17]. This approach classifies service parameter datatypes using HTML treated Web form files as the Web service's parameters using Naïve Bayes.

3 An Approach for the Automatic Semantic Annotation of RESTful Services

In this section, we present our approach, visualized in Figure 1, for automating the syntactic and semantic annotation of RESTful services. Our system consists of three main components, including invocation and registration, repository, and semantic annotation components, which are enriched by diverse external resources. Next, we briefly describe the different components, illustrating the descriptions with some sample services on the geospatial domain.

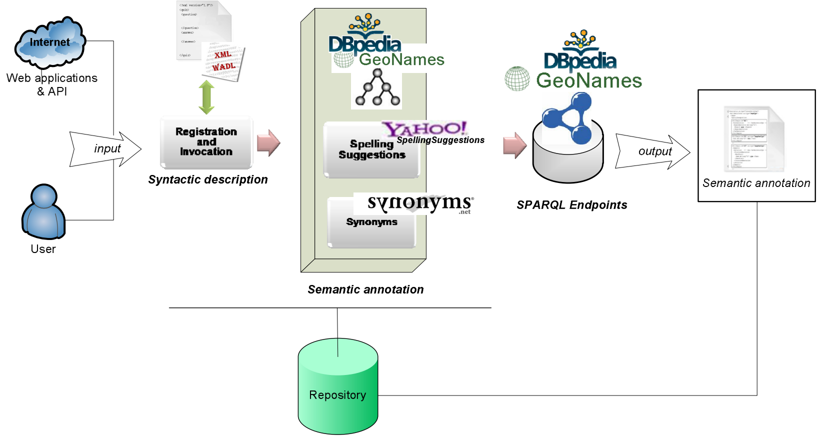


Fig. 1. RESTful Service Semantic Annotation System

3.1 A Sample Set of RESTful Services in the Geospatial Domain

Nowadays the largest online repository of information about Web 2.0 mashups and APIs is ProgrammableWeb.com. This aggregator site provides information on 5,401 mashups and 2,390 APIs that were registered between September 2005 and

November 2010. Mashups tagged as “mapping” represent a 44.5% mashups (2,403 mashups) of the listed ones, what represents the importance of geospatial information in the generation of these applications. With respect to APIs, GoogleMaps is the most used with an 89.4%, that is, this API is used on 2,136 mashups. These data show the importance of geospatial information in the context of the REST world. The following services, taken from the aforementioned site, are two representative geospatial RESTful services:

- **Service 1.** http://ws.geonames.org/countryInfo?country=ES
This service retrieves information related to a ‘country’. More specifically, it returns information about the following parameters: ‘capital’, ‘population’, ‘area’ (km^2), and ‘bounding box of mainland’ (excluding offshore islands). In the specified URL, we retrieve information about Spain.

- **Service 2.** http://api.eventful.com/rest/venues/search?app_key=p4t8BFcLDtCzpxdS&location=Madrid
This service retrieves information about places (venues). More specifically, it returns parameters like: ‘city’, ‘venue_name’, ‘region_name’, ‘country_name’, ‘latitude’, ‘longitude’, etc. In the specified URL, we retrieve information about Madrid.

3.2 Syntactic Description Storing: Invocation and Registration Details into a Repository

As aforementioned, RESTful services are normally described or registered in sites like *programmableWeb* by means of their URLs, plus some natural language descriptions, tags, and execution examples, if at all available. Hence, the first step in our system is to take as input the URL of an available RESTful service that is known by users (for instance, it has been discovered by a user by browsing this site, or it has been sent to the user by a friend).

In our system, the user adds the URLs of a service as a starting point, with the objective of obtaining automatically information related to it. Once the URLs is added, our system invokes the RESTful service with some sample parameters, obtained from the examples that are normally provided together with the URL (if this information is not available, our system cannot continue automatically without further human intervention), and analyzes the response to obtain a basic syntactic description of the parameter set, used as inputs and outputs.

In this process our system uses the Service Data Object[3] (SDO) API to perform the invocation of the RESTful service and determine whether it is available or not. SDO is a specification for a programming model that unifies data programming across data source types and provides robust support for common application patterns in a disconnected way [22]. The invocation process is performed as follows: first, it takes the input parameters and their values, which are given to the service as part of a URL. Then, the system invokes the service that translates our "RESTful service call" into a query to a specific service, including the URL and related parameters.

[3] http://www.oasis-opencsa.org/sdo

The service invocation of a specific RESTful service may return diverse formats, such as JSON, XML, etc. In our work we use any of these formats, although for presentation purposes in this paper we will show how we handle XML responses. The results of a sample invocation of the services that we presented in section 3.1 are showed in Table 1.

Table 1. XML response of two sample RESTful services

Service 1	Service 2
	<venue id="V0-001-000154997-6">
<geonames>	<url>http://eventful.com/madrid/venues/la-
<country>	ancha-/V0-001-000154997-6</url>
<countryCode>ES</countryCode>	**<country_name>Spain</country_name>**
<countryName>Spain</countryName>	<name>La Ancha</name>
<isoNumeric>724</isoNumeric>	<venue_name>La Ancha</venue_name>
<isoAlpha3>ESP</isoAlpha3>	<description></description>
<fipsCode>SP</fipsCode>	<venue_type>Restaurant</venue_type>
<continent>EU</continent>	<address></address>
<capital>Madrid</capital>	<city_name>Madrid</city_name>
<areaInSqKm>504782.0</areaInSqKm>	<region_name></region_name>
<population>40491000</population>	<region_abbr></region_abbr>
<currencyCode>EUR</currencyCode>	<postal_code></postal_code>
<languages>es-ES,ca,gl,eu</languages>	<country_abbr2>ES</country_abbr2>
<geonameId>2510769</geonameId>	<country_abbr>ESP</country_abbr>
<bBoxWest>-18.169641494751</bBoxWest	<longitude>-3.68333</longitude>
<bBoxNorth>43.791725</bBoxNorth>	<latitude>40.4</latitude>
<bBoxEast>4.3153896</bBoxEast>	<geocode_type>City Based GeoCodes
<bBoxSouth>27.6388</bBoxSouth>	</geocode_type>
</country>	<owner>frankg</owner>
</geonames>	<timezone></timezone>
	<created></created>
	<event_count>0</event_count>
	<trackback_count>0</trackback_count>
	<comment_count>0</comment_count>
	<link_count>0</link_count>
	<image></image>
	</venue>
	<venue id="V0-001-000154998-5">

These XML responses are processed using SDO, which enables to navigate through the XML and extract output parameters of each service[4]. The result of this invocation process is a syntactic definition of the RESTful service in XML, which can be expressed in description languages like WADL or stored into a relational model. Table 2 shows the different output parameters of each service, where we can observe by manual inspection that there is some similarity between diverse parameters (e.g., `countryName` and `country_name`) and that they return similar values (Spain). However, these parameters are written differently. These differences between parameters are described and dealt with in sections 3.3.1 and 3.3.2.

With URL and input/output parameters, we generate a WADL description that can be used as the input to the next process. Additionally, we register and store this description into a repository using an oriented-object model. This repository is implemented as a database that is specifically designed to store syntactic descriptions of RESTful services and parameters' values of invocations. We selected this storage model in order to increase efficiency in the recovery of the RESTful services.

[4] In the work reported here, we considered only XML tags with values.

Once the RESTful service is registered and the WADL description is generated, our system invokes the service without associated parameters. For example:

***Service 1'**. http://ws.geonames.org/countryInfo?
*This is an example of invocation (Service 1) without its associated parameters.

On the other hand, our system also considers service URLs as http://www.foo.org/weather/Madrid. These services belong to a specific RESTful entity and they are always invoked with its associated parameters.

In this way, the system invokes the service for retrieving a collection of instances (countries)[5] related to the service. The results of this invocation are stored into the oriented-object model. Thus, this process allows collecting additional information about a service (output parameters and instances), which is registered in our system, and retrieving it for future processes without the need to invoke the original service.

Table 2. Syntactic description of our two sample RESTful services

Service 1:
countryInfo($country,bBoxSouth,isoNumeric,continent,fipsCode,areaInSqKm,languages,isoAlpha3,countryCode,bBoxNorth,population,bBoxWest,currencyCode,bBoxEast,capital,geonameId,countryName)
Service 2:
rest/venues/search($location,$app_key,id,link_count,page_count,longitude,trackback_count,version,venue_type,owner,url,country_name,event_count,total_items,city_name,address,name,latitude,page_number,postal_code,country_abbr,first_item,page_items,last_item,page_size,country_abbr2,comment_count,geocode_type,search_time,venue_name)

3.3 Semantic Annotation

Some of the difficulties that arise in the semantic annotation of RESTful services are briefly described in [1, 11]. In order to cope with them, we rely on techniques and processes that permit: a) semantic annotation using only the syntactic description of the services and their input/output parameters, or b) semantic annotation by identifying a set of example values that allow the automatic invocation of the service.

The starting point of the semantic annotation process is the list of syntactic parameters obtained previously (a WADL file or the model stored into a relational database). Once the RESTful service is syntactically described with all its identified input and output parameters, we proceed into its semantic annotation. We follow a heuristic approach that combines a number of external services and semantic resources to propose annotations for the parameters as shown in Figure 2. Next, we describe the main components of the semantic annotation.

[5] The results of this service invocation are available at http://delicias.dia.fi.upm.es/RESTfulAnnotationWeb/RESTService1/RESTservice1.xml

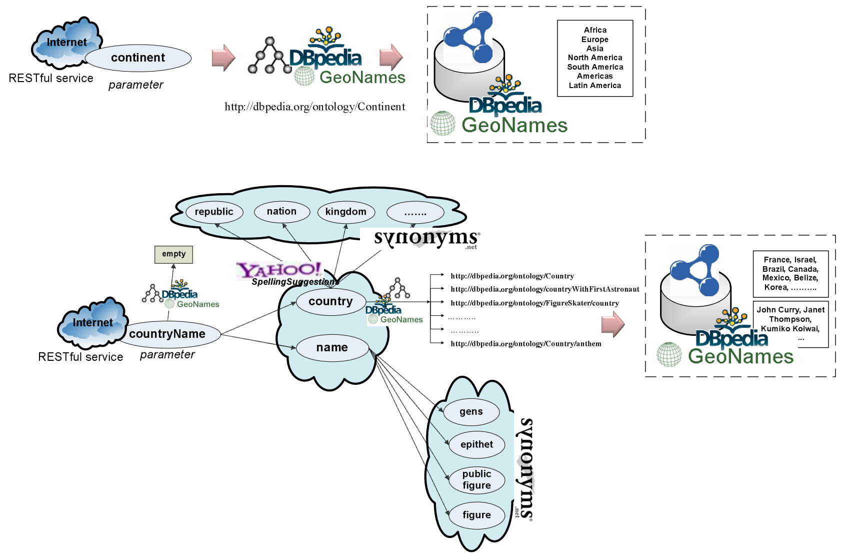

Internet
RESTful service
continent
parameter
DBpedia
GeoNames
http://dbpedia.org/ontology/Continent
Africa
Europe
Asia
North America
South America
Americas
Latin America
republic
nation
kingdom
....
synonyms.net
YAHOO! SpellingSuggestion
empty
country
name
countryName
http://dbpedia.org/ontology/Country
http://dbpedia.org/ontology/countryWithFirstAstronaut
http://dbpedia.org/ontology/FigureSkater/country
http://dbpedia.org/ontology/Country/anthem
France, Israel, Brazil, Canada, Mexico, Belize, Korea, ...
John Curry, Janet Thompson, Kumiko Koiwai, ...
gens
epithet
public figure
figure

Fig. 2. Semantic annotation process

3.3.1 A Model for Describing RESTful Services

In order to describe semantically these services we define a model to represent the relationships of the different service parameters with the diverse resources used for semantic annotation. Some of the elements of this model are domain-independent, while others are domain dependent (in our examples we use these related to the geospatial domain, where we have performed our experiments in order to evaluate the feasibility of our approach). With respect to the domain-independent component, we use DBpedia, a community driven knowledge base, as the main source of background knowledge for supporting the semantic annotation process. This is complemented by the domain-dependent component. In the context of the shown examples, we use GeoNames[6] as a source related to geospatial information. This model (Figure 3) defines the following components:

- `Parameter`. This class provides a list of all parameters (inputs and outputs) collected from different services. Likewise, we search for additional information for each parameter, such as suggestions and synonyms, for enriching the initial description of parameters. The relation `hasCollection` relates `Parameter` with `DBpediaOntology`. Every parameter can be related to any number of DBpedia classes or properties (from 0 to N).
- `Ontologies`. This class contains classes and properties of the DBpedia and GeoNames ontology related to the parameters of each service. This class is related to the classes `DBpediaInstance` and `GeonamesInstance` by

[6] http://www.geonames.org/

means of the relation `hasCollection`. `Ontologies` can be related to any number of DBpedia or GeoNames instances (from 0 to N).
- `DBpediaInstance`. This class collects values from the DBpedia SPARQL Endpoint, where a parameter may have one o more associated resources.
- `GeonamesInstance`. This class collects geospatial information related to `latitude`, `longitude`, and `bounding box parameters` from a GeoNames SPARQL Endpoint.

The information related to each parameter of the RESTful service (semantic annotations) is stored only once in the system repository. By doing this, we avoid to duplicate information related to the same parameters, hence storing annotations independently of services and increasing the efficiency of our system.

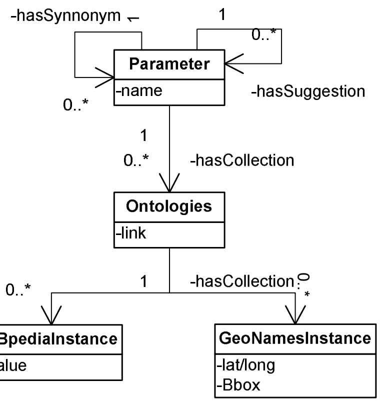


Fig. 3. Model for the description of geospatial RESTful service parameters

3.3.2 Using Semantic Sources in the Annotation Process

At this stage, the list of syntactic parameters obtained previously is used to query the DBpedia and GeoNames SPARQL Endpoints (the latter is only used in the case of the geospatial domain) and retrieve associated results for each parameter, as follows:

- First, the system retrieves all the classes from the DBpedia ontology whose names have a match with each parameter of the RESTful service. In this matching process we test two different techniques:

  - On the one hand, our approach uses an exact match to compare parameters of RESTful service with the labels of the ontologies' classes and properties.
  - On the other hand, our approach uses a combination of various similarity metrics (Jaro, JaroWinkler and Levenshtein metrics)[7] to compare parameters with the labels of the elements of these ontologies. This proposal allows

[7] http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html

matching between strings such as `countryName`, `country_name`, or `country`, for example.

If the system obtains correspondences from the matching process, it uses these DBpedia concepts individually to retrieve samples (concept instances) from the DBpedia SPARQL Endpoint. Likewise, when a parameter matches an ontology class related to some geospatial information; such as latitude, longitude, or bounding box, our system retrieves samples from the GeoNames SPARQL Endpoint. The resulting information (RDF) is suggested automatically to the system and registered as a possible value for the corresponding parameter. When a parameter matches more than once in the DBpedia ontology, our system only considers those concepts that have information (instances), and automatically discards those ontology concepts without instances.

- Next, the system tries to find correspondences between parameters of the RESTful service and ontology properties. If the system obtains some correspondences, it uses these DBpedia properties individually to retrieve information of the DBpedia or GeoNames SPARQL Endpoint, as described above. Furthermore, this information is registered as a possible correct value for the corresponding parameter.
- Finally, with the obtained classes and properties, the system calls the DBpedia and GeoNames SPARQL Endpoints to retrieve values (instances) for those classes and properties, so that now we have possible values for them.

3.3.3 Enriching the Semantic Annotations

Our system looks for matches with DBpedia (and GeoNames) classes and properties. Hence it is possible to have parameters with not correspondences identified, since there are many lexical and syntactic variations that the parameter names may have, and because in some cases the information that is being requested may not be available in any of the external sources that are consulted. In order to annotate semantically the parameters that did not match any DBpedia resource, we use additional external services to enrich the results. Below we describe the main characteristics of the external services that we consider.

Spelling Suggestion Services

Web search engines (e.g. Google, Yahoo, and Microsoft) usually try to detect and solve users' writing mistakes. Spelling Suggestion services, also called "Did You Mean", are algorithms which aim at solving these spelling mistakes. For example, when a user writes 'countryName' these algorithms suggest 'country' and 'name' separately.

In our system we use the Yahoo Boss service[8] to retrieve suggestions about the parameters that we have obtained in the previous steps and for which we have not obtained any candidate in our semantic resources. Thus, for each parameter that the system did not find a correspondence with classes or properties in DBpedia (nor GeoNames), this service is invoked to obtain a list of suggestions to query DBpedia

[8] http://developer.yahoo.com/search/boss/boss_guide/Spelling_Suggest.html

(and GeoNames) again. The output is registered and stored into the repository. Following the previous example, the parameter 'countryName' is not found in the DBpedia ontology. Nevertheless, the added service allows separating this parameter in 'country' and 'name', and then it calls to the DBpedia SPARQL Endpoint with these new strings for obtaining results.

Synonym services

This external service[9] is incorporated into the system to retrieve possible synonyms for a certain parameter. This service tries to improve the semantic annotation process when our system does not offer results for the previous steps, that is, when we still have parameters in a RESTful service without any potential annotations.

As an example, we may have a parameter called 'address'. The invocation process uses the synonyms service to retrieve a set of synonyms of 'address' such as *extension*, *reference*, *mention*, *citation*, *denotation*, *destination*, *source*, *cite*, *acknowledgment*, and so on. These outputs are registered and stored into the repository, and then, the service calls to the DBpedia (and GeoNames) SPARQL Endpoints for results again.

Both spelling suggestion and synonym services use the matching process described in section 3.3.1 to find possible matches between the output of these services and the components of the used ontologies.

3.4 Checking the Semantic Annotation of RESTful Services

In order to check the collected sample individuals and the initial semantic annotations obtained as a result of the previous process, our system invokes the RESTful services that were already registered in the repository (as we describe in Section 3.2) and validates the input and output parameters for checking which is the best option to describe each parameter.

For the validation of the **input parameters**, our system selects, for each parameter, a random subset of the example instances (of classes and/or properties) coming from the DBpedia (and GeoNames) ontology that we have obtained and registered before. Next, it makes several invocations of the RESTful service iterating over these registered values. The system does not check this with all the possible combination of collected instances for all parameters for two reasons: first, because of the combinatorial explosion that may be produced in such a case, and second because many RESTful services have invocation limitations.

When a service has one or more than one input parameter, the system obtains randomly some instances of this parameter for the validation process. Each parameter generates a collection (list) of instances from our repository. Then, the system joins instances to obtain a table of all combinations of each parameter. Likewise, the geospatial parameters, specifically latitude and longitude parameters, are combined to obtain some values (instances) that can be used for this invocation.

If the service returns results from the invocation, then the service is considered as executable, and the corresponding annotations are marked as valid. If a service cannot be invoked successfully, the service is classified as non-executable and is automatically discarded from the list of services that can be automatically annotated.

[9] http://www.synonyms.net/

For the validation of the **output parameters**, our system only takes into account executions with the correct inputs from the input sets that have been considered before. Next, the system compares the outputs obtained after execution with the information already stored in the repository due to the initial retrieval processes done before with DBPedia (and GeoNames), and external utility services. If the output can be matched, our system considers the output annotation as valid.

Finally, the correspondences that have been established between the different parameters of the RESTful service and the DBpedia (and GeoNames) ontology are registered and stored in the repository, so that they can be used later. In such a way, the RESTful service is annotated semantically and it will allow generating semantic descriptions or annotations of any of the types that were identified in the related work section (WADL, hREST, etc.). Table 3 provides an abbreviated form of this description for our exemplar service 1.

Table 3. Semantic annotation of a RESTful service

```
($country,http://www.w3.org/2003/01/geo/wgs84_pos#lat,http://
w ww.w3.org/2003/01/geo/wgs84_pos#long,isoNumeric,http://dbpedia
.org/ontology/Continent,fipsCode,http://dbpedia.org/property/
areaMetroKm,languages,isoAlpha3,http://dbpedia.org/ontology/cou
ntry,http://www.w3.org/2003/01/geo/wgs84_pos#lat,http://www.w3
.org/2003/01/geo/wgs84_pos#long,http://dbpedia.org/ontology/
populationDensity,http://www.w3.org/2003/01/geo/wgs84_pos#lat,ht
tp://www.w3.org/2003/01/geo/wgs84_pos#long,http://dbpedia.org/
ontology/Currency,http://www.w3.org/2003/01/geo/wgs84_pos#lat,
http://www.w3.org/2003/01/geo/wgs84_pos#long,http://dbpedia.org
/ontology/capitalgeonameId,http://dbpedia.org/ontology/country)
```

4 Experimental Results

In order to evaluate our approach in the geospatial domain we have used 60 different RESTful services found in *http://www.programmableweb.com/*, which we have selected randomly from those that were available and could be characterized to contain geospatial information by a manual lookup. The list of these services can be found in our experiment website[10]. In the syntactic registration of all these services in the system, by means of introducing the list of their URLs, our system successfully registered 56 of them into the repository (4 services could not be registered due to an invocation error). As a result of this syntactic registration, the system has produced a complete list of 369 different parameters (52 input parameters and 342 output parameters), without duplications.

This analysis follows the three steps described in our semantic annotation process. First, our system identifies correctly 191 of 369 parameters by calling directly the DBpedia and GeoNames ontologies. Second, the system uses initial parameters plus the suggestion service and calls the DBpedia and GeoNames ontologies. In this case,

[10] http://delicias.dia.fi.upm.es/RESTfulAnnotationWeb/SourcesList/sources.ods

it identifies 33 correspondences and adds 57 parameters to the initial ones. Third, the system uses the initial parameters plus the synonyms service, and calls the DBpedia and GeoNames ontologies. It identifies 126 correspondences and incorporates 1,147 additional parameters into the system. Finally, the system combines all the resources that result from the enrichment process and calls again the DBpedia and GeoNames SPARQL endpoint. Here it identifies 159 correspondences and adds 1,573 more parameters. A detailed view of these results is shown in Table 4.

Table 4. Enriching initial parameters with external resources

Attributes	**Total**	**Additional parameters**	**Matches** (*DBpedia and GeoNames ontologies*)
Initial parameters	369	-	191
Parameters + Suggestions	426	57	33
Parameters + Synonyms	1573	1147	126
Parameters + Suggestions + Synonyms	1573	1204	159

With respect to the validation of input parameters[11] (see Table 5), our system recognizes 152 inputs of the initial list, of which 76 parameters can be annotated automatically with the DBpedia (33 parameters) and GeoNames (45 parameters) ontologies.

Likewise, we have discovered with our evaluation that some other parameters are useless in terms of semantic annotation processes, since they refer to the navigation process through the RESTful service results or "special" parameters. These parameters (input/output) are not considered for this validation (nevertheless, they are considered to the invocation process), concretely 155 "special" parameters, for instance, `userID`, `api_key`, `page`, `total`, `hits`, etc.). These parameters were detected manually and a list of them is collected in this website[12]. Our system takes them out automatically from the service registration process[13].

One aspect of our system is that we cannot always guarantee a successful annotation, because in some cases the system cannot find any correspondence between the service parameters and the concepts or properties of the DBpedia or GeoNames ontologies. This is common, for instance, when parameter names are described by only one letter (e.g., `s`, `l` or `q`) and hence they are not sufficiently descriptive for our automated approach to find any correspondence. In our evaluation, we had 12 of this type of parameters. In these cases the parameters should be shown to users for a manual description of them.

In summary, for 56 of the 60 initial geospatial RESTful services we have obtained correct input parameter associations, except for 4 cases where we could not find any correspondence.

[11] A detailed analysis on these input parameters is available at http://delicias.dia.fi.upm.es/RESTfulAnnotationWeb/inputs/inputs.ods

[12] http://delicias.dia.fi.upm.es/RESTfulAnnotationWeb/parameters/Parameters.ods

[13] This was not described in the process described in section 3 since we did not consider it relevant for the description of the whole process.

Table 5. Results of the input and output paremeters

RESTful Service	Total parameters	Annotated parameters	Annotated parameters (DBpedia)	Annotated parameters (GeoNames)	Special parameters	Service validation
Input parameters	152	76	33	45	73	56✓ 4✗
Output parameters	862	315	202	113	299	-

With respect to the validation of output parameters[14] (see Table 5), our system recognizes 862 outputs that belong to the 56 services whose input parameters have been validated. This total of output parameters is divided into 315 whose correspondences can be found using DBpedia (202 parameters) and GeoNames (113 parameters) ontologies, and 391 (special (299) and not found (92) parameters) whose correspondences cannot be found.

Table 6. Output parameters metrics

RESTful Service	Found parameters	Not found parameters	Annotated	Not annotated	Right parameters	Precision	Recall
Output parameters	475	92	315	160	242	0.66	0.77

While in the context of the input parameters we are interested in determining whether we can call the service or not, in the case of output parameters, we are interested in the precision and recall metrics of the annotation process. Hence, we have generated a gold standard with the studied services in order to assign manually the annotations that have to be produced for all output parameters of these services, and we have performed an evaluation of the results obtained from the system for the parameters that are found. Regarding the parameters that are found, our system annotates 315 of them automatically, from which 242 parameters are annotated correctly according to the gold standard, while 160 parameters are not annotated. This provides us with an average value for precision equal to 0.66 and recall equal to 0.77 for both metrics.

To the best of our knowledge, there are no available results from existing research works to compare our results against. Likewise, these preliminary results prove the feasibility of our system and highlight that its possible to carry out an assisted semantic annotation of RESTful services.

5 Conclusions and Future Work

In this paper we have proposed an approach to perform an assisted semantic annotation process of RESTful services. This process is implemented in a system that

[14] A detailed analysis on these output parameters is available at http://delicias.dia.fi.upm.es/ RESTfulAnnotationWeb/ouputs/outputs.ods

takes into account the DBpedia ontology and its SPARQL Endpoint, for general annotation, and GeoNames and its SPARQL Endpoint for geospatial specific results, as well as different external resources such as synonyms and suggestion services. We use combinations of these resources to discover meanings for each of the parameters of the RESTful services that a user may select and perform semantic annotations of them.

To illustrate our work and guide the explanations of the proposed semantic annotation process we have used two exemplary RESTful services related to the geospatial domain. Besides, we have presented some preliminary experimental results that prove the feasibility of our approach, at least in the geospatial domain, and show that it is possible to assist the semantic annotation of RESTful services, again at least in this domain.

Future work will focus on the development of a GUI that will ease the introduction of existing services by users for their semantic annotation, probably incorporated in any existing RESTful semantic annotation tool/utility suite. Furthermore, we also plan to make improvements to the proposed system through the analysis of instances retrieved in the matching process, so as to improve the results that have been demonstrated in our evaluation. In the same sense, we also aim at improving the SPARQL queries to DBpedia and other semantic resources associated or not to a specific domain, to better explore this resource in the annotation process, and optimize the use of suggestion and synonyms services. Finally, we will incorporate more specific domain ontologies in the semantic process for taking advantage of specific domain characteristics.

Acknowledgments

This work has been supported by the R&D project España Virtual (CENIT2008-1030), funded by Centro Nacional de Información Geográfica and CDTI under the R&D programme Ingenio 2010.

References

1. Maleshkova, M., Kopecky, J., Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services. In: Workshop: Beyond SAWSDL at OnTheMove Federated Conferences & Workshops, Vilamoura, Portugal (2009)
2. Maleshkova, M., Pedrinaci, C., Domingue, J.: Semantically Annotating RESTful Services with SWEET, Demo at 8th ISWC, Washington D.C., USA (2009)
3. Maleshkova, M., Gridinoc, L., Pedrinaci, C., Domingue, J.: Supporting the Semi-Automatic Acquisition of Semantic RESTful Service Descriptions. In: ESWC (2009)
4. Pedrinaci, C., Domingue, J., Krummenacher, R.: Linked Data Meets Artificial Intelligence. In: Services and the Web of Data: An Unexploited Symbiosis, Workshop: Linked AI: AAAI Spring Symposium (2010)
5. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML Microformat for Describing RESTful Web Services. Web Intelligence, 619–625 (2008)
6. Lambert, D., Domingue, J.: Grounding semantic web services with rules. In: Workshop: Semantic Web Applications and Perspectives, Rome, Italy (2008)

7. Steinmetz, N., Lausen, H., Brunner, M.: Web Service Search on Large Scale. ICSOC/ServiceWave, 437–444 (2009)
8. Lathem, J., Gomadam, K., Sheth, A.P.: SA-REST and (S)mashups: Adding Semantics to RESTful Services. In: ICSC 2007, pp. 469–476 (2007)
9. García Rodríguez, M., Álvarez, J.M., Berrueta, D., Polo, L.: Declarative Data Grounding Using a Mapping Language. Communications of SIWN 6, 132–138 (2009)
10. Freitas Ferreira Filho, O., Grigas Varella Ferreira, M.A.: Semantic Web Services: A RESTful Approach. In: IADIS Int. Conference WWW/INTERNET 2009, Rome, Italy (2009)
11. Alowisheq, A., Millard, D.E., Tiropanis, T.: EXPRESS: EXPressing REstful Semantic Services Using Domain Ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 941–948. Springer, Heidelberg (2009)
12. Alarcon, R., Wilde, E.: Linking Data from RESTful Services. In: LDOW 2010, Raleigh, North Carolina (2010)
13. Lerman, K., Plangprasopchok, A., Knoblock, C.A.: Semantic Labeling of Online Information Sources. Int. J. Semantic Web Inf. Syst. 3(3), 36–56 (2007)
14. Ambite, J.L., Darbha, S., Goel, A., Knoblock, C.A., Lerman, K., Parundekar, R., Russ, T.: Automatically constructing semantic web services from online sources. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 17–32. Springer, Heidelberg (2009)
15. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: SIGMOD Conference 2001, pp. 509–520 (2001)
16. Doan, A., Domingos, P., Halevy, A.Y.: Learning to Match the Schemas of Data Sources: A Multistrategy Approach. Machine Learning 50(3), 279–301 (2003)
17. Heß, A., Kushmerick, N.: Learning to attach semantic metadata to web services. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 258–273. Springer, Heidelberg (2003)
18. Rahm, E., Bernstein, P.: On matching schemas automatically. VLDB. Journal 10(4) (2001)
19. Battle, R., Benson, E.: Brinding the semantic web and web 2.0 with Representational State Tranfer (REST). Web semantics 6, 61–69 (2008)
20. Braga, D., Ceri, S., Martinenghi, D., Daniel, F.: Mashing Up Search Services. IEEE Internet Computing 12(5), 16–23 (2008)
21. Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, Y.H., Simmen, D., Singh, A.: Damia: a data mashup fabric for intranet applications. In: Proceedings of the 33rd Int. Conference on VLDB 2007 Endowment, pp. 1370–1373 (2007)
22. Resende, L.: Handling heterogeneous data sources in a SOA environment with service data objects (SDO). In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 895–897. ACM, New York (2007)
23. Fielding, R.: Architectural Styles and The Design of Network-based Software Architectures. PhD thesis, University of California, Irvine (2000)

Towards Custom Cloud Services
Using Semantic Technology to Optimize Resource Configuration

Steffen Haak and Stephan Grimm

Research Center for Information Technology (FZI)
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe, Germany
{haak,grimm}@fzi.de

Abstract. In today's highly dynamic economy, businesses have to adapt quickly to market changes, be it customer, competition- or regulation-driven. Cloud computing promises to be a solution to the ever changing computing demand of businesses. Current SaaS, PaaS and IaaS services are often found to be too inflexible to meet the diverse customer requirements regarding service composition and Quality-of-Service. We therefore propose an ontology-based optimization framework allowing Cloud providers to find the best suiting resource composition based on an abstract request for a custom service. Our contribution is three-fold. First, we describe an OWL/SWRL based ontology framework for describing resources (hard- and software) along with their dependencies, interoperability constraints and meta information. Second, we provide an algorithm that makes use of some reasoning queries to derive a graph over all feasible resource compositions based on the abstract request. Third, we show how the graph can be transformed into an integer program, allowing to find the optimal solution from a profit maximizing perspective.

1 Introduction

In the last decades, most companies regarded their IT systems as a necessary but unimportant part of their business models. Investments into IT infrastructure were cost- rather than quality-driven and long product life cycles made investments long-term rather than flexible. However the Internet's increasing importance for global business, emerging new paradigms like Service Oriented Architectures (SOA) and Cloud computing, and the ever increasing competition fostered through globalization has made many companies rethink their IT strategy. Even in traditional industries, IT no longer just acts as a supporting unit - in many cases IT has become a strategic competitive factor. The challenge has become even harder, as product and service life cycles become shorter and adaptation to market changes needs to be more agile as ever before [6]. Naturally this challenge is encountered by all parts of modern business, putting high demands on flexibility and agility on a company's supporting IT services.

Cloud computing (and its SaaS, PaaS and IaaS offerings) promises to be a solution for cutting costs in IT spending while simultaneously increasing flexibility. According to thecloudmarket.com [1], there already exist over 11.000 different preconfigured Amazon EC2 images from various providers, that can be run as virtual appliance on Amazon's Elastic Compute Cloud (EC2). The large variety indicates the importance of custom service offers. However, the customer has little support in the technical and economic decision process for selecting an appropriate image and deploying it on the Cloud. It is neither guaranteed, that there finds an image that is configured exactly according to the customer needs.

Apparently there is a great business opportunity for Cloud providers who manage to offer Custom Cloud Services tailored to their customers' needs. Transferring the selection and deployment process to the provider simplifies the customer's decision process significantly. In such a scenario, the provider faces the challenging task of automatically finding the optimal service composition based on the customer request, from both a technical and an economic view. A customer requesting a database service having for example only preferences on the underlying operating system or the available storage space leaves a lot of room for economic optimization of the service composition. A MySQL database running on a Linux-based virtual machine (VM) might be advantageous over the more costly Oracle alternative on a Windows-based VM.

Custom Cloud Services, as referred to in this paper, are compositions of commercial off-the-shelf software, operating system and virtualized hardware, bundled to offer a particular functionality as requested by a customer. The functional requirements usually leave many choices when choosing required resources from groups that offer equal or similar functionalities as in the above mentioned example. Finding the optimal choice involves many different aspects, including (among others) technical dependencies and interoperability constraints, customer preferences on resources and Quality-of-Service (QoS) attributes, capacity constraints and license costs. In a sense, it is a configuration problem as known from typical configurators for products like cars, etc. However, stated as integer programming or constraint optimization problem [15,7], it is underspecified as not all configuration variables (the set of required service resource types) can be clearly specified ex ante from the customer request. The missing variables however can be derived dynamically as they are implicitly contained in the resources' dependencies. For example, a customer request for a CRM software might implicitly require an additional database, some underlying operating system on some virtualized hardware.

As mentioned before, traditional linear or constraint programming techniques require the knowledge of all variables. In order to overcome this problem, we propose an ontology-based approach for a convenient and standardized knowledge representation of all known Cloud service resources, allowing to derive the complete configuration problem by subsequently resolving resource dependencies.

The remainder of this paper is structured as follows. Section 2 describes an example use case, which helps us to derive requirements for the designated framework. These requirements are also used for a qualitative evaluation of our approach and to distinguish it from related work.

In Section 3 we describe our main contribution. We start by describing our understanding of functional requirements, which serve as input for the ontology-based optimization framework. We then propose the usage of a three-fold ontology system to serve as knowledge base. We distinguish between a generic service ontology, that contains the meta concepts provided in this paper, a domain ontology that makes use of these concepts and contains the actual knowledge about known infrastructure resources, and a result ontology that is used to represent a graph, spanning a network over different choices based on abstract dependency relations that can exist between different resources (e.g. that every application needs some operating system). We formally describe this dependency graph and show how it can be derived algorithmically, making use of different queries to the ontology system. Further we show how this graph can be used to obtain all feasible infrastructure compositions. In addition, we show how the graph can be transformed into an integer program for finding the profit optimal configuration with respect to customer preferences and costs. For knowledge representation, we make use of the Semantic Web ontology and rule languages OWL and SWRL combined with SPARQL querying facilities.

Section 4 describes a proof-of-concept implementation of the presented framework and reviews the requirements from Section 2 with respect to our contribution. In Section 5 we conclude this paper by giving an overview on open issues and an outlook on future research.

2 Use Case, Requirements and Related Work

For a better understanding and evaluating our research we present a use case describing an example request for a Custom Cloud Service. We use it to derive a set of required properties for our contribution. The section is concluded by an overview on the related literature.

2.1 Use Case

Consider a service request from a customer who wants to set up an online survey for a two months period. The customer has no preferences regarding the survey system, however would slightly prefer a Windows-based operating system over Linux. As he is quite acquainted with Oracle products, the survey system's underlying database is preferably also Oracle-based. The expected workload in form of concurrent users is unclear, however this is not considered a problem due to scalability of Cloud services.

In a typical scenario the provider wants to maximize profit, i.e. the difference between the offer price and the accruing costs. The challenge is to find the configuration that is in line with the customer preferences, thus having a high offer price, while simultaneously considering cost aspects.

2.2 Requirements

Based on the use case, we can now derive a set of requirements, defining and clarifying the goals of the desired solution. We identify five major properties that we find necessary to provide an adequate solution for finding the optimal service configuration in this case:

R 1 (Top Down Dependency Resolution). *Automatic resolution of all transitive dependencies between resource classes, starting from the top level functional requirement resources until no more unresolved dependencies exist. Thus deducting all variables for the Custom Cloud Service configuration problem.*

R 2 (Functional Requirements). *The functional requirements for the designated service should be describable by abstract or concrete resources (on different levels).*

R 3 (Customer Preferences). *The approach should be able to consider customer preferences regarding different configuration options.*

R 4 (Interoperability Check). *The interoperability/compatibility between resources has to be validated. For reducing the modeling overhead, this validation should be possible on both instance and higher abstraction levels.*

R 5 (Profit Optimization). *The profit maximizing Custom Cloud Service configuration has to be found. I.e. the configuration yielding the greatest difference between the achievable offer price for a configuration and the accruing costs on provider side.*

Additionally we require *correctness*, *completeness* and *consistency* for the designated decision support mechanism. However, as these properties are rather generic we do not consider them for our related literature review.

2.3 Related Work

For solving the technically and economically complex problem of deriving and optimizing configuration alternatives, we have to touch a broad spectrum of research areas, from Web service composition to techniques from operations research and constraint programming.

Berardi et al. [4] address the problem of automatic service composition by describing a service in terms of an execution tree, then making use of finite state machines to check for possible service compositions that match a requested behavior. Lécué et al. [11] present an AI planning-oriented approach using Semantics. Based on causal link matrices, the algorithm calculates a regression-based optimal service chain. Both [4] and [11] concentrate on input/output based matching of Web services, thus they are not suitable for infrastructure service compositions, where the interfaces are much more complex and cannot be described in terms of input and output. Another work from the Semantic Web context by Lamparter et al. [10] describes the matching process between requests and offers of Web services. This approach is a related example for the matching of interoperability constraints (R 4), however it does not include dependency resolution.

Blau et al. [5] propose an ontology-based tool for planning and pricing of service mash-ups. The tool can be used to compose complex Web services from a set of known atomic services, which are stored in a domain specific ontology. Afterwards the complex service can be validated based on axioms and rules in the ontology.

Sabin et al. [15] present different constraint programming approaches for product configuration. Van Hoeve [7] describes optimization approaches for constraint satisfaction problems. In [9] an optimization framework combining constraint programming with a description logic is provided. All related to R 5.

We also want to mention two software tools that include a transitive dependency management, thus are mainly related to requirements 1 and 4. Advanced Packaging Tool (APT) [16] is a package management system to handle the installation and removal of software packages on Linux distributions. APT allows to automatically install further packages required by the desired software to avoid missing dependencies. The dependency management also includes compatibility checks, however only in form of a rather simple version level check. Another dependency manager is Apache Ivy [2]. Dependencies in Ivy are resolved transitively, i.e. you have to declare only direct dependencies, further dependencies of required resources are resolved automatically. Both approaches do not allow semantic annotations to allow more complex dependencies or interoperability constraints.

3 Custom Cloud Service Configuration

The contribution of our work consists of several elements in the context of configuring Custom Cloud Services. For a common understanding we first want to provide our own definition:

Definition 1 (Custom Cloud Service). *A Custom Cloud Service is a Cloud Computing service, composed by a set of software components and existing Cloud infrastructure resources, according to the abstract functional requirements and non-functional preferences of an individual customer or target customer group.*

In the following section we provide a formal definition on the functional requirements on Custom Cloud Services, tailored to the described use case. We then present an ontology model, that can be used to formally capture the knowledge about Cloud resources, their interdependencies and compatibility constraints as well as cost meta information. We also present an algorithm to build a dependency graph based on this knowledge, which can be seen as completely specified configuration problem. Lastly we show how the graph can be transformed into an integer program, allowing to find the optimal solution from a cost-benefit perspective. A sequence diagram to visualize the proposed interaction is depicted in Figure 1.

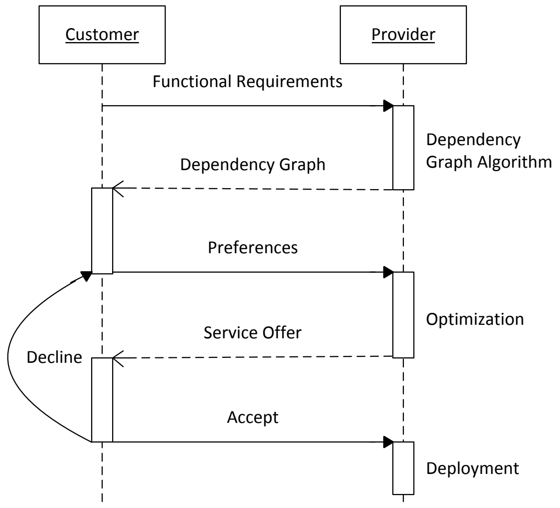


Fig. 1. Interaction Process

3.1 Functional Requirements

Functional requirements describe *what* is needed, i.e. the components required for a functioning service, whereas non-functional requirements describe *how* it is needed, i.e. the desired quality of the service. For the declarative description of functional requirements, we build on concepts taken from an *ontology* that contains background information in form of an abstract resource description layer (service ontology) and a domain-specific layer containing domain-dependent descriptions of available resources (domain ontology). We define functional requirements as follows.

Definition 2 (Functional Requirements). *For a background ontology O the functional requirements for deploying a service is described by the tuple $R = (\mathcal{C}, t)$, with $\mathcal{C} = \{C_1, \ldots, C_n\}$, a set of concepts in O, and t, the requested time period for the service.*

For illustration, the functional requirements of the presented use case would be the tuple $R = (\{OnlineSurvey\}, 1440h)$.

3.2 Knowledge Representation

For knowledge representation, we rely on the Web Ontology Language (OWL) [12] specification for several reasons. It has been established as the leading Semantic Web standard, is widely spread thus offering a large set of modeling tools and inference engines, is well documented and offers description logics (DL) expressiveness that fits well our anticipated description of services at class-level. In addition, OWL comes with a standardized, web-compliant serialization which ensures interoperability over the borders of single institutions.

For stating more complex compatibility constraints that go beyond the DL expressivity, we also make use of the Semantic Web Rule Language (SWRL) [8] in combination with OWL, while we restrict ourselves to DL-safe rules [14] as is a decidable fragment supported by OWL reasoners like Pellet [17] or KAON2 [13].

We make use of three different ontologies. A generic *service ontology*, which defines the fundamental concepts of our model. The actual knowledge on the known software and Cloud resources is modeled in the *domain ontology* and will differ for the various users or use cases. While the service ontology is a static model, the domain ontology has to be dynamic, i.e. it will need constant updates on the current infrastructure situation. The third ontology is used to store the results of the algorithm that derives all alternatives by dissolving the resources' dependencies. Both domain and *result ontology* import the fundamental concepts defined in the *service ontology*.

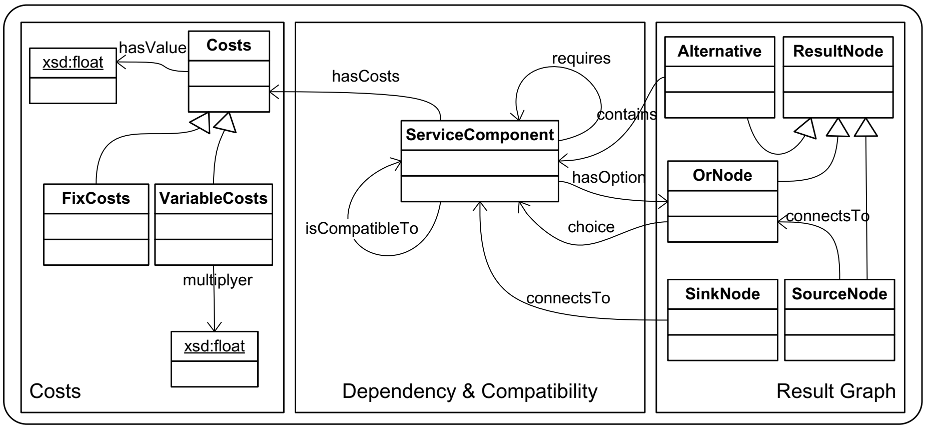


Fig. 2. Service Ontology

Service Ontology. The service ontology is partially depicted in Figure 2. For the reader's convenience we illustrate the ontology in UML notation where UML classes correspond to OWL concepts, UML associations to object properties, UML inheritance to sub-concept relations and UML objects to OWL instances. Basically, the service ontology provides concepts for three different aspects:

1. Service resources along with dependencies and compatibility information, needed to derive all valid service configuration alternatives
2. Cost meta information for evaluating these alternatives
3. Structure elements needed for an ontology representation of the dependency graph

The most fundamental concept for deriving all feasible deployment alternatives is the class *ServiceComponent* along with the corresponding object properties *requires* and *isCompatibleTo*. The *requires* property is used to describe the functional dependency between two resource instances. In most cases, dependencies can and should be described in an abstract way at class-level. We can do this by including the dependency relation into the class axiom in conjunction with an object restriction in the form of an existential quantifier on the required class:

$$\begin{aligned} ComponentA \sqsubseteq\ & ServiceComponent \\ & \sqcap \exists requires.ComponentB \end{aligned}$$

Hereby we state that each resource of type *ComponentA* requires some resource of type *ComponentB*. As a more concrete example, we could state that every instance of the class Application requires at least some operating system:

$$\begin{aligned} \mathit{Application} \sqsubseteq\ & ServiceComponent \\ & \sqcap \exists requires.OS \end{aligned}$$

The compatibility can be asserted on instance level using the *isCompatibleTo* object property. That implies that there has to be one object relation between all possible combinations of interdependent resources. To reduce this modeling overhead, we propose the usage of SWRL rules, which allow us to state compatibility on class level:

$$\mathit{isCompatibleTo(x,y)} \leftarrow \tag{L1.1}$$
$$\mathit{ComponentA(x)} \tag{L1.2}$$
$$\mathit{ComponentB(y)} \tag{L1.3}$$

We can exploit the full expressiveness of DL-safe SWRL rules. E.g. to state that all versions of *MySQL* are compatible to all *Windows* versions except *Windows 95*, we include the following rule:

$$\mathit{isCompatibleTo(x,y)} \leftarrow \tag{L2.1}$$
$$\mathit{MySQL(x)} \tag{L2.2}$$
$$\mathit{Windows(y)} \tag{L2.3}$$
$$\mathit{differentFrom(y,'Windows95')} \tag{L2.4}$$

For inclusion of cost information we distinguish between non-recurring *FixCosts* and recurring *VariableCosts*. The latter being more complex, as the total amount depends on another variable, which has to be defined in the context to serve as a multiplier. E.g. the overall usage fee for a cloud provider depends on the planned usage period for the service.

Domain Ontology. The domain ontology uses the concepts described in the preceding section to capture the knowledge about the Cloud service resources of a certain domain of interest. This allows to easily use the same technology for many different contexts, just by loading a different domain ontology. In addition, knowledge can be combined by loading several domain ontologies, as long as this does not lead to inconsistencies.

Result Ontology. By resolving the transitive dependencies for the set of resources from the functional requirements, it is clear that we cannot add any knowledge, we can only make additional knowledge explicit, that is contained in the knowledge base implicitly.

For persisting the extended model, we also rely on an OWL ontology, such that it can be used for further reasoning tasks. The result ontology makes use of the concepts *SourceNode*, *SinkNode*, *OrNode* and *Alternative*, all defined in the service ontology. *SourceNode* and *SinkNode* are a helper nodes to have a distinct starting and ending points in the graph. They correspond to the source and sink nodes in a network. The *OrNode* is introduced to capture the branching whenever there is more than one compatible resource instance that fulfills the dependency requirement.

In the remainder of the paper we work with a more formal notation for the dependency graph. Note that both notations are semantically the same. An example graph is depicted in Figure 3.

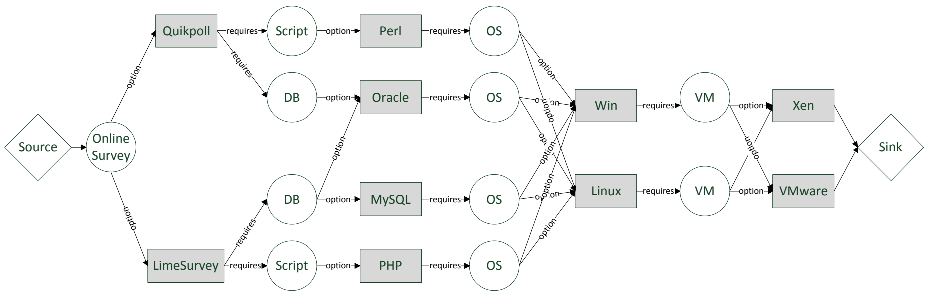


Fig. 3. Example Dependency Graph

Definition 3 (Dependency Graph). *For a background ontology O and a functional requirements tuple $R = (\mathcal{C}, t)$, the* dependency graph *$G = (V, E)$ is a directed, acyclic, labeled graph with vertices V and edges E and a labeling function $\mathcal{L}$, recursively defined as follows:*

- *$n_0 \in V$ is the source node of G*
- *$n_\infty \in V$ is the sink node of G*
- *there is a node $n_C \in V$ with label $\mathcal{L}(n_C) = C$ for each (atomic or nominal) class $C \in \mathcal{C}$ and an edge $(n_0, n_C) \in E$*
- *if $n \in V$ is a node with an atomic class label $\mathcal{L}(n) = A$ then there is a node n_o with label $\mathcal{L}(n_o) = \{o\}$ for each individual $o \in O$ with $O \models A(o)$, and an edge $e = (n, n_o)$ with label $\mathcal{L}(e) = \textsf{or}$*

– *if $n \in V$ is a node with a nominal class label $\mathcal{L}(n) = \{o\}$ then there is a node n_C with label $\mathcal{L}(n_C) = C$ for each atomic or nominal class C with $O \models C(x)$ for all x such that $O \models requires(o, x)$ and $O \models isCompatibleTo(o, x)$, and an edge $e = (n, n_C)$ with label $\mathcal{L}(e) =$ and.*

3.3 Dependency Graph Algorithm

OWL reasoners typically construct models for answering standard reasoning tasks but do not expose them as such. Since we need to explicitly access such models as configuration alternatives at the instance-level, we chose an algorithmic solution, making use of OWL reasoning capabilities in between the construction of dependency graphs.

Algorithm 1. determineDependencies(R, O; G) – Initiate the construction of a dependency graph.

```
Require: a functional requirement tuple $R = (\mathcal{C}, F)$ and ontology $O$
Ensure: $G$ contains the dependency graph for $R$
  $V := \{n_0, n_\infty\}$, $V^* := E := \emptyset$
  for all $C \in \mathcal{C}$ do
    $V := V \cup \{n_C\}$, $\mathcal{L}(n_C) := C$
    $E := E \cup \{(n_0, n_C)\}$, $\mathcal{L}((n_0, n_C)) =$ and
    for all $o$ with $O \models C(o)$ do
      $V := V \cup \{n_o\}$, $\mathcal{L}(n_o) = \{o\}$
      $E := E \cup \{(n_C, n_o)\}$, $\mathcal{L}((n_C, n_o)) =$ or
      deductServiceInfrastructure($O$, $o$, $G$, $V^*$)
    end for
  end for
```

The procedure determineDependencies in Algorithm 1 initiates the construction of a dependency graph, starting from functional requirements R, and calls the procedure deductServiceInfrastructure in Algorithm 2, which recursively finds suitable service resource instances by following the object property *requires*.

In the procedure getRequiredClasses we invoke the reasoning engine with the following SPARQL query to find all implicitly stated dependencies, i.e. through a class axiom rather than explicitly on instance level:

```
SELECT ?sub ?t ?obj
WHERE {
    ?sub owl:sameAs dm:component .
    ?sub so:requires _:b0 .
    _:b0 rdf:type ?t .
    ?obj rdf:type ?t }
ORDER BY ?t
```

so hereby refers to the name space of the service ontology, *dm* to the name space of the domain ontology. The literal *_:b0* refers to a blank node, i.e. there

Algorithm 2. deductServiceInfrastructure(O, o; G, V^*) – Recursively construct a dependency graph for a given ontology and resource instance.

```
Require: an ontology $O$ and a resource instance $o \in O$
Ensure: $G = (V, E)$ contains a dependency graph for $o$, $V^*$ contains all resource
  instance nodes visited
  $V^* := V^* \cup \{n_o\}$
  $\mathcal{C} := \emptyset$, getRequiredClasses($o$; $\mathcal{C}$)
  if $\mathcal{C} = \emptyset$ then $E := E \cup \{(n_o, n_\infty)\}$
  for all $C \in \mathcal{C}$ do
    $V := V \cup \{n_C\}$, $\mathcal{L}(n_C) := C$
    $E := E \cup \{(n_o, n_C)\}$, $\mathcal{L}((n_o, n_C)) = \mathsf{and}$
    for all $o'$ with $O \models C(o')$ and $O \models isCompatibleTo(o, o')$ do
      $V := V \cup \{n_{o'}\}$, $\mathcal{L}(n_{o'}) = \{o'\}$
      $E := E \cup \{(n_C, n_{o'})\}$, $\mathcal{L}((n_C, n_{o'})) = \mathsf{or}$
      if $n_{o'} \notin V^*$ then deductServiceInfrastructure($O$, $o'$ , $G$, $V^*$)
    end for
  end for
```

do not exist two individuals for which we find the requires property, but based from the axiomatic knowledge we know there has to be at least one.

The query will answer us with a set of all types of these anonymous individuals. This has one disadvantage: we are only interested in the most specific class assertions. E.g. if it was stated that every application needs an operating system, the query would have the class *OS* in its result set, however also every superclass up to *Thing*.

Therefore, in a second step, we need to find out the most specific classes, i.e. all classes that have no subclasses also contained in the result set. This can be achieved by a simple algorithm which has a worst case runtime of $O(n^2)$ subsumption checks.

As there are redundant dependencies, which by themselves again might have further dependencies, we memorize the visited resources (V^*), as we do not need to resolve their dependencies more than once.

Further the algorithm remembers unfulfilled requirements and recursively traces them back, deleting unfeasible paths. We have not included these steps in the above printed pseudo algorithm, as they would only confuse the reader.

Constraint Satisfaction Problem In another formal representation of the graph (Figure 4), denoted as G^F, we can recognize the analogy to a constraint satisfaction problem [7], which is defined by a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$, a set of domains $\mathcal{D} = \{D_{X_1}, \ldots, D_{X_n}\}$ defining the possible values for $\mathcal{X}$ and a set of constraints $\mathbb{C}$. In G^F the variables $\mathcal{X}$ are equal to vertices labeled with classes $C \in O$, and the domains D_{X_i} equal to vertices labeled with resource instances $o \in O$. Vertices, labeled with an identical resource class C, are subsumed by one variable X_C. The constraints in $\mathbb{C}$ can be derived from the edges denoting the interoperability between resources.

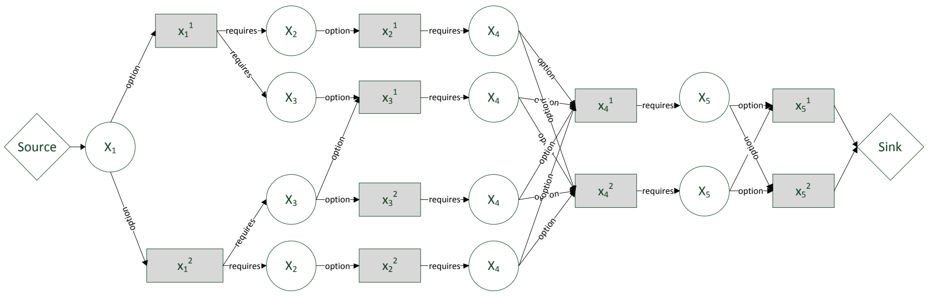


Fig. 4. Formal Dependency Graph

3.4 Preferences

As shown in Figure [1], after receiving the dependency graph, the customer can specify preferences. For quantifying the customer satisfaction for a certain configuration, we introduce the notion of a scoring function [3]. The scoring function maps the customer preferences on certain configuration choices to a real number in the interval $[0, 1]$. We achieve that by a weighted aggregation of the single preference values regarding the different configuration choices (cf. Section 3.5). For expressing non-compensating preferences, we define a set of additional restrictions $\mathcal{R}$ added to set of constraints $\mathbb{C}$.

Definition 4 (Preferences). *For a dependency graph G^F the customer preferences are described by the triplet $\mathcal{P} = (P, \Lambda, \mathcal{R})$, with $\mathcal{P} = \{P_1, \ldots, P_n\}$, a set of preference vectors, $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ a set of weights for each variable X_i in $\mathcal{X}$ with $\sum_i \lambda_i = 1$ and $\mathcal{R}$ a set of non-compensating restrictions.*

Example. For a variable X_{OS} representing the various operating system choices with $D_{X_{OS}} = \{Win, Linux\}$, the preferences are denoted by the vector $P_{OS} = (1, 0.8)^T$, expressing the slight preference for a Windows-based system, as described in our use case. Analogously, we would denote $P_{DB} = (0.5, 1)^T$ with $D_{X_{DB}} = \{MySQL, Oracle\}$. If both preference values are considered equally important, we would denote $\lambda_{OS} = \lambda_{DB} = 0.5$. A non-compensating restriction could be that the online survey has to be PHP-based, denoted by $\mathcal{R} = \{X_{Script} \overset{!}{=} PHP\}$.

3.5 Optimization

Having the dependency graph G^F and the customer preferences $\mathcal{P}$ we want to find the optimal configuration that will be offered to the customer. The optimal configuration can differ in various scenarios with different pricing schemes and potential additional constraints (like capacity restrictions). In this work we define the optimum as the configuration that yields the highest profit, i.e. the achieved price minus the accruing costs. Hereby we assume, that customer is sharing his willingness to pay for a service perfectly fulfilling all his preferences, denoted by α. Extensions to this simple economic model are considered in ongoing research.

In a naive approach, we could try to iterate over all feasible configurations in G^F. One configuration is a sub graph, as the meaning of the edges can be interpreted as *and* and *or*. As a matter of fact, we can rewrite the dependency graph as Boolean formula.If we convert the Boolean formula (which is already in negation normal form) into its disjunctive normal form (DNF) by a step-wise replacement using de Morgan's laws, we exactly get an enumeration over all sets of resource instances that reflect the different configurations.

However, we are interested in the optimal configuration, thus iterating over all configurations might not be the best choice as it is very costly with respect to runtime. We therefore set up an integer program, which calculates the profit maximizing configuration. The variables from the constraint satisfaction problem are by modeled a vector of binary decision variables $X_i = (X_i^1, \dots, X_i^m)$ for m different choices.

$$\begin{aligned} \underset{\mathcal{X}}{\text{maximize}} \quad & \alpha \cdot S(\mathcal{X}) - C(\mathcal{X}) & \\ \text{subject to} \quad & \sum_{x_i^j \in X_i} x_i^j = 1 & \forall X_i \in \mathcal{X} \\ & X_i \cdot X_k \leq I_{ik} & \forall i, j : X_i \rightarrow_r X_k \\ & R \in \mathcal{R} & \text{constraints from } \mathcal{P} \end{aligned}$$

with

$$S(\mathcal{X}) = \sum_{X_i \in \mathcal{X}} \lambda_i \cdot X_i \cdot P_i$$
$$C(\mathcal{X}) = \sum_{X_i \in \mathcal{X}} C_i \cdot X_i$$

I_{ik} hereby denotes an interoperability matrix between X_i and X_k that can be derived from G^F. The cost function $C(\mathcal{X})$ merely is the sum over all costs for the chosen resources, which are stored in the ontology O. In case of variable costs, we multiply the cost value with the requested time period for the service t from R (cf. Section 3.1).

4 Evaluation

We evaluate our contribution qualitatively by having implemented a proof-of-concept prototype and comparing the presented approach to the requirements from Section 2.2.

4.1 Implementation

The implemented prototype can be executed using Java Web Start[1]. For using the prototype, a domain ontology (an example ontology is provided) has to be

[1] http://research.steffenhaak.de/ServicePlanner/

loaded, before one can add the set of resources $\mathcal{C}$. Eventually, the algorithms can be started by using the menu items *ResolveDependencies* and *Evaluate*. However, not all functionalities presented in this paper are integrated. We can derive all feasible configurations using the described DNF approach. The integer program has been implemented using CPLEX, thus not being part of the downloadable prototype. As reasoning engine we have chosen Pellet [17], as to our knowledge it is the only OWL DL reasoner that is capable of both SWRL rules and SPARQL queries that involve blank nodes.

4.2 Requirements Review

Taking a look back to the requirements derived from the use case, we have presented a framework to model knowledge about Cloud resource dependencies and compatibilities. Based on this knowledge base a set of functional requirements can be defined in from of classes from the ontology on arbitrary abstraction level. It is used to derive a dependency graph ensuring interoperability of all configurations. Based on the graph, the customer can define preferences for each variable, which are subsumed in a scoring function. Stated as constraint satisfaction problem, an integer program finds out the profit maximizing configuration, making use of the scoring function and the cost information stored in the ontology. The proposed approach therefore fulfills requirements R 1 to R 5.

5 Conclusion

In this paper, we propose an ontology-based optimization framework for finding the optimal Cloud service configuration based on a set of functional requirements, customer preferences and cost information. We do this by means of an OWL DL approach, combined with DL-safe SWRL rules and SPARQL querying facilities. We can model dependencies between resource classes of any abstraction level and use complex rules to ensure compatibility between resources. An integer program allows us to find the profit maximizing configuration. We provide a prototypical implementation as proof-of-concept.

We recognize several reasonable extensions and shortcomings to our approach. The economic model for the profit maximization is very simplistic. It is arguable that the customer is willing to give price his preferences and willingness to pay in a truthful manner. Further extensions from an economic perspective are the integration of capacity constraints and optimizing several concurrent requests. Ongoing research is dealing with both issues.

From the semantic perspective, a more complex cost and quality model would be beneficial. In addition, we plan to investigate on how the proposed domain ontology can be maintained collaboratively by incentivizing resource suppliers to contribute the necessary knowledge about their resources as interoperability and dependencies themselves.

We also recognize the need for a better evaluation, qualitatively, through relying on an industry use case, and quantitatively, by analyzing both the economic benefit of our solution and its computational complexity.

References

1. The Cloud Market EC2 Statistics (2010), http://thecloudmarket.com/stats
2. Apache. Apache Ivy (2010), http://ant.apache.org/ivy/
3. Asker, J., Cantillon, E.: Properties of Scoring Auctions. The RAND Journal of Economics 39(1), 69–85 (2008)
4. Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M.: Automatic service composition based on behavioral descriptions. Int. J. of Cooperative Information Systems 14(4), 333–376 (2005)
5. Blau, B., Neumann, D., Weinhardt, C., Lamparter, S.: Planning and pricing of service mashups. In: 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pp. 19–26 (2008)
6. Gaimon, C., Singhal, V.: Flexibility and the choice of manufacturing facilities under short product life cycles. European Journal of Operational Research 60(2), 211–223 (1992)
7. van Hoeve, W.: Operations Research Techniques in Constraint Programming. Ph.D. thesis, Tepper School of Business (2005)
8. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission 21 (2004)
9. Junker, U., Mailharro, D.: The logic of ilog (j) configurator: Combining constraint programming with a description logic. In: Proceedings of Workshop on Configuration, IJCAI, vol. 3, pp. 13–20. Citeseer (2003)
10. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: Proceedings of the 16th international conference on World Wide Web, pp. 1013–1022. ACM Press, New York (2007)
11. Lécué, F., Léger, A.: A formal model for web service composition. In: Proceeding of the 2006 conference on Leading the Web in Concurrent Engineering, pp. 37–46. IOS Press, Amsterdam (2006)
12. McGuinness, D.L., Van Harmelen, F., et al.: OWL web ontology language overview. W3C recommendation 10, 2004–03 (2004)
13. Motik, B., Studer, R.: KAON2–A Scalable Reasoning Tool for the Semantic Web. In: Proceedings of the 2nd European Semantic Web Conference (ESWC 2005), Heraklion, Greece (2005)
14. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 3(1), 41–60 (2005)
15. Sabin, D., Freuder, E.: Configuration as composite constraint satisfaction. In: Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop, pp. 153–161 (1996)
16. Silva, G.: APT howto (2003), http://www.debian.org/doc/manuals/apt-howto/index.en.html
17. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)

One Tag to Bind Them All: Measuring Term Abstractness in Social Metadata

Dominik Benz[1,⋆], Christian Körner[2,⋆],
Andreas Hotho[3], Gerd Stumme[1], and Markus Strohmaier[2]

[1] Knowledge and Data Engineering Group (KDE), University of Kassel
{benz,stumme}@cs.uni-kassel.de
[2] Knowledge Management Institute, Graz University of Technology
{christian.koerner,markus.strohmaier}@tugraz.at
[3] Data Mining and Information Retrieval Group, University of Würzburg
hotho@informatik.uni-wuerzburg.de

Abstract. Recent research has demonstrated how the widespread adoption of collaborative tagging systems yields emergent semantics. In recent years, much has been learned about how to harvest the data produced by taggers for engineering light-weight ontologies. For example, existing measures of tag similarity and tag relatedness have proven crucial step stones for making latent semantic relations in tagging systems explicit. However, little progress has been made on other issues, such as understanding the different levels of tag generality (or tag abstractness), which is essential for, among others, identifying hierarchical relationships between concepts. In this paper we aim to address this gap. Starting from a review of linguistic definitions of word abstractness, we first use several large-scale ontologies and taxonomies as grounded measures of word generality, including Yago, Wordnet, DMOZ and WikiTaxonomy. Then, we introduce and apply several folksonomy-based methods to measure the level of generality of given tags. We evaluate these methods by comparing them with the grounded measures. Our results suggest that the generality of tags in social tagging systems can be approximated with simple measures. Our work has implications for a number of problems related to social tagging systems, including search, tag recommendation, and the acquisition of light-weight ontologies from tagging data.

Keywords: tagging, generality, measures, emergent semantics, folksonomies.

1 Introduction

Since the advent of participatory web applications like Flickr[1], Youtube[2] or Delicious[3], social annotations (especially in the form of collaboratively created

⋆ Both authors contributed equally to this work.
[1] http://www.flickr.com
[2] http://www.youtube.com
[3] http://www.delicious.com

keywords or *tags*) form an integral part of current approaches to collaborative knowledge management. Analyses of the structure of the resulting large-scale bodies of human-annotated resources have shown several interesting properties, especially regarding the presence of *emergent semantics*. Motivated by the vision of bridging the gap towards the Semantic Web, much has been learned in recent years about how to harvest the data produced by taggers for engineering light-weight ontologies. However, little progress has been made on other issues, such as understanding the different levels of tag generality (or tag abstractness), which is essential for e.g. identifying hierarchical relationships between concepts. While several methods of deriving taxonomies from tagging systems have been proposed, a systematic comparison of the underlying notion of abstractness is largely missing.

This paper aims to address this gap by presenting a systematic analysis of various folksonomy-derived notions of term abstractness. Starting from a review of linguistic definitions of word abstractness, we first use several large-scale ontologies and taxonomies as grounded measures of word generality, including Yago, Wordnet, DMOZ and WikiTaxonomy. Then, we introduce and apply several folksonomy-based methods to measure the level of generality of given tags. We evaluate these methods by comparing them with the grounded measures.

Our results show that the abstractness judgments by some of the measures under consideration come close to those of well-defined and manually built taxonomies. Furthermore, we provide empirical evidence that tag abstractness can be approximated by simple measures. The results of this research are relevant to all applications who benefit from a deeper understanding of tag semantics, e.g. ontology learning or clustering algorithms, tag recommendations systems or folksonomy navigation facilities. In addition, our results can help to alleviate the problem of varying "basic levels" in folksonomies [11] by matching more specific terms (used usually by domain experts) to more general ones.

This paper is structured as follows: At first we give an overview about related work, especially regarding term abstractness and emergent semantics (Section 2). This is followed by some basic notions in Section 3. In the subsequent section we give an overview of the introduced measures (section 4) and evaluate them in Section 5 with the help of established datasets as ground truth and a user study. Finally we conclude in Section 6 and point to future work.

2 Related Work

The first research direction relevant to this work has its roots in the analysis of the structure of collaborative tagging systems. Golder and Huberman [11] provided a first systematic analysis, mentioning among others the hypothesis of "varying basic levels" – according to which users use more specific tags in their domain of expertise. However, the authors only provided exemplary proofs for this hypothesis, lacking a well-grounded measure of tag generality. In the following, a considerable number of approaches proposed methods to make the implicit semantic structures within a folksonomy explicit [19,13,22,2]. All of the previous works comprise in a more or less explicit way methods to capture the "generality" of a tag (e.g. by investigating the centrality of tags in a similarity graph or by applying a statistical model of subsumption) – however, a comparison of the chosen methods has not been given. Henschel et al. [12] claim to generate more precise taxonomies by an entropy filter. In our own recent work [17] we showed that the quality of semantics within a social tagging system is also dependent on the tagging habits of individual users, Heymann [14] introduced another entropy-based tag generality measure in the context of tag recommendation.

From a completely different point of view, the question of which factors determine the generality or abstractness of natural language terms has been addressed by researchers coming from the areas of Linguistics and Psychology. The psychologist Paivio [20] published in 1968 a list of 925 nouns along with human concreteness rankings; an extended list was published by Clark [8]. Kammann [16] compared two definitions of word abstractness in a psychological study, namely imagery and the number of subordinate words, and concluded that both capture basically independent dimensions. Allen et al. [1] identify the generality of texts with the help of a set of "reference terms", whose generality level is known. They also showed up a correlation between a word's generality and its depth in the WordNet hierarchy. In their work they developed statistics from analysis of word frequency and the comparison to a set of reference terms. In [25], Zhang makes an attempt to distinguish the four linguistic concepts fuzziness, vagueness, generality and ambiguity.

3 Basic Notions

As stated above, the main intent of a term generality measure is to allow a differentiation of lexical entities $l_1, l_2, \ldots$ by their degree of abstractness (i.e. their ability to "bind" other tags). As a prerequisite for a formalization of this problem, we will first introduce a common terminology which allows us to refer to the usage of lexical entities in the context of taxonomies and collaborative tagging systems in a unified way.

Taxonomies, Core Ontologies and Lexicons. First of all, according to [7] a *taxonomy* can also be regarded as a part of a *core ontology*, [3] $\mathbb{O} := (C, \mathit{root}, \geq_C, L^C, \mathcal{F})$, whereby C is a set of concept identifiers and *root* is a designated root concept for the partial order $\geq_C$ on C. $\geq_C$ is called concept hierarchy or *taxonomy*; if $c_1 \geq_C c_2 (c_1, c_2 \in C)$, then c_1 is a *superconcept* of c_2, and hence we assume c_1 to be more abstract or "general" than c_2. L^C is a set of lexical labels for concepts and a mapping relation $\mathcal{F}$ which associates concepts with their respective label. Please note that a concept c can be associated with one or more labels, i.e. $\forall l \in L^C \colon |\{l : (c, l) \in \mathcal{F}\}| \geq 1$. As an example, in scientific contexts the terms "article" and "paper" are often used synonymously, which would be reflected by $(c_1, \mathit{paper}) \in \mathcal{F}$ and $(c_1, \mathit{article}) \in \mathcal{F}$, given that c_1 is the concept identifier of scientific articles. In the literature, one often defines a separate *lexicon* $\mathbb{L} = (L^C, \mathcal{F})$ and associates it with a core ontology [18]; but as it suffices for the context of this work, we assume the lexicon to be an integral part of the ontology itself for the sake of simplicity.

Folksonomies. As an alternative approach to taxonomies, collaborative tagging systems have gained a considerable amount of attention. Their underlying data structure is called *folksonomy*; according to [15], a folksonomy is a tuple $\mathbb{F} := (U, T, R, Y)$ where U, T, and R are finite sets, whose elements are called *users*, *tags* and *resources*, respectively. Y is a ternary relation between them, i.e. $Y \subseteq U \times T \times R$. An element $y \in Y$ is called a *tag assignment* or TAS. A *post* is a triple (u, T_{ur}, r) with $u \in U$, $r \in R$, and a non-empty set $T_{ur} := \{t \in T \mid (u, t, r) \in Y\}$. Intrinsically, concepts are not explicitly present within a folksonomy; however, the set of tags T contains lexical items similar to the vocabulary set L^C of a core ontology.

Term Graphs. Both core ontologies and folksonomies introduce various kinds of relations among the lexical items contained in them. A typical example are *tag cooccurrence networks*, which constitute an aggregation of the folksonomy structure indicating which tags have occurred together. Generally spoken, these *term graphs* $\mathbb{G}$ can be formalized as weighted undirected graphs $\mathbb{G} = (L, E, w)$ whereby L is a set of vertices (corresponding to lexical items), $E \subseteq L \times L$ model the edges and $w\colon E \to \mathbb{R}$ is a function which assigns a weight to the edges. As an example, given a folksonomy (U, T, R, Y), one can define the post-based[4] *tag-tag cooccurrence graph* as $\mathbb{G}_{cooc} = (T, E, w)$ whose set of vertices corresponds to the set T of tags. Two tags t_1 and t_2 are connected by an edge, iff there is at least one post (u, T_{ur}, r) with $t_1, t_2 \in T_{ur}$. The *weight* of this edge is given by the number of posts that contain both t_1 and t_2, i.e. $w(t_1, t_2) := \text{card}\{(u, r) \in U \times R \mid t_1, t_2 \in T_{ur}\}$

As we will define term abstractness measures based on core ontologies, folksonomies and term graphs, we will commonly refer to them as *term structures* $\mathbb{S}$ in the remainder of this paper. $L(\mathbb{S})$ is a projection on the set of lexical items contained in $\mathbb{S}$. Based on the above terminology, we now formally define a term abstractness measure in the following way:

Definition 1. *A* term abstractness measure $\sqsupset^{\mathbb{S}}$ *based upon a term structure* $\mathbb{S}$ *is a partial order among the lexical items L present in* $\mathbb{S}$*, i.e.* $\sqsupset^{\mathbb{S}} \subseteq L(\mathbb{S}) \times L(\mathbb{S})$*. If* $(l_1, l_2) \in \sqsupset^{\mathbb{S}}$ *(or* $l_1 \sqsupset^{\mathbb{S}} l_2$*) we say that* l_1 *is more abstract than* l_2*.*

In the following, we will make frequent use of *ranking functions* $r\colon L(\mathbb{S}) \to \mathbb{R}$ for lexical items in order to define a tag abstractness measure; please note that a ranking function corresponds to a partial order according to $(l_1, l_2) \in \sqsupset^{\mathbb{S}} \Leftrightarrow r(l_1) > r(l_2)$. We will denote the resulting term abstractness measure as $\sqsupset^{\mathbb{S}}_r$.

[4] Other possibilities are resource-based and user-based cooccurrence; we use post-based cooccurrence in the scope of this work as it is efficiently computable and captures a sufficient amount of information.

4 Measures of Tag Generality

Based on the notions defined above, we will now introduce a set of ranking functions r which are supposed to order lexical items within a folksonomy $\mathbb{F}$ by their degree of abstractness, inducing a partial order $\sqsupset_r^{\mathbb{F}}$ among the set of tags.[5] The measures are partially based on prior work in related areas, and build on different intuitions. One commonality they all share is that none of them considers the textual content of a tag itself (e.g. with linguistic methods). All measures operate solely on the folksonomy structure itself or on a derived term network, making them language-independent.

Frequency-based measures. A first natural intuition is that more abstract tags are simply used more often, because there exist more resources which they describe – as an example, the number of "computer"s in the world is much larger than the number of "notebook"s; hence one might assume that within a folksonomy, the tag "computer" is used more often than the tag "notebook". We capture this intuition in the abstractness measure $\sqsupset_{freq(t)}^{\mathbb{F}}$ induced by the ranking function *freq* which counts the number of tag assignments according to $freq(t) = \text{card}\{(u, t', r) \in Y : t = t'\}$

Entropy-based measures. Another intuition stems from information theory: Entropy measures the degree of uncertainty associated with a random variable. Considering the application of tags as a random process, one can expect that more general tags show a more even distribution, because they are probably used at a relatively constant level to annotate a broad spectrum of resources. Hence, more abstract terms will have a higher entropy. This approach was also used by Heymann [14] to capture the "generality" of tags in the context of tag recommendation. We adapt the notion from there and define

$$entr(t) = - \sum_{t' \in cooc(t)} p(t'|t) \log p(t'|t) \tag{1}$$

whereby $cooc(t)$ is the set of tags which cooccur with t, and $p(t'|t) = \frac{w(t',t)}{\sum_{t'' \in cooc(t)} w(t'',t)}$ (with $w(t',t)$ being the cooccurrence weight defined in Section 3). $entr(x)$ induces the term abstractness measure $\sqsupset_{entr}^{\mathbb{F}}$.

Centrality Measures. In network theory the centrality of a node $v \in V$ in a network G is usually an indication of how important the vertex is [24]. Applied to our problem at hand, centrality can also be contemplated as a measure of abstractness or generality, following the intuition that more abstract terms are also more "important". We adopted three standard centralities (degree, closeness, betweenness). All of them can be applied to a term graph $\mathbb{G}$, leaving us

[5] Note that all term abstractness measures based on real-value ranking functions are by construction total orders, but this is not mandatory.

with three measures $\sqsupset^{\mathbb{G}}_{dc}$, $\sqsupset^{\mathbb{G}}_{bc}$ and $\sqsupset^{\mathbb{G}}_{cc}$ as follows: *Degree centrality* simply counts the number of direct neighbors $d(v)$ of a vertex v in a graph $G = (V, E)$:

$$dc(v) = \frac{d(v)}{|V| - 1} \tag{2}$$

According to *betweenness centrality* a vertex has a high centrality if it can be found on many shortest paths between other vertex pairs:

$$bc(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{3}$$

Hereby σ_{st} denotes the number of shortest paths between s and t and $\sigma_{st}(v)$ is the number of shortest paths between s and t passing through v. As its computation is obviously very expensive, it is often approximated [4] by calculating the shortest paths only between a fraction of points. Finally, a vertex ranks higher according to *closeness centrality* the shorter its shortest path length to all other reachable nodes is:

$$cc(v) = \frac{1}{\sum_{t \in V \setminus v} d_G(v, t)} \tag{4}$$

$d_G(v, t)$ denotes hereby the geodesic distance (shortest path) between the vertices v and t.

Statistical Subsumption. Schmitz et.al. [22] applied a statistical *model of subsumption* between tags when trying to infer hierarchical relationships. It is based on the assumption that a tag t subsumes another tag t' if $p(t|t') > \xi$ and $p(t'|t) < \xi$ for a suitable threshold ξ. For measuring generality, the number of subsumed tags can be seen as an indicator of abstractness – the more tags a tag subsumes the more general it is:

$$subs(t) = \text{card}\{t' \in T : p(t|t') > \xi) \wedge p(t'|t) < \xi\} \tag{5}$$

5 Evaluation

In order to assess the quality of the tag abstractness measures $\sqsupset^{\mathbb{F}}_{freq}$, $\sqsupset^{\mathbb{F}}_{entr}$, $\sqsupset^{\mathbb{G}}_{dc}$, $\sqsupset^{\mathbb{G}}_{bc}$, $\sqsupset^{\mathbb{G}}_{cc}$ and $\sqsupset^{\mathbb{F}}_{subs}$ introduced above, a natural approach is to compare them against a ground truth. A suitable grounding should yield reliable judgments about the "true" abstractness of a given lexical item. Of special interest are hereby taxonomies and concept hierarchies, whose hierarchical structure typically contains more abstract terms like "entity" or "thing" close to the taxonomy root, whereby more concrete terms are found deeper in the hierarchy. Hence, we have chosen a set of established core ontologies and taxonomies, which cover each a rather broad spectrum of topics. They vary in their degree of controlledness – WordNet (see below) on the one hand being manually crafted by language experts, while the Wikipedia category hierarchy and DMOZ on the other hand are built in a much less controlled manner by a large number of motivated web users. In the following, we first briefly introduce each dataset; an overview about their statistical properties can be found in Table 1.

Table 1. Statistical properties of the datasets used in the evaluation

Core ontology	$\|C\|$	$\| \geq_C \|$	$\|L^C\|$	$\|\mathcal{F}\|$	$\| \sqsupseteq^{\mathbb{O}} \|$
WORDNET	79,690	81,866	141,391	141,692	2,028,925
YAGO	244,553	249,465	206,418	244,553	2,078,788
WIKI	2,445,974	4,447,010	2,445,974	2,445,974	13,171,439
DMOZ	767,019	767,019	241,910	767,019	5,210,226

Folksonomy	$\|U\|$	$\|T\|$	$\|R\|$	$\|Y\|$
DEL (Delicious)	667,128	2,454,546	18,782,132	140,333,714

Term Graphs	$\|T\|$	$\|E\|$
COOC	892,749	38,210,913
SIM	10,000	405,706

5.1 Grounding Datasets

WordNet [9] is a semantic lexicon of the English language. In WordNet, words are grouped into *synsets*, sets of synonyms that represent one concept. Among other relations, the *is-a* relation connects a *hyponym* (more specific synset) to a *hypernym* (more general synset). A synset can have multiple hypernyms, so that the graph is not a tree, but a directed acyclic graph. In order to allow for comparison with the other grounding datasets, we focussed on the noun subsumption network[6]. As it consists of several disconnected hierarchies, it is useful to add a fake top-level node subsuming all the roots of those hierarchies, making the graph fully connected and allowing a relative abstractness judgment between all contained pairs of nouns.

Yago [23] is a large ontology which was derived automatically from Wikipedia and WordNet. Manual evaluation studies have shown that its precision (i.e. the percentage of "correct" facts) lies around 95%. It has a much higher coverage than WordNet (see Table 1), because it also contains named entities like people, books or products. The complete ontology contains 1.7 million entities and 15 million relations; as our main interest lies in the taxonomy hierarchy, we restricted ourselves to the contained *is-a* relation[7] among concepts.

WikiTaxonomy [21] is the third dataset used for evaluation. This large scale domain independent taxonomy[8] was derived by evaluating the semantic network between Wikipedia concepts and labeling the relations as *isa* and *notisa*, using methods based on the connectivity of the network and on lexico-syntactic patterns. It contains by far the largest number of lexical items (see Table 1), but this comes at the cost of a much lower level of manual controlledness.

DMOZ[9] (also known as the open directory project or ODP) is an open content directory for links of the World Wide Web. Although it is hierarchically structured, it differs from the above-mentioned datasets insofar as its internal link structure does not always reflect a sub-concept/super-concept relationship. Despite this fact, we we included the DMOZ category hierarchy as a grounding

[6] `http://wordnet.princeton.edu/wordnet/download/` (v2.1)
[7] `http://www.mpi-inf.mpg.de/yago-naga/yago/subclassof.zip` (v2008-w40-2)
[8] `http://www.h-its.org/english/research/nlp/download/wikitaxonomy.php`
[9] `http://www.dmoz.org/`

dataset because it was built for a similar purpose like many collaborative bookmarking services (namely organizing WWW references). In addition, some of its top level categories (like “arts” or “business”) are described by rather abstract terms.

5.2 Tagging Dataset

In order to test the performance of our proposed term abstractness measures, we used a dataset crawled from the social bookmarking system Delicious in November 2006.[10] From the raw data, we first derived the *tag-tag cooccurrence graph* $COOC = (T', E_{cooc}, w_{cooc})$. Two tags t_1 and t_2 are connected by an edge, iff there is at least one post (u, T_{ur}, r) with $t_1, t_2 \in T_{ur}$. The edge weight is given by $w_{cooc}(t_1, t_2) := \text{card}\{(u, r) \in U \times R \mid t_1, t_2 \in T_{ur}\}$. In order to exclude cooccurrences introduced by chance and to enable an efficient computation of the centrality measures, we removed all tags from the resulting graph with a degree of less than 2.

In a similar way to [13], we also derived a *tag-tag similarity graph* $SIM = (T'', E_{sim}, w_{sim})$ by computing the Resource-Context-Similarity described in [5]. The latter is based on a frequency-based representation of tags in the vector space of all resources, in which similarity is computed by the cosine similarity. Because rarely used tags have very sparse vector representations, we restricted ourselves to the 10,000 most frequently used tags. Based on the resulting pairwise similarity values, we added an edge (t_1, t_2) to the edge list E_{sim} when the similarity was above a given threshold $min\_sim = 0.04$. This threshold was determined by inspecting the distribution of all similarity values. Table 1 summarizes the statistics of all tagging datasets.

Subsequently, we computed all term abstractness measures introduced in the previous chapter based on *DEL*, *COOC* and *SIM*, i.e. $\sqsupset_{freq}^{DEL}$, $\sqsupset_{entr}^{DEL}$, $\sqsupset_{dc}^{COOC}$, $\sqsupset_{bc}^{COOC}$, $\sqsupset_{cc}^{COOC}$, $\sqsupset_{bc}^{SIM}$, $\sqsupset_{cc}^{SIM}$ and $\sqsupset_{subs}^{\mathbb{F}}$.

5.3 Direct Evaluation Metric

As stated above, our grounding datasets contain information about concept subsumptions. If a concept c_1 subsumes concept c_2 (i.e. $(c_1, c_2) \in \geq_C$), we assume c_1 to be more abstract than c_2; as the taxonomic relation is transitive, we can infer $(c_1, c_2), (c_2, c_3) \in \geq_C \Rightarrow (c_1, c_3) \in \geq_C$ and hence that c_1 is also more abstract than c_3. In other words, thinking of the taxonomic relation as a directed graph, a given concept c is more abstract than all other concepts contained in the subgraph rooted at c. As we are interested in abstractness judgments about lexical items, we can consequently infer that concept labels for more abstract concepts are more abstract themselves. However, hereby we are facing the problem of polysemy: A given lexical item l can be used as a label for several concepts

[10] The data set is publicly available at `http://www.uni-koblenz-landau.de/koblenz/fb4/AGStaab/Research/DataSets/PINTSExperimentsDataSets/index_html`

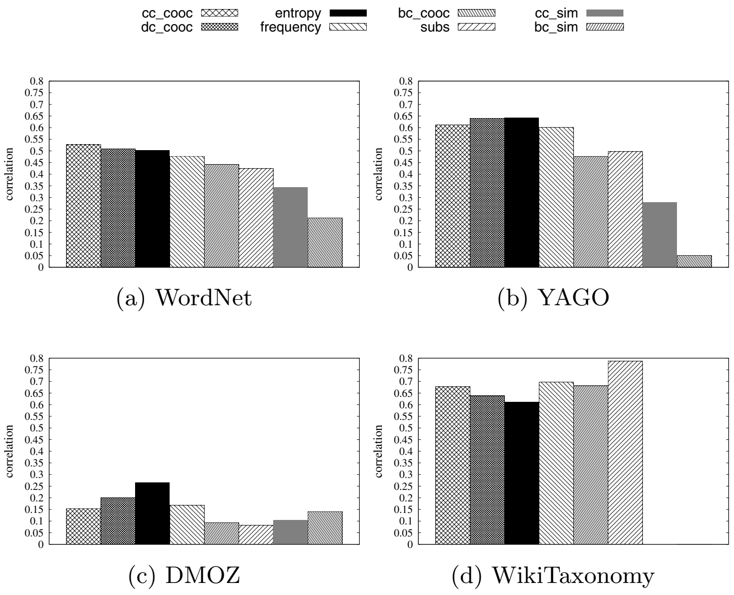


Fig. 1. Grounding of each introduced term abstractness measure $\sqsupset^{\mathbb{S}}$ against four ground-truth taxonomies. Each bar corresponds to a term abstractness measure; the y-axis depicts the gamma correlation as defined in Equation 7. (Values for `cc_sim` and `bc_sim` in (d) are -0.05 and -0.005, resp.).

of different abstractness levels. Consequently, l has "several" abstractness levels, depending in which context it is used. As a most simple approach, which removes possible effects of word sense disambiguation techniques, we "resolve" ambiguity in the following way: The abstractness measure $\sqsupset^{\mathbb{O}} \subseteq L^C \times L^C$ on the vocabulary of a core ontology $\mathbb{O}$ is constructed according to

$$(l_1, l_2) \in \sqsupset^{\mathbb{O}} \Leftrightarrow (c_1, l_1) \in \mathcal{F} \wedge (c_2, l_2) \in \mathcal{F} \wedge (c_1, c_2) \in \geq_C \tag{6}$$

whereby $\mathcal{F}$ is the label assigment relation defined in Section 3 . Due to the polysemy effect described above, $\sqsupset^{\mathbb{O}}$ is not necessarily a partial order, as it may contain cycles. But despite this fact, $\sqsupset^{\mathbb{O}}$ contains the complete information which terms $l_i \in L^C$ are more abstract than other terms $l_j \in L^C$ according to the taxonomy of $\mathbb{O}$. Hence we can use it as a "ground truth" to judge the quality of a given term abstractness measure $\sqsupset^{\mathbb{S}}$.

We are interested how well $\sqsupset^{\mathbb{O}}$ correlates to $\sqsupset^{\mathbb{S}}$; picking up the idea of the *gamma rank correlation* [6], we define *concordant* and *discordant* pairs between $\sqsupset^{\mathbb{S}}$ and $\sqsupset^{\mathbb{S}}$ as follows: a pair of terms l and k is called concordant w.r.t. two partial orderings $\sqsupset, \sqsupset_*$, if they agree on it, i.e. $(l \sqsupset k \wedge l \sqsupset_* k) \vee (k \sqsupset l \wedge k \sqsupset_* l)$. It is called discordant if they disagree, i.e. $(l \sqsupset k \wedge k \sqsupset_* l) \vee (k \sqsupset l \wedge l \sqsupset_* k)$.

Note that there may exist pairs which are neither concordant nor discordant. Based on these notions, the gamma rank correlation is defined as

$$CR(\sqsupset, \sqsupset_*) = \frac{|C| - |D|}{|C| + |D|} \tag{7}$$

whereby C and D denote the set of concordant and discordant pairs, respectively.

In our case, $\sqsupset_*$ is not a partial ordering, but only a relation – which means that in the worst case, a pair l, k can be concordant and discordant at the same time. As is obvious from the definition of the gamma correlation (see Eq. 7), such inconsistencies lead to a lower correlation. Hence, our proposed method of "resolving" term ambiguity by constructing $\sqsupset^{\mathbb{O}}$ according to Eq. 6 leads to a lower bound of correlation. Figure 1 summarizes the correlation of each of our analyzed measures, grounded against each of our ground truth taxonomies. First of all, one can observe that the correlation values between the different grounding datasets differ significantly. This is most obvious for the DMOZ hierarchy, where almost all measures perform only slightly better than random guessing. A slight exception is the entropy-based abstractness measure $\sqsupset^{\mathbb{F}}_{entropy}$, which in general gives greater than 0.25 across all datasets. Another relatively constant impression is that the centrality measures based on the tag similarity graph (*cc_sim* and *bc_sim*) show a smaller correlation than the other measures. The globally best correlations are found for the WikiTaxonomy dataset, namely by the subsumption-model-based measure *subs*. Apart from that, the centrality measures based on the tag cooccurrence graph and the frequency-based measure show a similar behavior.

5.4 Derived Measures

The grounding approach of the previous section gave a first impression of the ability of each measure to predict term abstractness judgments explicitly present in a given taxonomy. This methodology allowed only for an evaluation based on term pairs between which a connection exists in $\sqsupset^{\mathbb{O}}$, i.e. pairs where l_1 is either a predecessor or a successor of l_2 in the term subsumption hierarchy. However, our proposed measures make further distinctions among terms between which no connection exists within a taxonomy (e.g. the *freq* states that the most frequent term t is more abstract than *all* other terms). This phenomenon can probably also be found when asking humans – e.g. if one would ask which of the terms "art" or "theoretical computer science" is more abstract, most people will probably choose "art", even though both words are not connected by the is-a relation in (at least most) general-purpose taxonomies.

In order to extend our evaluation to these cases, we derived two straightforward measures from a taxonomy which allow for a comparison of the abstractness level between terms occurring in disconnected parts of the taxonomy graph. Because this approach goes beyond the explicitly encoded abstractness information, the question is justified to which extent it makes sense to compare the generality of completely unrelated terms, e.g. between "waterfall" and "chair".

Table 2. Results from the user study

Category	Number of classifications
One tag more general	41
Same level	11
Not comparable	154
Do not know one or two tags	3

Besides our own intuition, we are not aware of any reliable method to determine when humans perceive the abstractness of two terms as *comparable* or not. For this reason, we validated the derived measures – namely (i) the shortest path to the taxonomy root and (ii)the number of subordinate terms – by an experiment with human subjects.

Shortest path to taxonomy root. As stated above, most taxonomies are built in a top-down fashion, whereby more abstract terms are more likely to occur closer to the taxonomy root. Hence, a natural candidate for judging the abstractness of a term is to measure its distance to the root node. This corresponds to a ranking function $sp\_root(l)$, which ranks the terms l contained in a taxonomy in ascending order by the length of the shortest path between *root* and l.

Number of subordinate terms. Another measure is inspired by Kammann et al. [16], who stated that "*the abstractness of a word or a concept is determined by the number of subordinate words it embraces[...]*". Given a taxonomy $\mathbb{O}$ and its comprised term subsumption relation $\sqsupset^{\mathbb{O}}$, we can easily determine the number of "sub-terms" by $subgraph\_size(l) = \text{card}\{(l, l') \in \sqsupset^{\mathbb{O}}\}$. We are aware that this measure is strongly influenced e.g. by fast-evolving domains like e.g. "mobile computing", whose rapid growth along with a strong expansion of the included vocabulary might lead to an overestimation of its abstractness level. This is another motivating reason for the user study presented in the next paragraph.

Validation by user study. In order to check whether $sp\_root(l)$ and $subgraph\_size(l)$ correspond to human judgments of term abstractness, we performed an exemplary user study with 12 participants[11]. As a test set, we drew a random sample of 100 popular terms occurring in each of our datasets; for each term, we selected 3 candidate terms, taking into account cooccurrence information from the folksonomy *DEL*. The resulting 300 term pairs were shown to the each subject via a web interface[12], asking them to label the pair by one of 5 options (see Table 2)

We calculated Fleiss' κ [10] to get a closer look at the agreement of the study participants. In our experiment, $\kappa = 0.2836$ is indicating fair agreement. Table 2 shows the results of the number of classifications given that an agreement of 6 or more participants signalizes significant agreement. The relatively high number

[11] Students and staff from two IT departments.

[12] `http://www.kde.cs.uni-kassel.de/benz/generality_game.html`

Table 3. Accuracy of the taxonomy-derived abstractness measures

	Wordnet	Yago	DMOZ	WikiTaxonomy
sp_root	0.94	0.42	0.88	0.45
subgraph_size	0.94	0.96	0.8	0.87

of "not comparable" judgments show that even with our elaborate filtering, the task of differentiating abstractness levels is quite difficult. Despite this fact, our user study provided us with a well-agreed set of 41 term pairs, for which we got reliable abstractness judgments. Denoting these pairs as $\sqsupset_{manual}$, we can now check the accuracy of the term abstractness measures introduced by *sp_root* and *subgraph_size*, i.e. the percentage of correctly predicted pairs. Table 3 contains the resulting accuracy values. From our sample data, it seems that the subgraph size (i.e. the number of subordinate terms) is a more reliable predictor of human abstractness judgments. Hence, we will use it for a more detailed grounding of our folksonomy-based abstractness measures.

The ranking function *subgraph_size* naturally induces a partial order $\sqsupset^{\mathbb{O}}_{subgraph\_size}$ among the set of lexical items present in a core ontology $\mathbb{O}$. In order to check how close each of our introduced term abstractness measures correlate, we computed the *gamma correlation coefficient* [6] between the two partial orders (see Eq. 7). Figure 2 shows the resulting correlations. Again, the correlation level between the datasets differs, with DMOZ having the lowest values. This is consistent with the first evaluation based solely on the taxonomic relations (see Figure 1). Another consistent observation is that the measure based on the tag similarity network (*bc_sim* and *cc_sim*) show the weakest performance. The globally best value is found for the subsumption model, compared to the WikiTaxonomy (0.5); for the remaining conditions, almost all correlation values lie in the range between 0.25 and 0.4, and correlate hence weakly.

5.5 Discussion

Our primary goal during the evaluation was to check if folksonomy-based term abstractness measures are able to make reliable judgments about the relative abstractness level of terms. A first consistent observation is that measures based on frequency, entropy or centrality in the tag cooccurrence graph do exhibit a correlation to the abstractness information encoded in gold-standard-taxonomies. One exception is DMOZ, for which almost all measures exhibit only very weak correlation values. We attribute this to the fact that the semantics of the DMOZ topic hierarchy is much less precise compared to the other grounding datasets; as an example, the category `Top/Computers/Multimedia/Music_and_Audio/Software/Java` does hardly imply that `Software` "is a kind of" `Music_and_Audio`. WordNet on the contrary subsumes the term *Java* (among others) under taxonomically much more precise parents: `[...] > communication > language > artifical language > programming language > java` The same holds for Yago, and the WikiTaxonomy was also built with a strong focus on *is-a* relations [21]. This is actually an interesting observation: Despite the fact that both DMOZ and Delicious were built for similar

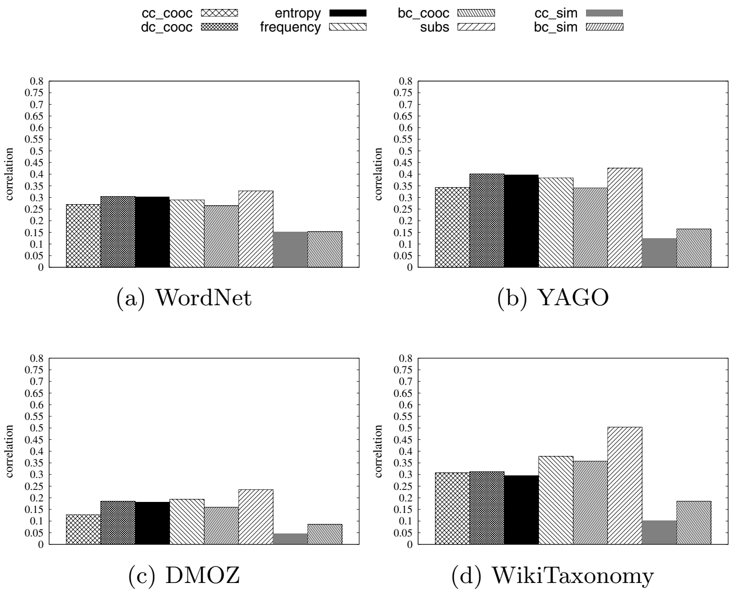


Fig. 2. Grounding of the term abstractness measure $\sqsupset^{\mathbb{S}}$ against $\sqsupset^{\mathbb{O}}_{subgraph\_size}$ derived from four ground-truth taxonomies. Each bar corresponds to a term abstractness measure; the y-axis depicts the gamma correlation as defined in Equation 7.

purposes (namely organizing WWW references), the implicit semantics within Delicious resembles more closely to well-established semantic repositories than to the bookmark-folder-inspired hierarchical organization scheme of DMOZ.

Another consistent observation is that abstractness measures based on tag similarity graphs (as used e.g. by [13]) perform worst through all experimental conditions. This is consistent with observations in our own prior work [5], where we showed that distributional similarity measures (like the one used in this paper or by [13]) induce connections preferably among tags having the same generality level. On the contrary, applying e.g. centrality measures to the "plain" tag cooccurrence graph yield better results. Hence, a justifiable conclusion is that tag-tag cooccurrence encodes a considerable amount of "taxonomic" information.

But this information is not solely present in the cooccurrence graph – also a probabilistic model of subsumption [22] yields good results in some conditions, especially when grounding against the taxonomy-derived *subgraph_size* ranking. We attribute this to the fact that both measures (the subsumption model and the subgraph size) are based on the same principle, namely that a term is more general the more other terms it subsumes.

Apart from that, even the simplest approach of measuring term abstractness by the mere frequency (i.e. the number of times a tag has been used) already exhibits a considerable correlation to our gold-standard taxonomies. This has an interesting application to the *popularity/generality problem*: Our results point in the direction that popular tags are on average more abstract (or more general) than less frequently used ones. In summary, the interpretation of our results can be condensed in two statements: First, folksonomy-based measures of term abstractness do exhibit a partially strong correlation to well-defined semantic repositories; and second, the abstractness level of a given tag can be approximated well by simple measures.

6 Conclusions

In this paper, we performed a systematic analysis of folksonomy-based term abstractness measures. To this end, we first provided a common terminology to subsume the notion of term abstractness in folksonomies and core ontologies. We then contributed a methodology to compare the abstractness information contained in each of our analyzed measures to established taxonomies, namely WordNet, Yago, DMOZ and the WikiTaxonomy. Our results suggest that centrality and entropy measures can differentiate well between abstract and concrete terms. In addition, we have provided evidence that the tag cooccurence graph is a more valuable input to centrality measures compared to tag similarity graphs in order to measure abstractness. Apart from that, we also shed light on the *tag generality vs. popularity* problem by showing that in fact, popularity seems to be a fairly good indicator of the "true" generality of a given tag. These insights are useful for all kinds of applications who benefit from a deeper understanding of tag semantics. As an example, tag recommendation engines could take generality information into account in order to improve their predictions, or folksonomy navigation facilities could offer a new direction of browsing towards more general or more specific directions. Finally, our results inform the design of algorithms geared towards making the implicit semantics in folksonomies explicit.

As next steps, we plan to apply our measures to identify generalists and specialists in social tagging systems. A possible hypothesis hereby is that specialists use a more specific vocabulary whereas generalists rely mainly on abstract tags.

Acknowledgments. We would like to thank Dr. Denis Helic and Beate Krause for fruitful discussions during the creation of this work. The research presented in this work is in part funded by the Know-Center, the FWF Austrian Science Fund Grant P20269, the European Commission as part of the FP7 Marie Curie IAPP project TEAM (grant no. 251514), the WebZubi project funded by BMBF and the VENUS project funded by Land Hessen.

References

1. Allen, R., Wu, Y.: Generality of texts. In: Digital Libraries: People, Knowledge, and Technology. LNCS, Springer, Heidelberg (2010)
2. Benz, D., Hotho, A., Stumme, G.: Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge. In: Proc. of WebSci 2010, Raleigh, NC, USA (2010)
3. Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., Zacharias, V.: KAON - Towards a Large Scale Semantic Web. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) EC-Web 2002. LNCS, vol. 2455, pp. 304–313. Springer, Heidelberg (2002)

4. Brandes, U., Pich, C.: Centrality estimation in large networks. I. J. Bifurcation and Chaos 17(7), 2303–2318 (2007)
5. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic analysis of tag similarity measures in collaborative tagging systems. In: Proc. of the 3rd Workshop on Ontology Learning and Population (OLP3), Patras, Greece, pp. 39–43 (July 2008)
6. Cheng, W., Rademaker, M., De Baets, B., Hüllermeier, E.: Predicting partial orders: Ranking with abstention. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS, vol. 6321, pp. 215–230. Springer, Heidelberg (2010)
7. Cimiano, P., Hotho, A., Staab, S.: Learning concept hierarchies from text corpora using formal concept analysis. Journal of Artificial Intelligence Research (JAIR) 24, 305–339 (2005)
8. Clark, J., Paivio, A.: Extensions of the Paivio, Yuille, and Madigan 1968 norms. Behavior Research Methods, Instruments, & Computers 36(3), 371 (2004)
9. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
10. Fleiss, J., et al.: Measuring nominal scale agreement among many raters. Psychological Bulletin 76(5), 378–382 (1971)
11. Golder, S., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science 32(2), 198–208 (2006)
12. Henschel, A., Woon, W.L., Wächter, T., Madnick, S.: Comparison of generality based algorithm variants for automatic taxonomy generation. In: Proc. of IIT 2009, pp. 206–210. IEEE Press, Piscataway (2009)
13. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Tech. Rep. 2006-10, CS dep. (April 2006)
14. Heymann, P., Ramage, D., Garcia-Molina, H.: Social tag prediction. In: SIGIR 2008: Proc. of the 31st Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 531–538. ACM, New York (2008)
15. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS(LNAI), vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
16. Kammann, R., Streeter, L.: Two meanings of word abstractness. Journal of Verbal Learning and Verbal Behavior 10(3), 303–306 (1971)
17. Körner, C., Benz, D., Hotho, A., Strohmaier, M., Stumme, G.: Stop thinking, start tagging: tag semantics emerge from collaborative verbosity. In: Proc. of WWW 2010, pp. 521–530. ACM, New York (2010)
18. Maedche, A.: Ontology Learning for the Semantic Web. Kluwer Academic Publishing, Boston (2002)
19. Mika, P.: Ontologies are us: A unified model of social networks and semantics. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 522–536. Springer, Heidelberg (2005)
20. Paivio, A., Yuille, J.C., Madigan, S.A.: Concreteness, imagery, and meaningfulness values for 925 nouns. Journal of Experimental Psychology 76 (1968)
21. Ponzetto, S.P., Strube, M.: Deriving a large-scale taxonomy from wikipedia. In: AAAI, pp. 1440–1445. AAAI Press, Menlo Park (2007)
22. Schmitz, P.: Inducing ontology from flickr tags (2006)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th international World Wide Web conference, WWW 2007 (2007)
24. Wasserman, S., Faust, K.: Social network analysis: Methods and applications. Cambridge Univ. Pr, Cambridge (1994)
25. Zhang, Q.: Fuzziness - vagueness - generality - ambiguity. Journal of Pragmatics 29(1), 13–31 (1998)

Semantic Enrichment of Twitter Posts for User Profile Construction on the Social Web

Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao

Web Information Systems, Delft University of Technology
{f.abel,q.gao,g.j.p.m.houben,k.tao}@tudelft.nl

Abstract. As the most popular microblogging platform, the vast amount of content on Twitter is constantly growing so that the retrieval of relevant information (streams) is becoming more and more difficult every day. Representing the semantics of individual Twitter activities and modeling the interests of Twitter users would allow for personalization and therewith countervail the information overload. Given the variety and recency of topics people discuss on Twitter, semantic user profiles generated from Twitter posts moreover promise to be beneficial for other applications on the Social Web as well. However, automatically inferring the semantic meaning of Twitter posts is a non-trivial problem.

In this paper we investigate semantic user modeling based on Twitter posts. We introduce and analyze methods for linking Twitter posts with related news articles in order to contextualize Twitter activities. We then propose and compare strategies that exploit the semantics extracted from both tweets and related news articles to represent individual Twitter activities in a semantically meaningful way. A large-scale evaluation validates the benefits of our approach and shows that our methods relate tweets to news articles with high precision and coverage, enrich the semantics of tweets clearly and have strong impact on the construction of semantic user profiles for the Social Web.

Keywords: semantic enrichment, twitter, user profile construction, news, linkage.

1 Introduction and Motivation

With the advent of social networking, tagging or microblogging that become tangible in Social Web systems like Facebook, Delicious and Twitter, a new culture of participation penetrates the Web. Today, more than 190 million people are using Twitter and together publish more than 65 million messages (*tweets*) per day[1]. Recent research shows that the exploitation of tweets allows for valuable applications such as earthquake warning systems [1], opinion mining [2] or discovery and ranking of fresh Web sites [3]. These applications mainly analyze and utilize the wisdom of the crowds as source of information rather than relying on individual tweets. Analogously, previous research in the field of microblogging

[1] http://techcrunch.com/2010/06/08/twitter-190-million-users/

studied information propagation patterns in the global Twitter network [4,5] and exploited network structures to identify influentual users [6,7] as well as malicious users [8,9]. While related work reveals several insights regarding the characteristics of the global Twitter network, there exists little research on understanding the semantics of individual microblogging activities and modeling individual users on Twitter with Semantic Web technologies.

Learning and modeling the semantics of individual Twitter activities is important because the amount of tweets published each day is continuously growing so that users need support to benefit from Twitter information streams. For example, given the huge amount of information streams available on Twitter, user profiling and personalization techniques that support users in selecting streams to follow or particular items to read are becoming crucial [6]. Further, given the variety and recency of topics people discuss on Twitter [5], user profiles that capture the semantics of individual tweets are becoming interesting for other applications on the Social Web as well. In order to provide personalization functionalities in Twitter and moreover enable Social Web applications to consume semantically meaningful representations of the users' Twitter activities, there is thus urgent need to research user modeling strategies that allow for the construction of user profiles with rich semantics.

In this paper we introduce approaches for enriching the semantics of Twitter posts and modeling users based on their microblogging activities. Given the realtime nature and news media characteristics of Twitter [5], we explore possibilities of linking Twitter posts with news articles from the Web. Therefore, we present and evaluate different strategies that link tweets with news articles and contextualize the semantics of tweets with semantics extracted from the corresponding news articles. Based on a large dataset of more than 3 million tweets published by more than 45,000 users, we compare the performance of different strategies for linking tweets with news and analyze their impact on user modeling.

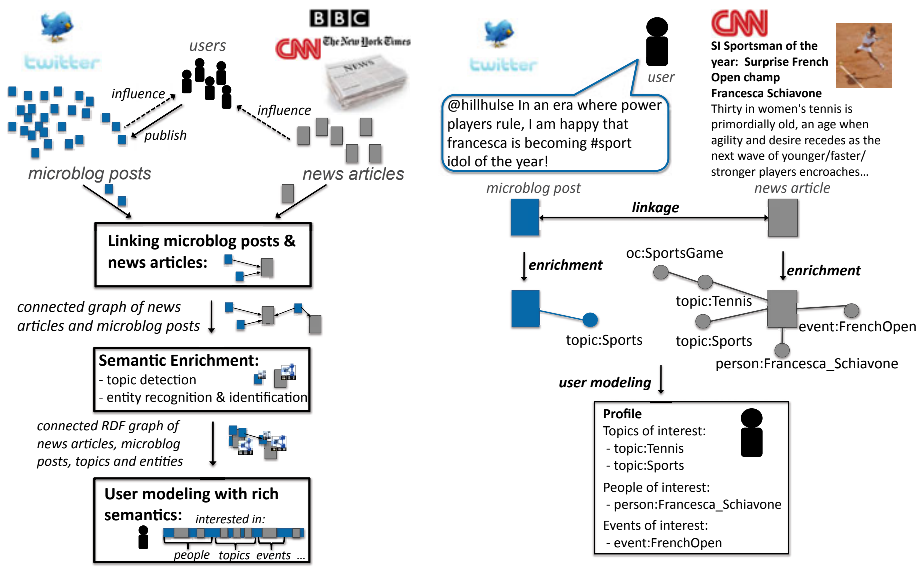
2 Related Work

Since Twitter was launched in 2007 research started to investigate the phenomenon of microblogging. Most research on Twitter investigates the network structure and properties of the Twitter network, e.g. [4,5,6,7]. Kwak et al. conducted a temporal analysis of trending topics in Twitter and discovered that over 85% of the tweets posted everyday are related to news [5]. They also show that hashtags are good indicators to detect events and trending topics. Huang et al. analyze the semantics of hashtags in more detail and reveal that tagging in Twitter rather used to join public discussions than organizing content for future retrieval [10]. Laniada and Mika [11] have defined metrics to characterize hashtags with respect to four dimensions: frequency, specificity, consistency, and stability over time. The combination of measures can help assessing hashtags as strong representative identifiers. Miles explored the retrieval of hashtags for recommendation purposes and introduced a method which considers user interests in a certain topic to find hashtags that are often applied to posts related to this topic [12]. In this paper, we compare hashtag-based methods with methods that extract and analyze the semantics of tweets. While SMOB [13], the semantic microblogging framework, enables users to explicitly attach semantic annotations (URIs) to their short messages by applying MOAT [14] and therewith allows for making the meaning of (hash)tags explicit, our ambition is to infer the semantics of individual Twitter activities automatically.

Research on information retrieval and personalization in Twitter focussed on ranking users and content. For example, Cha et al. [7] present an in-depth comparison of three measures of influence, in-degree, re-tweets, and mentions, to identify and rank influential users. Based on these measures, they also investigate the dynamics of user influence across topics and time. Weng et al. [6] focus on identifying influential users of microblogging services as well. They reveal that the presence of reciprocity can be explained by phenomenon of homophily, i.e. people who are similar are likely to follow each other. Content recommendations in Twitter aim at evaluating the importance of information for a given user and directing the user's attention to certain items. Anlei et al. [3] propose a method to use microblogging streams to detect fresh URLs mentioned in Twitter messages and compute rankings of these URLs. Chen et al. also focus on recommending URLs posted in Twitter messages and propose to structure the problem of content recommendations into three separate dimensions [15]: discovering the source of content, modeling the interests of the users to rank content and exploiting the social network structure to adjust the ranking according to the general popularity of the items. Chen et al. however do not investigate user modeling in detail, but represent users and their tweets by means of a bag of words, from which they remove stop-words. In this paper we go beyond bag-of-word representations and link tweets to news articles from which we extract entities to generate semantically more meaningful user profiles.

Interweaving traditional news media and social media is the goal of research projects such as SYNC3[2], which aims to enrich news events with opinions from the blogosphere. Twitris 2.0 [16] is a Semantic Web platform that connects event-related Twitter messages with other media such as YouTube videos and Google News. Using Twarql [17] for the detection of DBpedia entities and making the semantics of hashtags explicit (via *tagdef*[3]), it captures the semantics of major news events. TwitterStand [18] also analyzes the Twitter network to capture tweets that correspond to late breaking news. Such analyses on certain news events, such as the election in Iran 2009 [2] or the earthquake in Chile 2010 [19], have also been conducted by other related work. However, analyzing the feasibility of linking individual tweets with news articles for enriching and contextualizing the semantics of user activities on Twitter to generate valuable user profiles for the Social Web – which is the main contribution of this paper – has not been researched yet.

[2] http://www.sync3.eu
[3] http://tagdef.com


twitter
users
influence
publish
influence
microblog posts
news articles
Linking microblog posts & news articles:
connected graph of news articles and microblog posts
Semantic Enrichment:
- topic detection
- entity recognition & identification
connected RDF graph of news articles, microblog posts, topics and entities
User modeling with rich semantics: interested in:
people topics events ...
user
@hillhulse In an era where power players rule, I am happy that francesca is becoming #sport idol of the year!
SI Sportsman of the year: Surprise French Open champ Francesca Schiavone
Thirty in women's tennis is primordially old, an age when agility and desire recedes as the next wave of younger/faster/ stronger players encroaches...
microblog post
news article
linkage
enrichment
enrichment
oc:SportsGame
topic:Tennis
topic:Sports
topic:Sports
event:FrenchOpen
person:Francesca_Schiavone
user modeling
Profile
Topics of interest:
- topic:Tennis
- topic:Sports
People of interest:
- person:Francesca_Schiavone
Events of interest:
- event:FrenchOpen

(a) Generic solution (b) Example scenario

Fig. 1. Generic solution for semantic enrichment of tweets and user profile construction: (a) generic architecture and (b) example of processing tweets and news articles

3 How to Exploit Twitter for Semantic User Modeling?

The length of Twitter messages is limited to 140 characters which makes it difficult to detect the semantics of these messages. For example, posts such as "Interesting: http://bit.ly/iajV21 #politics" or "@nytimes this makes me scared" are even for humans difficult to understand without knowing the context. However, by following the links one can explore this context and grasp the semantics of the tweets. Many Twitter activities are related to news events. According to Kwak et al., more than 85% of the tweets in the Twitter network are related to news [5]. This observation motivates our idea of linking Twitter activities with news articles to automatically capture and enrich the semantics of microblogging activities. Such relations between tweets and news further allow for capturing user interests regarding trending topics and support applications that require recent user interests like recommender systems for news or other fresh items.

Figure 1(a) visualizes the components of our approach for constructing user profiles with rich semantics based on Twitter posts. We relate Twitter messages with news articles and exploit the content of both tweets and news articles to derive the semantics of the users' microblogging activities. Therefore, we aggregate individual posts of Twitter users as well as news articles published by mainstream media such as CNN, BBC, or New York Times and propose the following components.

Linkage. The challenge of linking tweets and news articles is to identify these articles a certain Twitter message refers to. Sometimes, users explicitly link to the corresponding Web sites, but often there is no hyperlink within a Twitter message which requires more advanced strategies. In Section 4 we introduce and evaluate different strategies that allow for the discovery of relations between tweets and news articles.

Semantic Enrichment. Given the content of tweets and news articles, another challenge is to extract valuable semantics from the textual content. Further, when processing news article Web sites an additional challenge is to extract the main content of the news article. While RSS facilitates aggregation of news articles, the main content of a news article is often not embedded within the RSS feed, but is available via the corresponding HTML-formatted Web site. These Web sites contain supplemental content (*boilerplate*) such as navigation menus, advertisements or comments provided by readers of the article. To extract the main content of news articles we use BoilerPipe [20], a library that applies linguistic rules to separate main content from the boilerplate.

In order to support user modeling and personalization it is important to – given the raw content of tweets and news articles – distill topics and extract entities users are concerned with. We therefore utilize Web services provided by OpenCalais[4], which allow for the extraction of entities such as people, organizations or events and moreover assign unique URIs to known entities and topics.

The connections between the semantically enriched news articles and Twitter posts enable us to construct a rich RDF graph that represents the microblogging activities in a semantically well-defined context.

User Modeling. Based on the RDF graph, which connects Twitter posts, news articles, related entities and topics, we introduce and analyze user modeling strategies that create semantically rich user profiles describing different facets of the users (see Section 5).

Figure 1(b) further illustrates our generic solution by means of an example taken from our dataset: a user is posting a message about the election of the sportsman of the year and states that she supports Francesca Schiavone, an Italian tennis player. The Twitter message itself just mentions the given name *francesca* and indicates with a hashtag (*#sport*) that this post is related to sports. Hence, given just the text from this Twitter message it is not possible to automatically infer that the user is concerned with the tennis player. Given our linkage strategies (see Section 4), one can relate the Twitter message with a corresponding news article published by CNN, which details on the SI sportsman election and Francesca Schiavone in particular. Entity and topic recognition reveal that the article is about tennis (*topic:Tennis*) and Schiavone's (*person:Francesca_Schiavone*) success at French Open (*event:FrenchOpen*) and therewith enrich the semantics which can be extracted from the Twitter message itself (*topic:Sports*).

[4] `http://www.opencalais.com`

4 Analyzing Linkage between Tweets and News for Semantic Enrichment

The key idea of our approach to enrich the semantics of Twitter messages is based on relating individual tweets to news articles so that semantics extracted from news articles can be applied to clarify the meaning of tweets. In this section we introduce different strategies for linking Twitter posts with news articles that provide details on these posts. To evaluate the impact of these strategies on the semantic enrichment of Twitter posts we conduct an analysis on a large dataset gathered from Twitter and major news publishing Web sites.

4.1 Strategies for Discovering Tweet-News Relations

The strategies, which we propose to find correlations between Twitter posts and external news resources, can be divided into URL-based strategies, which exploit interaction patterns and hyperlinks mentioned in tweets, and content-based strategies, which exploit the content of tweets and news articles. In the following definitions, T denotes the set of all tweets available in our dataset while N refers to the set of news articles.

URL-based Strategies. URLs (mostly short URLs shortened by services such as *bit.ly*) that are contained in tweets can be considered as indicators for news-related tweets. In particular, if a tweet contains a URL that points to an external news resource, there is a very high possibility that this tweet is closely related to the linked resource. Based on this principle we defined two URL-based strategies.

Definition 1 (Strict URL-based strategy). *If a Twitter post $t \in T$ contains at least one URL that is from certain mainstream news publishers and links to a news article $n \in N$, then we consider t and n as related: $(t, n) \in R_s$, where $R_s \subseteq T \times N$.*

For this strategy, we select BBC, CNN and the New York Times as the set of mainstream news publishers and apply URL-patterns to discover the corresponding tweets that point to these Web sites. A potential drawback of the strict URL-based strategy is that it will miss relevant relations for Twitter messages that contains no URL. For example, if a user replies to Twitter message that is according to the strict URL-based strategy related to a news article n then this reply message might be related to n as well. Based on this idea, we define a second URL-based strategy that is more flexible than the first one.

Definition 2 (Lenient URL-based strategy). *If a tweet $t_r \in T$ is a reply or re-tweet from another tweet $t \in T$, which contains at least one URL that is linked to a news article $n \in N$ authored by certain mainstream news publishers, then we consider both t_r and t as being related to n: $(t_r, n) \in R_l, (t, n) \in R_l$, where $R_l \subseteq T \times N$.*

Hence, the lenient URL-based strategy extends the strict strategy with tweets that were published as part of an interaction with a tweet that is according to the strict strategy news-related so that $R_s \subseteq R_l$.

Content based Strategies. As tweets do not necessarily contain a URL, we propse another set of strategies that exploit the content of tweets and news articles to connect tweets with news. For example, the Twitter post about Francesca Schiavone in Figure 1(b) should be linked to the corresponding news article even though the tweet does not have a URL directly pointing to the article. We thus propose three further strategies that analyze the content of Twitter posts to allow for linkage between Twitter activities and news articles.

Definition 3 (Bag-of-Words Strategy). *Formally, a Twitter post $t_j \in T$ can be represented by a vector $\boldsymbol{t} = (\alpha_1, \alpha_2..\alpha_m)$ where α_i is the frequency of a word i in t and m denotes the total number of words in t. Each news article $n \in N$ is also represented as a vector $\boldsymbol{n} = (\beta_1, \beta_2..\beta_k)$ where β_i is the frequency of a word i in the title of the news article n and k denotes the total number of words in n.*

The bag-of-word strategy relates a tweet t with the news article n, for which the $TF \times IDF$ score is maximized: $(t, n) \in R_b$, where $R_b \subseteq T \times N$.

The bag-of-words strategy thus compares a tweet t with every news article in N and chooses the most similar ones to build a relation between t and the corresponding article n. $TF \times IDF$ is applied to measure the similarity. Given a Twitter post t and a news article n, the term frequency TF_i of a term i (with $\alpha_i > 0$ in the vector representation of t) is β_i, i.e. the number of occurrences of the word i in n. And IDF_i, the inverse document frequency, is $IDF_i = 1 + log(\frac{|N|}{|\{n \in N : \beta_i > 0\}|+1})$, where $|\{n \in N : \beta_i > 0\}|$ is the number of news articles, in which the term i appears. Given TF and IDF, the similarity between t and n is calculated as follows.

$$sim(t, n) = \sum_{i=1}^{m} TF_i \cdot IDF_i \tag{1}$$

Given a ranking according to the above similarity measure, we select top ranked tweet-news pairs (t, n) as candidates for constructing a valid relation. Following the realtime nature of Twitter, we also add a temporal constraint to filter out these candidates, for which the publishing date of the Twitter message and news article differs more than two days.

The bag-of-words strategy treats all words in a Twitter post as equally important. However, in Twitter, hashtags can be considered as special words that are important features to characterize a tweet [11]. For news articles, some keywords such as person names, locations, topics, etc. are also good descriptors to characterize a news article. Conveying these observations, we introduce hashtag-based and entity-based strategies for discovering relations between tweets and news articles. These strategies follow the idea of the bag-of-words strategy (see Definition 3) and differ in the way of representing news articles and tweets.

Definition 4 (Hashtag-based strategy). *The hashtag-based strategy represents a Twitter post $t \in T$ via its hashtags: $\boldsymbol{h} = (\alpha_1, \alpha_2..\alpha_m)$, where α_i is the number of occurrences of a hashtag i in t and m denotes the total number of hashtags in t.*

The hashtag-based strategy relates a tweet t *(represented via its hashtags) with the news article* n*, for which the* $TF \times IDF$ *score is maximized:* $(t, n) \in R_h$*, where* $R_h \subseteq T \times N$*.*

While the hashtag-based strategy thus varies the style of representing Twitter messages, the entity-based strategy introduces a new approach for representing news articles.

Definition 5 (Entity-based strategy). *Twitter posts* $t \in T$ *are represented by a vector* $\boldsymbol{t} = (\alpha_1, \alpha_2..\alpha_m)$ *where* α_i *is the frequency of a word* i *in* t *and* m *denotes the total number of words in* t*. Each news article* $n \in N$ *is represented by means of a vector* $\boldsymbol{n} = (\beta_1, \beta_2..\beta_k)$*, where* β_i *is the frequency of an entity within the news article,* i *is the label of the entity and* k *denotes the total number of distinct entities in the news article* n*.*

The entity-based strategy relates the Twitter post t *(represented via bag-of-words) with the news article* n *(represented via the labels of entities mentioned in* n*), for which the* $TF \times IDF$ *score is maximized:* $(t, n) \in R_e$*, where* $R_e \subseteq T \times N$*.*

Entities are extracted by exploiting OpenCalais as described in Section 3. For the hashtag- and entity-based strategies, we thus use Equation 1 to generate a set of candidates of related tweet-news pairs and then filter out these pairs, which do not fulfill the temporal constraint that prescribes that the tweet and news article should be published within a time span of two days. Such temporal constraints may reduce the recall but have a positive effect on the precision as we will see in our analysis below.

4.2 Analysis and Evaluation

To analyze the impact of the strategies on semantic enrichment of Twitter posts, we evaluate the performance of the strategies with respect to coverage and precision based on a large data corpus which we crawled from Twitter and three major news media sites: BBC, CNN and New York Times.

Data Collection and Characteristics. Over a period of three weeks we crawled Twitter information streams via the Twitter streaming API. We started from a seed set of 56 Twitter accounts (U_n), which are maintained by people associated with one of the three mainstream news publishers, and gradually extended this so that we finally observed the Twitter activities of 48,927 extra users (U_u), who are not explicitly associated with BBC, CNN or the New York Times. The extension was done in a snowball manner: we added users to U_u, who interacted with another user $u \in U_n \cup U_u$. The 56 Twitter accounts closely related to mainstream news media enable publishers of news articles to discuss their articles and news events with the Twitter audience. For the 48,927 *casual users* we crawled all Twitter activities independently whether the activity was part of an interaction with the Twitter accounts of the mainstream news media or not. In total we thereby obtained more than 3.3 million tweets.

Figure 2 shows the number of tweets per user and depicts how often these users interacted with a Twitter account associated with mainstream media. The

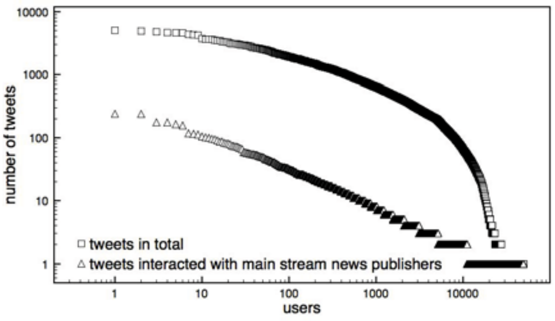


Fig. 2. Number of tweets per user $u \in U_u$ as well as the number of interactions (re-tweeting or reply activities) with Twitter accounts maintained by mainstream news media

distribution of the number of tweets per user shows a power-law-like distribution. For many users we recorded less than 10 Twitter activities within the three week observation period. Less than 500 users were highly active and published more than 1000 tweets. Further, the majority (more than 75%) of users interacted only once with a news-related user $u \in U_n$. We observed only nine users, who re-tweeted or replied to messages from news-related users more than 100 times and identified one of these users as spam user, who just joined the discussion to promote Web sites.

To connect tweets with news articles, we further crawled traditional news media. Each of the three mainstream news publishers (BBC, CNN, and New York Times) also provides a variety of news channels via their Web site. These news channels correspond to different news categories such as politics, sports, or culture and are made available via RSS feed. We constantly monitored 63 different RSS feeds from the corresponding news publishers and crawled the main content as well as supplemental metadata (title, author, publishing data, etc.) of more than 44,000 news articles.

Experimental Results. In order to evaluate the accuracy of the different strategies for relating tweets with news articles, we randomly selected tweet-news pairs that were correlated by a given strategy and judged the relatedness of the news article and the tweet message on a scale between 1 ("not related") and 4 ("perfect match"), where 2 means "not closely related" and 3 denotes "related" tweet-news pairs. For example, given a Twitter message about Francesca Schiavone's victory at French Open 2010, we considered news articles that report about this victory as "perfect match" while news articles about Francesca Schiavone, for which this victory is not the main topic but just mentioned as background information were considered as "related". In total, we (the authors of this paper) judged 1427 tweet-news pairs where each of the strategies depicted in Figure 3 was judged at least 200 times. For 85 pairs (5.96%) we were not able to decide whether the corresponding Twitter posts and the news article are related. We considered these pairs as "not related" and tweet-news relations, which were rated at least with 3 as truly related.

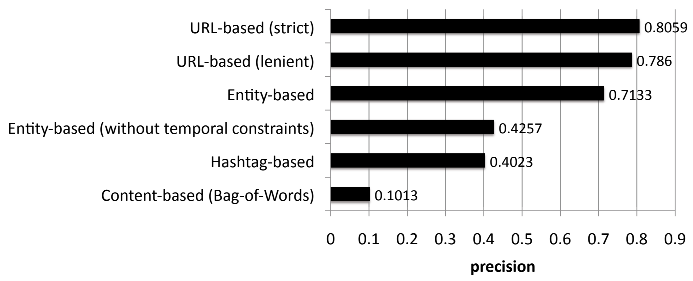


Fig. 3. Precision of different strategies for relating Twitter messages with news articles. (considered to refer to accurate).

Given this ground truth of correct tweet-news relations, we compare the precision of the different strategies, i.e. the fraction of correctly generated tweet-news relations. Figure 3 plots the results and shows that the URL-based strategies perform best with a precision of 80.59% (strict) and 78.8% (lenient) respectively. The naive content-based strategy, which utilizes the entire Twitter message (excluding stop-words) as search query and applies TFxIDF to rank the news articles, performs worst and is clearly outperformed by all other strategies. It is interesting to see that the entity-based strategy, which considers the publishing date of the Twitter message and news article, is nearly as good as the lenient URL-based strategy and clearly outperforms the hashtag-based strategy, which uses the temporal constraints as well. Even without considering temporal constraints, the entity-based strategy results in higher accuracy than the hashtag-based strategy. We conclude that the constellation/set of entities mentioned in a news article and Twitter message correspondingly, i.e. the number of shared entities, is a good indicator of relating tweets and news articles.

Figure 4 shows the coverage of the strategies, i.e. the number of tweets per user, for which the corresponding strategy found an appropriate news article. The URL-based strategies, which achieve the highest accuracy, are very restrictive: for less than 1000 users the number of tweets that are connected to news articles is higher than 10. The coverage of the lenient URL-based strategy is clearly higher than for the strict one, which can be explained by the number of interactions with Twitter accounts from mainstream news media (see Figure 2). The hashtag-based and entity-based strategies even allow for a far more higher number of tweet-news pairs. However, the hashtag-based strategy fails to relate tweets for more than 79% of the users, because most of these people do not make use of hashtags. By contrast, the entity-based strategy is applicable for the great majority of people and, given that it showed an accuracy of more than 70% can be considered as the most successful strategy.

Combining all strategies results in the highest coverage: for more than 20% of the users, the number of tweet-news relations is higher than 10. In the next section we will show that given these tweet-news relations we can create rich profiles that go beyond the variety of profiles, which are just constructed based on the tweets of the users.

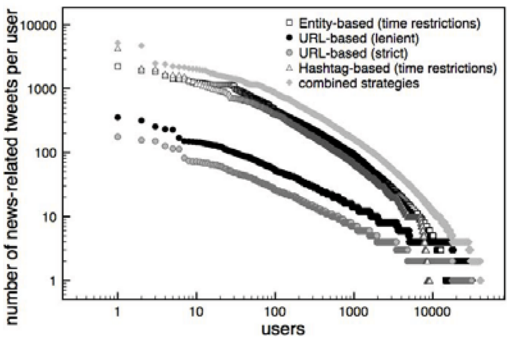


Fig. 4. Number of tweets per user, which are according to the different strategies related to news articles

5 Analyzing User Profile Construction Based on Semantic Enrichment

Based on the linkage of Twitter activities with news articles, we can exploit the semantics embodied in the news articles to create and enrich user profiles. In this section, we first present approaches for user modeling based on Twitter activities and then analyze the impact of exploiting related news articles for user profile construction in Twitter.

5.1 User Modeling Strategies

In this study we focus on two types of profiles: entity-based and topic-based profiles. An entity-based profile models a user's interests into a given set of entities such as persons, organizations, or events and can be defined as follows.

Definition 6 (Entity-based profile). *The* entity-based profile *of a user $u \in U$ is a set of weighted entities where the weight of an entity $e \in E$ is computed by a certain strategy w with respect to the given user u.*

$$P(u) = \{(e, w(u,e)) | e \in E, u \in U\} \tag{2}$$

$w(u,e)$ is the weight that is associated with an entity e for a given user u. E and U denote the set of entities and users respectively.

In Twitter, a naive strategy for computing a weight $w(u,e)$ is to count the number of u's tweets that refer to the given entity e. $|P(u)|$ depicts the number of distinct entities that appear in a profile $P(u)$. While entity-based profiles represent a user in a detailed and fine-grained fashion, topic-based profiles describe a user's interests into topics such as sports, politics or technology that can be specified analogously (see Definition 7).

Definition 7 (Topic-based profile). *The* topic-based profile $P_T(u)$ *of a user $u \in U$ is the restriction of an entity-based profile $P(u)$ to a set of topics $T \subseteq E$.*

From a technical point of view, both types of profiles specify the interest of a user into a certain URI, which represents an entity or topic respectively. Given the URI-based representation, the entity- and topic-based profiles become part of the Web of Linked Data and can therewith not only be applied for personalization purposes in Twitter (e.g., recommendations of tweet messages or information streams to follow) but in in other systems as well. For the construction of entity- and topic-based profiles we compare the following two strategies.

Tweet-based. The tweet-based baseline strategy constructs entity- and topic-based user profiles by considering only the Twitter messages posted by a user, i.e. the first step of our user modeling approach depicted in Figure 1(a) is omitted so that tweets are not linked to news articles. Entities and topics are directly extracted from tweets using OpenCalais. The weight of an entity corresponds to the number of tweets, from which an entity was successfully extracted, and the weight of a topic corresponds to the number of tweets, which were categorized with the given topic.

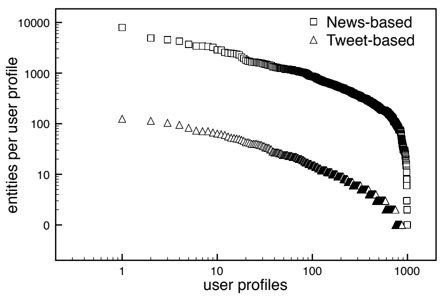
News-based. The news-based user modeling strategy applies the full pipeline of our architecture for constructing the user profiles (see Figure 1(a)). Twitter messages are linked to news articles by combining the URL-based and entity-based (with temporal restrictions) strategies introduced in Section 4 and entities and topics are extracted from the news articles, which have been linked with the Twitter activities of the given user. The weights correspond again to the number of Twitter activities which relate to an entity and topic respectively.

Our hypothesis is that the news-based user modeling strategy, which benefits from the linkage of Twitter messages with news articles, creates more valuable profiles than the tweet-based strategy.

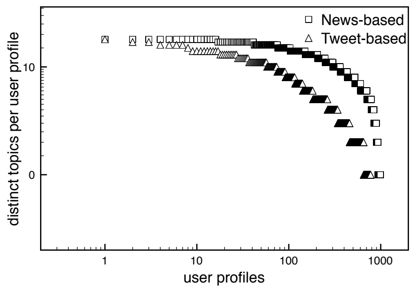
5.2 Analysis and Evaluation

To validate our hypothesis we randomly selected 1000 users (from U_u) and applied both strategies to create semantic user profiles from their Twitter activities. Figure 5 compares the number of distinct entities and topics available in the corresponding profiles ($|P(u)|$). Even though the number of Twitter activities, which can be linked to news articles, is smaller than the total number of Twitter activities of a user (cf. Fig. 2, Fig. 4), the number of entities and topics available in the profiles generated via the news-based strategy is higher than for the tweet-based approach. Regarding the entity-based profiles this difference is higher than for the topic-based profiles, because each Twitter message and news article is usually categorized with one topic at most whereas for the number of entities there is no such limit. News articles provide much more background information (a higher number of entities) than Twitter messages and thus allow for the construction of more detailed entity-based user profiles.

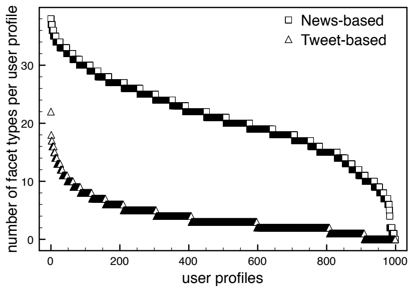
Further, the variety of the entity-based profiles generated via the news-based strategy is much higher than for the tweet-based strategy as depicted in Figure 5(c). For the tweet-based strategy, more than 50% of the profiles contain



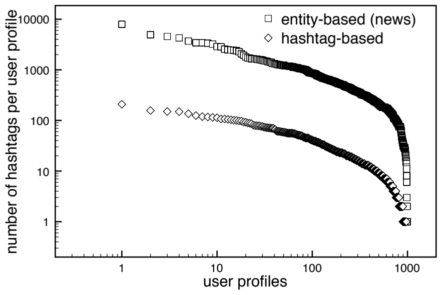
(a) Entity-based profiles



(b) Topic-based profiles



(c) User profile facets



(d) Hashtag- vs. entity-based profiles

Fig. 5. Comparison between tweet-based and news-based user modeling strategies: (a) for creating entity-based profiles and (b) topic-based profiles, (c) with respect to the variety of facet types available in the user profiles (example facet types: person, event, location, product). Further, (d) hashtag-based vs. entity-based profiles: number of distinct hash tags and entities per profile.

just less than four types of entities (mostly persons and organizations) while for the news-based strategy more than 50% of the profiles reveal interests in more than 20 types of entities. For example, they show that users are – in addition to persons or organizations – also concerned with certain events or products. The news-based strategy, i.e. the complete user construction pipeline proposed in Figure 1, thus allows for the construction of profiles that cover different facets of interests which increases the number of applications that can be built on top of our user modeling approaches (e.g., product recommendations).

Related research stresses the role of hashtags for being valuable descriptors [11,12,10]. However, a comparison between hashtag-based profiles and entity-based profiles created via the news-based strategy shows that for user modeling on Twitter, hashtags seem to be a less valuable source of information. Figure 5(d) reveals that the number of distinct hashtags available in the corresponding user profiles is much smaller than the number of distinct entities that are discovered with our strategy, which relates Twitter messages with news articles. Given that each named entity as well as each topic of an entity- and topic-based user profile has a URI, the semantic expressiveness of profiles generated with the news-based user modeling strategy is much higher than for the hashtag-based profiles.

6 Conclusions and Future Work

In this article, we introduced and analyzed strategies that enrich the semantics of microblogging activities for creating semantically rich user profiles on the Social Web. We present different strategies that connect Twitter messages with related news articles and exploit semantics extracted from news articles to deduce and contextualize the semantic meaning of individual Twitter posts. Our evaluation on a large Twitter dataset (more than 3 million tweets posted by more than 45,000 users) showed that, given the name of entities mentioned in a news article (such as persons or organizations) as well as the temporal context of the article, we can relate tweets and news articles with high precision (more than 70%) and high coverage (approx. 15% of the tweets can be linked to news articles). Our analysis further revealed that the exploitation of tweet-news relation has significant impact on user modeling and allows for the construction of more meaningful profiles (more profile facets and more detailed knowledge regarding user interests/concerns) than user modeling based on tweets only.

Semantic enrichment of Twitter user activities based on semantics extracted from news articles thus leads to meaningful representations of Twitter activities, ready for being applied in Twitter and other Social Web systems. In our ongoing research, we would deepen the investigation of how the profiles constructed by this type of user modeling strategies impact personalization on the Social Web[5]. Given the variety and recency of the constructed profiles, there are different applications worthwhile to explore such as Twitter stream and message recommendations, product recommendations or recommending news.

Acknowledgements. This work is partially sponsored by the EU FP7 projects ImREAL (http://imreal-project.eu) and GRAPPLE (http://grapple-project.org).

References

1. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proc. of 19th Int. Conf. on World Wide Web, pp. 851–860. ACM, New York (2010)
2. Gaffney, D.: #iranElection: quantifying online activism. In: Proc. of the WebSci10: Extending the Frontiers of Society On-Line (2010)
3. Dong, A., Zhang, R., Kolari, P., Bai, J., Diaz, F., Chang, Y., Zheng, Z., Zha, H.: Time is of the essence: improving recency ranking using Twitter data. In: Proc. of 19th Int. Conf. on World Wide Web, pp. 331–340. ACM, New York (2010)
4. Lerman, K., Ghosh, R.: Information contagion: an empirical study of spread of news on Digg and Twitter social networks. In: Cohen, W.W., Gosling, S. (eds.) Proc. of 4th Int. Conf. on Weblogs and Social Media. AAAI Press, Menlo Park (2010)
5. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Proc. of the 19th Int. Conf. on World Wide Web, pp. 591–600. ACM, New York (2010)

[5] Code and further results: `http://wis.ewi.tudelft.nl/umap2011/`

6. Weng, J., Lim, E.P., Jiang, J., He, Q.: TwitterRank: finding topic-sensitive influential Twitterers. In: Davison, B.D., Suel, T., Craswell, N., Liu, B. (eds.) Proc. of 3rd ACM Int. Conf. on Web Search and Data Mining, pp. 261–270. ACM, New York (2010)
7. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring user influence in twitter: The million follower fallacy. In: Cohen, W.W., Gosling, S. (eds.) Proc. of 4th Int. Conf. on Weblogs and Social Media. AAAI Press, Menlo Park (2010)
8. Lee, K., Caverlee, J., Webb, S.: The social honeypot project: protecting online communities from spammers. In: Proc. of 19th Int. Conf. on World Wide Web, pp. 1139–1140. ACM, New York (2010)
9. Lee, K., Caverlee, J., Webb, S.: Uncovering social spammers: social honeypots + machine learning. In: Proc. of 33rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 435–442. ACM, New York (2010)
10. Huang, J., Thornton, K.M., Efthimiadis, E.N.: Conversational tagging in twitter. In: Proc. of 21st Conf. on Hypertext and Hypermedia, pp. 173–178. ACM, New York (2010)
11. Laniado, D., Mika, P.: Making sense of twitter. 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. 470–485. Springer, Heidelberg (2010)
12. Efron, M.: Hashtag retrieval in a microblogging environment. In: Proc. of 33rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 787–788. ACM, New York (2010)
13. Passant, A., Hastrup, T., Bojars, U., Breslin, J.: Microblogging: A Semantic Web and Distributed Approach. In: Bizer, C., Auer, S., Grimnes, G.A., Heath, T. (eds.) Proc. of 4th Workshop Scripting For the Semantic Web (SFSW 2008) co-located with ESWC 2008, vol. 368 (2008), CEUR-WS.org
14. Passant, A., Laublet, P.: Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. In: Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW 2008), Beijing, China (2008)
15. Chen, J., Nairn, R., Nelson, L., Bernstein, M., Chi, E.: Short and tweet: experiments on recommending content from information streams. In: Proc. of 28th Int. Conf. on Human Factors in Computing Systems, pp. 1185–1194. ACM, New York (2010)
16. Jadhav, A., Purohit, H., Kapanipathi, P., Ananthram, P., Ranabahu, A., Nguyen, V., Mendes, P.N., Smith, A.G., Cooney, M., Sheth, A.: Twitris 2.0: Semantically empowered system for understanding perceptions from social data. In: Proc. of the Int. Semantic Web Challenge (2010)
17. Mendes, P.N., Passant, A., Kapanipathi, P.: Twarql: tapping into the wisdom of the crowd. In: Proc. of the 6th International Conference on Semantic Systems, pp. 45:1–45:3. ACM, New York (2010)
18. Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M.D., Sperling, J.: Twitterstand: news in tweets. In: Proc. of 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 42–51. ACM, New York (2009)
19. Mendoza, M., Poblete, B., Castillo, C.: Twitter Under Crisis: Can we trust what we RT? In: Proc. of 1st Workshop on Social Media Analytics (SOMA 2010). ACM Press, New York (2010)
20. Kohlschütter, C., Fankhauser, P., Nejdl, W.: Boilerplate detection using shallow text features. In: Proc. of 3rd ACM Int. Conf. on Web Search and Data Mining, pp. 441–450. ACM, New York (2010)

Improving Categorisation in Social Media Using Hyperlinks to Structured Data Sources*

Sheila Kinsella[1], Mengjiao Wang[1], John G. Breslin[1,2], and Conor Hayes[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
firstname.lastname@deri.org
[2] School of Engineering and Informatics, National University of Ireland, Galway
john.breslin@nuigalway.ie

Abstract. Social media presents unique challenges for topic classification, including the brevity of posts, the informal nature of conversations, and the frequent reliance on external hyperlinks to give context to a conversation. In this paper we investigate the usefulness of these external hyperlinks for categorising the topic of individual posts. We focus our analysis on objects that have related metadata available on the Web, either via APIs or as Linked Data. Our experiments show that the inclusion of metadata from hyperlinked objects in addition to the original post content significantly improved classifier performance on two disparate datasets. We found that including selected metadata from APIs and Linked Data gave better results than including text from HTML pages. We investigate how this improvement varies across different topics. We also make use of the structure of the data to compare the usefulness of different types of external metadata for topic classification in a social media dataset.

Keywords: social media, hyperlinks, text classification, Linked Data, metadata.

1 Introduction

Social media such as blogs, discussion forums, micro-blogging services and social-networking sites have grown significantly in popularity in recent years. By lowering the barriers to online communication, social media enables users to easily access and share content, news, opinions and information in general. Recent research has investigated how microblogging services such as Twitter enable real-time, first-hand reporting of news events [15] and how question-answering sites such as Yahoo! Answers allow users to ask questions on any topic and receive community-evaluated answers [1]. Social media sites like these are generating huge amounts of user-generated content and are becoming a valuable source of information for the average Web user.

* The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

However, navigating this wealth of information can be challenging. Posts are unstructured with little or no metadata and are usually much shorter than a typical Web document. Due to the casual environment and minimal curation in social media sites, the quality is highly variable [1]. Much of the information contained in social media is generated during conversations between people who have a shared context that allows them to communicate without explicitly stating all relevant information. In many cases, vital pieces of information are provided not within the text of the post, but behind hyperlinks that users post to refer to a relevant resource. For example, a poster may recommend a book by posting a hyperlink to a webpage where you can buy it, rather than using the traditional method of providing the book title and name of the author. In the message board dataset described in Section 4, we found that 65% of posts that linked to books mentioned neither the complete title nor the complete author name, and at least 11% did not contain even a partial title or author name.

Such hyperlinks to external resources are a potential source of additional information for information retrieval in online conversations. Hyperlinks to sources of structured data are particularly promising because the most relevant metadata can be extracted and associated with the original post content. The resulting rich representations of posts can be used for enhanced search and classification. Recently, there has been a growing amount of structured information available on the Web. Many of the most popular websites such as Amazon and YouTube provide developer APIs that can be used to programmatically access metadata about resources, and there is also a growing amount of RDFa and Linked Data being published. The Linking Open Data [5] project in particular has resulted in many structured datasets from diverse domains becoming available on the Web, some of which we use as data sources in our experiments.

In this paper, we focus on the task of improving categorisation in social media using metadata from external hyperlinks, building on initial experiments reported in [14]. To ensure that our conclusions are valid across different websites, we investigate datasets from two disparate types of social media, a discussion forum and a micro-blogging website. We compare the results of topic classification based on post content, on text from hyperlinked HTML documents, on metadata from external hyperlinks, and on combinations of these. Our experiments show that incorporating metadata from hyperlinks can significantly improve the accuracy of categorisation of social media items. We make use of the structure of the data to empirically evaluate which metadata types are most useful for categorisation. We also investigate how the results vary by topic, in order to determine the circumstances where this approach would add the most benefit. Our results demonstrate that thanks to the linked nature of the Web, structured data can be useful to improve classification even in non-structured data.

2 Related Work

The enhancement of Web documents with external information for information retrieval is a long-established technique, an early example being Google's use of anchor text for Web search [17]. Previous studies in the field of Web document categorisation have proven that the classification of webpages can be boosted by taking into account the text of neighbouring webpages ([2], [16]). Our work differs in that we focus on social media and rather than incorporating entire webpages or parts of webpages, we include specific metadata items that have a semantic relation to the objects discussed in a post. This approach enables us to compare the effectiveness of different metadata types for improving classification.

Other work has looked at classifying particular types of Web objects using metadata. Our work is related to that of Figueiredo et al. [8], who assess the quality of various textual features in Web 2.0 sites such as YouTube for classifying objects within that site. They do not use any external data. Yin et al. [21] propose improving object classification within a website by bridging heterogeneous objects so that category information can be propagated from one domain to another. They improve the classification of Amazon products by learning from the tags of HTML documents contained within ODP[1] categories.

There is previous work on classifying social media using the metadata of the post itself. Berendt and Hanser [4] investigated automatic domain classification of blog posts with different combinations of body, tags and title. Sun et al. [19] showed that blog topic classification can be improved by including tags and descriptions. Our work differs from these because we use metadata from objects on the Web to describe a social media post that links to those objects.

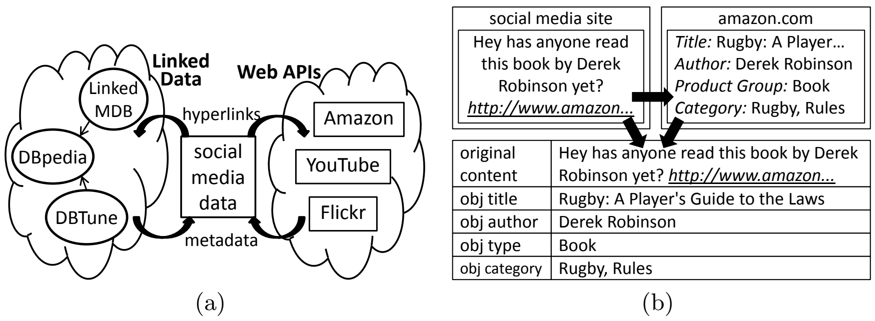
There has also been related work in the area of classifying Twitter messages. Garcia Esparza et al. [9] investigated tweet categorisation based on content. Jansen et al. [12] classified posts relating to consumer brands according to their sentiment. Irani et al. [11] studied the problem of identifying posters who aim to dishonestly gain visibility by misleadingly tagging posts with popular topics. They build models that correspond to topics in order to identify messages that are tagged with a topic but are in fact spam. They take advantage of hyperlinks by augmenting their models with text from webpages linked to within posts. Our work differs in that we focus on the potential offered by structured Web data and show that extracting relevant metadata gives superior results to using entire webpages. In the Semantic Web domain, Stankovic et al. [18] proposed a method for mapping conference-related posts to their corresponding talks and then to relevant DBpedia topics, enabling the posts to be more easily searched.

A relevant study that used structured data from hyperlinks in social media was performed by Cha et al. [7] who used hyperlinks in a blog dataset to study information propagation in the blogosphere. They downloaded the metadata of YouTube videos and analysed the popularity of categories, the age distribution of videos, and the diffusion patterns of different categories of videos.

3 Enhanced Post Representations

Hyperlinks are often an integral part of online conversations. Users share videos or photos they have seen, point to products or movies they are interested in,

[1] Open Directory Project, `http://www.dmoz.org/`, accessed March 2011.



original content	Hey has anyone read this book by Derek Robinson yet? http://www.amazon...
obj title	Rugby: A Player's Guide to the Laws
obj author	Derek Robinson
obj type	Book
obj category	Rugby, Rules

Fig. 1. (a) Web sources that were used to enrich social media data; (b) example of a how a social media post can be enhanced with external structured data relating to it

and use external articles as references in discussions. These external resources can provide useful new data such as author information in the case of books, or genre information in the case of movies. Many of these hyperlinks are to websites that publish metadata about objects, such as videos (YouTube) or products (Amazon), and make this metadata available via an API or as Linked Data. These websites are particularly useful since they allow particular pieces of relevant data to be identified and extracted, along with their relationship to the hyperlinked resource. In some cases the metadata is published by external sources, *e.g.*, DBpedia [3] provides a structured representation of Wikipedia. Figure 1(a) gives a graphical representation of how a social media dataset can be enhanced with information from various Web sources. The sources shown are those that will be used in experiments later in this paper.

Figure 1(b) gives an example of a post where additional useful information can be gained by considering the metadata of a hyperlinked object. Some of the information such as the author is redundant, but the title of the book and the categories are new. Both the title and the categories can be of use for classifying this post, which was posted in a Rugby forum. The name of the book could be useful for information retrieval, for example in a search scenario where a user queries for a book title. It is likely that certain metadata types will generally be more useful than others - for example, while the name of a book's publisher may sometimes indicate the topic of a book, in many cases it will not be helpful.

In order to integrate information from structured data sources in our post representations we make use of the vector space model so that we can re-use established methods that operate on vector models. A document d_i is represented as a vector of terms $d_i = \{w_{i,1}, w_{i,2}...w_{i,t}\}$ where $w_{i,j}$ denotes the weight of term j in document i. Functions such as tf-idf and document length normalisation can be applied to reduce the impact of variations in term frequency and document length. For each post, we create one feature vector based on the text from the original post, another based on hyperlinked HTML documents, and another based on hyperlinked object metadata. We experiment with different ways of combining the text sources into unified feature vectors.

4 Data Corpus

In our experiments, we use datasets originating from two different types of social media: *Forum*, from an online discussion forum, and *Twitter*, from the microblogging site[2]. We examined the domains linked to in each dataset and identified the most common sources of structured data. We extracted the posts that contained hyperlinks to these sources, and for each hyperlink we retrieved the related metadata as well as the corresponding HTML page. An identical pre-processing step was applied to each text source - post content, metadata and HTML documents. All text was lower-cased, non-alphabetic characters were omitted and stopwords were removed. Table 1 shows the average number of unique tokens remaining per post for each text source after preprocessing. The discrepancy in lengths of metadata and HTML between datasets is due to differences in the distribution of domains linked to in each dataset. Wikipedia articles, for example, tend to have particularly long HTML documents.

We now describe each stage of the data collection in more detail. The process of metadata collection for the forum dataset is described further in [13].

***Forum* dataset.** We use the corpus from the 2008 `boards.ie` SIOC Data Competition[3], which covers ten years of discussion forum posts represented in the SIOC (Semantically-Interlinked Online Communities) format [6]. Each post belongs to a thread, or conversation, and each thread belongs to a forum, which typically covers one particular area of interest. Our analysis considers only the posts contained in the final year of the dataset, since the more recent posts contain more links to structured data sources. From the most common domains in the dataset we identified MySpace, IMDB and Wikipedia as sources of Linked Data, via third-party data publishers detailed later in this section. We identified Amazon, YouTube and Flickr as sources of metadata via APIs. We use forum titles as categories for the classification experiments, since authors generally choose a forum to post in according to the topic of the post. We selected ten forums for these experiments based on the criteria that they were among the most popular forums in the dataset and they each have a clear topic (as opposed to general "chat" forums). The percentage of posts that have hyperlinks varies between forums, from 4% in Poker to 14% in Musicians, with an average of 8% across forums. These are a minority of posts; however, we believe they are worth focusing on because the presence of a hyperlink often indicates that the post is a

Table 1. Average unique tokens from each text source (± standard deviation)

Dataset	Content	Metadata	HTML
Forum	37.8 ± 42.6	19.6 ± 26.0	597.0 ± 659.1
Twitter	5.9 ± 2.6	26.9 ± 28.1	399.7 ± 302.9

[2] `http://twitter.com/`, accessed March 2011.

[3] `http://data.sioc-project.org/`, accessed March 2011.

useful source of information rather than just chat. Of the posts with hyperlinks, we focus on the 23% that link to one or more of the structured data sources listed previously. For the 23% of posts that have a title, this is included as part of the post content. Since discussion forums are typically already categorised, performing topic classification is not usually necessary. However, this data is representative of the short, informal discussion systems that are increasingly found on Web 2.0 sites, so the results obtained from utilising the class labels in this dataset should be applicable to similar uncategorised social media sites.

***Twitter* dataset.** The Twitter dataset[4] comes from Yang and Leskovec [20], and covers 476 million posts from June 2009 to December 2009. Twitter is a microblogging site that allows users to post 140 character status messages (tweets) to other users who subscribe to their updates. Due to the post length restriction, Twitter users make frequent use of URL shortening services such as bit.ly[5] to substantially shorten URLs in order to save space. Therefore for this dataset it was necessary to first decode short URLs via cURL[6]. From the most common domains we identified Amazon, YouTube and Flickr as sources of metadata via APIs. Like many social media websites, but in contrast to the previous dataset, Twitter does not provide a formal method for categorising tweets. However, a convention has evolved among users to tag updates with topics using words or phrases prefixed by a hash symbol (#). We make use of these hashtags to create six categories for classification experiments. Our approach borrows the hashtag-to-category mappings method from Esparza et al. [9] to identify tweets that relate to selected categories. We reuse and extend the hashtag categories of [9]; Table 2 shows the mappings between hashtags and categories. These categories were chosen because they occur with a high frequency in the dataset and they have a concrete topic. Tweets belonging to more than one category were omitted, since our goal is to assign items to a single category. All hashtags were removed from tweets, including those that do not feature in Table 2, since they may also contain category information. Any URLs to websites other than the selected metadata sources were eliminated from tweets. Finally, to avoid repeated posts caused by users retweeting (resending another post), all retweets were omitted.

External metadata. Amazon product, Flickr photo and YouTube video metadata was retrieved from the respective APIs. MySpace music artist information was obtained from DBTune[7] (an RDF wrapper of various musical sources including MySpace), IMDB movie information from LinkedMDB[8] (a movie dataset with links to IMDB) and Wikipedia article information from DBpedia[9]. The latter three services are part of the Linking Open Data project [5]. The number

[4] `http://snap.stanford.edu/data/twitter7.html`, accessed March 2011.
[5] `http://bit.ly`, accessed March 2011.
[6] `http://curl.haxx.se/`, accessed March 2011.
[7] `http://dbtune.org/`, accessed March 2011.
[8] `http://linkedmdb.org/`, accessed March 2011.
[9] `http://dbpedia.org/`, accessed March 2011.

Table 2. Categories and corresponding hashtags in the *Twitter* dataset

Category	#hashtags
Books	book, books, comic, comics, bookreview, reading, readingnow, literature
Games	game, pcgames, videogames, gaming, gamer, xbox, psp, wii
Movies	movie, movies, film, films, cinema
Photography	photography, photo
Politics	politics
Sports	nfl, sports, sport, football, f1, fitness, nba, golf

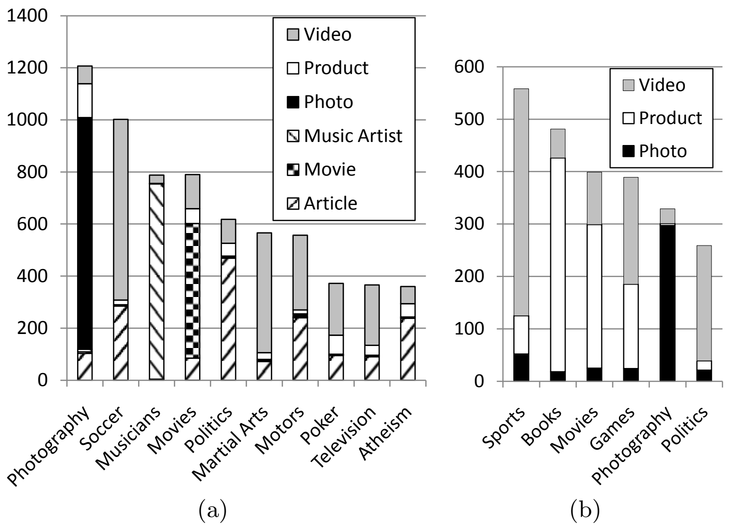


Fig. 2. No. of posts containing links to each type of object for (a) *Forum*, (b) *Twitter*

of posts containing links to each type of object in the *Forum* dataset is shown in Figure 2(a), and the number of posts containing links to each type of object for *Twitter* is shown in Figure 2(b). For the *Forum* dataset, hyperlinks to music artists occur mainly in the Musicians forum, movies in the Films forum, and photos in the Photography forum. The other object types are spread more evenly between the remaining seven forums. In total, *Forum* contains 6,626 posts and *Twitter* contains 2,415 posts. Note that in rare cases in *Forum*, a post contains links to multiple object types, in which case that post is included twice in a column. Therefore the total counts in Figure 2(a) are inflated by approximately 1%. For our analysis, we select only the most commonly available metadata types in order to make comparisons between them, but our method could be applied using arbitrary metadata. The metadata types that we chose were Title, Category (includes music/movie genre), Description (includes Wikipedia abstract), Tags and Author/Director (for Amazon books and IMDB movies only).

HTML documents. We crawled the corresponding HTML document for each hyperlink remaining in the datasets. For any cases where a HTML document was not retrievable, this object was removed from the dataset. We stripped out HTML tags and retained only the visible text of the webpage.

5 Analysis of the External Metadata

We now investigate some features of the metadata that was collected for the *Forum* dataset. Statistics are not reported for *Twitter* due to space constraints. Note that this analysis was performed after pre-processing the metadata text.

The first section of Table 3 shows the percentage of non-empty metadata for each type of object. This is of interest since a metadata type that occurs rarely will have limited usefulness. Due to the unique features of each website, not every object type can have every metadata type. There are large variations in the percentage of non-empty features for different metadata types. Titles are typically essential to identify an object and categories are typically required by a website's browsing interface, so these features are almost always present. For user-generated content, the frequency of non-empty fields is depends on whether the field is mandatory. For example, tags are often absent in Flickr because they are optional, while for videos they are almost always present because in the absence of user-provided tags, YouTube automatically assigns tags. For products, the author feature is often empty since this field is only available for books. For movies, the director feature is sometimes empty, presumably due to some inconsistencies in the various sources from which LinkedMDB integrates data.

The second section of Table 3 shows the average number of unique tokens found in non-empty metadata fields. These figures are an indicator of how much information each feature provides. In general, titles and authors/directors provide few tokens since they are quite short. For categories, the number of tokens depends on whether the website allows multiple categories (*e.g.*, Wikipedia) or single categories (*e.g.*, YouTube). The number of unique tokens obtained from descriptions and tags are quite similar across all object types studied.

The third section of Table 3 gives the average percentage of unique tokens from metadata that do not occur in post content. This section is important since it shows which features tend to provide novel information. Note that for article titles, the percentage is zero since all titles are contained within the article's URL. For music artist titles, the figure is low since bands often use their title as their username, which is contained within the artist's URL. All other object types have URLs that are independent of the object properties. This section also allows us to see how users typically describe an object. For example, 40% of the tokens from product titles are novel, indicating that posters often do not precisely name the products that they link to. For the subset of products that are books, 23% of tokens from titles were novel. Approximately 32% of the tokens from book authors and 43% of the tokens from movie directors are novel, showing that posters often mention these names in their posts, but that in many other cases this is new information which can aid retrieval.

Table 3. Properties of external metadata content for *Forum*

	Title	**Category**	**Description**	**Tags**	**Author/ Director**
Average % of text features that are non-empty after pre-processing					
Article	100.0	100.0	99.7	-	-
Movie	100.0	100.0	-	-	39.9
Music Artist	99.7	100.0	-	-	-
Photo	100.0	-	58.8	84.9	-
Product	100.0	100.0	-	75.2	65.4
Video	100.0	100.0	99.5	99.5	-
Average unique metadata tokens for non-empty fields (± standard deviation)					
Article	2.1 ± 0.9	13.6 ± 12.1	15.8 ± 8.3	-	-
Movie	1.7 ± 0.7	4.1 ± 1.8	-	-	2.2 ± 0.6
Music Artist	1.8 ± 0.9	2.7 ± 0.9	-	-	-
Photo	2.0 ± 1.1	-	10.9 ± 17.2	6.5 ± 4.9	-
Product	5.2 ± 3.0	11.5 ± 7.8	-	5.7 ± 2.1	2.0 ± 0.4
Video	3.7 ± 1.6	1.0 ± 0.0	13.1 ± 26.3	7.2 ± 5.0	-
Average % of unique metadata tokens that are novel (do not occur in post content)					
Article	0.0	78.5	68.4	-	-
Movie	17.4	76.2	-	-	43.3
Music Artist	10.1	85.4	-	-	-
Photo	72.5	-	50.3	74.6	-
Product	39.5	81.0	-	51.1	32.2
Video	62.0	95.7	78.5	74.4	-

6 Classification Experiments

In this section, we evaluate the classification of posts in the *Forum* and *Twitter* datasets, based on different post representations including the original text augmented with external metadata.

6.1 Experimental Setup

For each post, the following representations were derived, in order to compare their usefulness as sources of features for topic classification:

Content (without URLs): Original post content with hyperlinks removed.
Content: The full original content with hyperlinks intact.
HTML: The text parsed from the HTML document(s) to which a post links.
Metadata: The external metadata retrieved from the hyperlinks of the post.

Document length normalisation and tf-idf weighting were applied to each feature vector. We also generate aggregate feature vectors for the combinations of Content+HTML and Content+Metadata. An aggregate vector for two text sources is obtained by adding their individual feature vectors, after document length normalisation and tf-idf weighting. We tested two methods for combining different sources of textual information into a single vector:

Bag of words: The same term in different sources is represented by the same element in the document vector. For these experiments, we test different weightings of the two sources, specifically {0.1:0.9, 0.2:0.8, ... , 0.9:0.1}. Two vectors v_1 and v_2 are combined into a single vector v where a term i in v is given by, for example, $v[i] = (v_1[i] \times 0.1) + (v_2[i] \times 0.9)$.

Concatenate: The same term in different sources is represented by different elements in the feature vector - *i.e.*, "music" appearing in a post is distinct from "music" in a HTML page. Two vectors v_1 and v_2 are combined into a single vector v via concatenation, *i.e.*, $v = \langle v_1, v_2 \rangle$.

Classification of documents was performed with the Multinomial Naïve Bayes classifier implemented in Weka [10]. A ten-fold cross validation was used to assess the performance of the classifier on each type of document representation. K-fold cross validation involves randomly splitting a dataset into K folds, and using one fold as test data and the remaining $K - 1$ folds as training data. The process is repeated so that each of the K folds is used as a test set exactly once. Duplication of hyperlinks across splits was disallowed, so the metadata of a particular object cannot occur in multiple folds. In order to avoid duplication of post content due to one post quoting another, *Forum* was split by thread so that multiple posts from one thread do not occur in separate folds. Duplication was not an issue in *Twitter* since retweets had been removed. These restrictions resulted in the omission of approximately 11% of the *Forum* posts from any fold.

6.2 Experimental Results

The accuracy of classification for each representation is measured using the F_1 measure, which takes into account both precision p and recall r and is defined as $F_1 = \frac{2.p.r}{p+r}$. Micro-averaged F_1 is calculated by averaging F_1 over each test instance and is therefore more heavily influenced by common categories, while macro-averaged F_1 is calculated by averaging F_1 over the result for each category and is therefore more heavily influenced by rare categories.

The results of the classification experiments for each post representation are shown in Table 4 with their 90% confidence intervals. For both datasets, classification results based on content improve when tokens from URLs within posts are included. Classification using only the HTML pages linked to by posts gives relatively poor results, while classification using only metadata from hyperlinked objects improves accuracy for *Forum*, but decreases accuracy for *Twitter*. Those differences are all statistically significant. For the combined representations, the bag-of-words representation gives slightly better results than concatenation. The results reported are for the best-performing weightings. For *Forum*, these were 0.9:0.1 for Content+HTML and 0.5:0.5 for Content+Metadata. For *Twitter*, these were 0.9:0.1 for Content+HTML and 0.8:0.2 for Content+Metadata. For both HTML and Metadata, a bag-of-words combination with Content outperforms results for Content alone. The Content+Metadata approach significantly outperforms the Content+HTML approach, for both datasets.

Table 4. Micro-averaged F_1 for ($\pm$ 90% Confidence Interval)

Data Source	*Forum*		*Twitter*	
	Bag of Words	Concatenate	Bag of Words	Concatenate
Content (without URLs)	0.745 ± 0.009	-	0.722 ± 0.019	-
Content	0.811 ± 0.008	-	0.759 ± 0.015	-
HTML	0.730 ± 0.007	-	0.645 ± 0.020	-
Metadata	0.835 ± 0.009	-	0.683 ± 0.018	-
Content+HTML	0.832 ± 0.007	0.795 ± 0.004	0.784 ± 0.016	0.728 ± 0.016
Content+Metadata	0.899 ± 0.005	0.899 ± 0.005	0.820 ± 0.013	0.804 ± 0.018

Table 5. F_1 achieved by classifier for each category, ordered by performance

Forum				*Twitter*			
Forum	Content	Metadata	Content +M'data	Forum	Content	Metadata	Content +M'data
Musicians	*0.973*	0.911	0.981	Books	0.804	*0.836*	0.877
Photography	*0.922*	0.844	0.953	Photography	*0.785*	0.728	0.842
Soccer	0.805	*0.902*	0.945	Games	*0.772*	0.675	0.830
Martial Arts	0.788	*0.881*	0.917	Movies	0.718	*0.777*	0.827
Motors	0.740	*0.869*	0.911	Sports	*0.744*	0.563	0.781
Movies	0.825	*0.845*	0.881	Politics	*0.685*	0.499	0.733
Politics	*0.791*	0.776	0.846				
Poker	0.646	*0.757*	0.823				
Atheism	*0.756*	0.732	0.821				
Television	0.559	*0.664*	0.716				
Macro-Avgd	0.781	0.818	0.879	Macro-Avgd	0.751	0.680	0.815

Table 5 shows the detailed results for each category, for Content, Metadata and Content+Metadata (using the bag-of-words weighting with the best performance). There is a large variation in classification results for different categories. For post classification based on Content, *Forum* results vary from 0.973 down to 0.559 and *Twitter* results vary from 0.804 down to 0.685. Despite the variation between categories, Content+Metadata always results in the best performance. For the two single source representations, some categories obtain better results using Content and others using Metadata. The higher result between these two representations is highlighted with italics.

Table 6 shows the gains in accuracy achieved by performing classification based on different types of metadata from Wikipedia articles and YouTube videos, for the *Forum* dataset. We limit our analysis to these object types because they have consistently good coverage across all of the forums, apart from Musicians which we excluded from this analysis. These results are based only on the posts with links to objects that have non-empty content for every metadata type and amount to 1,623 posts for Wikipedia articles and 2,027 posts for YouTube videos. We compare the results against Content (without URLs), because Wikipedia URLs contain article titles and our aim is to measure the

Table 6. Micro-averaged F_1 for classification based on selected metadata types in *Forum* (± 90% Confidence Interval)

Metadata Type	Content (w/o URLs)	Metadata Only	Content+Metadata
Wikipedia Articles			
Category	0.761 ± 0.014	0.811 ± 0.012	0.851 ± 0.009
Description		0.798 ± 0.016	0.850 ± 0.009
Title		0.685 ± 0.016	0.809 ± 0.011
YouTube Videos			
Tag	0.709 ±0.011	0.838 ± 0.019	0.864 ± 0.012
Title		0.773 ± 0.015	0.824 ± 0.013
Description		0.752 ± 0.010	0.810 ± 0.013
Category		0.514 ± 0.017	0.753 ± 0.014

effects of the inclusion of titles and other metadata. Table 6 shows that the results for different metadata types vary considerably. For posts containing links to Wikipedia articles, the article categories alone result in a better classification of the post's topic than the original post content, with an F_1 of 0.811 compared to 0.761. Likewise, for posts that contain links to YouTube videos, the video tags provide a much better indicator of the post topic than the actual post text. The Content+Metadata column shows results where each metadata type was combined with post content (without URLs), using a bag-of-words representation with 0.5:0.5 weightings. Every metadata type examined improved post classification relative to the post content alone. However some metadata types improve the results significantly more than others, with Content+Category achieving the best scores for articles, and Content+Tags achieving the best scores for videos.

7 Discussion

The usage of external information from hyperlinks for categorisation or retrieval on the Web is a well-established technique. Our experiments show that categorisation of social media posts can be improved by making use of semantically-rich data sources where the most relevant data items can be experimentally identified. Both datasets showed similar patterns, although the *Twitter* scores are consistently lower. It may be that the Twitter hashtags are not as accurate descriptors of topic as the forum categories. Also, for *Forum* the external metadata is a better indicator of the category than the post content while for *Twitter* the reverse is true. This may be partially due to the fact that the distribution of domains linked to in each dataset is different and some domains may provide more useful information than others, either within URLs or within metadata.

We also observe that results vary considerably depending on the topic that is under discussion. For example in *Forum*, classification of a post in the Musicians forum is trivial, since almost all posts that feature a link to MySpace belong here. In contrast, the classification of a Television forum post is much more challenging, because this forum mentions a wide variety of topics which are televised. We also note that some topics achieve better classification results using only external metadata but others have better results with the original content. In the case of the Musicians and Photography forums, the good results for Content may be due to the fact that links to MySpace are highly indicative of the Musicians forum, and links to Flickr are usually from the Photography forum. The Politics and Atheism forums also achieve better results based on post content - this may be because they have a high percentage of links to Wikipedia articles, whose URLs include title information. We can conclude for posts whose hyperlinks contain such useful indicators, the addition of external metadata may give only a slight improvement, but for posts whose URLs do not give such explicit clues, the addition of external metadata can be an important advantage for topic classification.

A major benefit of using structured data rather than HTML documents is that it becomes possible to compare the improvements gained by integrating different metadata types. Our results show that the effect of the addition of different metadata types varies greatly, *e.g.*, Wikipedia categories and descriptions are much more useful than article titles. The benefit of different metadata types is not consistent across sites - Wikipedia's rich categories are far more useful than YouTube's limited categories. Often particular metadata types from hyperlinked objects in a post can be a better descriptor of the post topic than the post itself, for example YouTube tags, titles and descriptions. In these cases the structure of the data could be exploited to highly weight the most relevant metadata types. Thus, even classification on unstructured Web content can immediately benefit from semantically-rich data, provided that there are hyperlinks to some of the many websites that do provide structured data. While this paper focused on commonly-available metadata types, our approach could be applied to arbitrary metadata types from unknown sources, where machine-learning techniques would be employed to automatically select and weight the most useful metadata.

In our experiments, we used the structure of the external data to identify which types of metadata provide the most useful texts for improving classification. In addition to providing metadata, the Linked Data sources are also part of a rich interconnected graph with semantic links between related entities. We have shown that the textual information associated with resources can improve categorisation, and it would be interesting to also make use of the semantic links between concepts. For example, imagine a Television post contains links to the series `dbpedia:Fawlty_Towers`. A later post that links to the series `dbpedia:Mr_Bean` could be classified under the same category, due to the fact that the concepts are linked in several ways, including their genres and the fact that they are both produced by British television channels. Just as we used machine-learning techniques to identify the most beneficial metadata types, we could also identify the most useful properties between entities.

Potential applications for our approach include categorisation of either new or existing post items. For example, on a multi-forum site (*i.e.*, one that contains a hierarchy of individual forums categorised according to topic area), a user may not know the best forum where they should make their post, or where it is most likely to receive comments that are useful to the user. This can be the case where the content is relevant to not just one forum topic but to multiple topic areas. On post creation, the system could use previous metadata-augmented posts and any links if present in the new post to suggest potential categories for this post. Similarly, posts that have already been created but are not receiving many comments could be compared against existing augmented posts to determine if they should be located in a different topic area than they are already in.

This approach also has potential usage across different platforms. While it may be difficult to use augmented posts from Twitter to aid with categorisation of posts on forums due to the differing natures of microblogs and discussion forums, there could be use cases where augmented posts from discussion forums, news groups or mailing lists (*e.g.*, as provided via Google Groups) could be used to help categorisations across these heterogeneous, yet similar, platforms. Also, the categories from augmented discussion forum posts could be used to recommend tags or topics for new blog content at post creation time.

8 Conclusion

In this work, we have investigated the potential of using metadata from hyperlinked objects for classifying the topic of posts in online forums and microblogs. The approach could also be applied to other types of social media. Our experiments show that post categorisation based on a combination of content and object metadata gives significantly better results than categorisation based on either content alone or content and hyperlinked HTML documents. We observed that the significance of the improvement obtained from including external metadata varies by topic, depending on the properties of the URLs that tend to occur within that category. We also found that different metadata types vary in their usefulness for post classification, and some types of object metadata are even more useful for topic classification than the actual content of the post. We conclude that for posts that contain hyperlinks to structured data sources, the semantically-rich descriptions of entities can be a valuable resource for post classification. The enriched structured representation of a post as content plus object metadata also has potential for improving search in social media.

References

1. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: 1st Int'l Conference on Web Search and Data Mining, WSDM 2008. ACM, New York (2008)
2. Angelova, R., Weikum, G.: Graph-based text classification: Learn from your neighbors. In: 29th Int'l SIGIR Conference on Research and Development in Information Retrieval. SIGIR 2006. ACM, New York (2006)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: Aberer, K., 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. 722–735. Springer, Heidelberg (2007)

4. Berendt, B., Hanser, C.: Tags are not metadata, but "just more content"–to some people. In: 5th Int'l Conference on Weblogs and Social Media, ICWSM 2007 (2007)
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The story so far. International Journal on Semantic Web and Information Systems 5(3) (2009)
6. Breslin, J.G., Harth, A., Bojars, U., Decker, S.: Towards semantically-interlinked online communities. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 500–514. Springer, Heidelberg (2005)
7. Cha, M., Pérez, J., Haddadi, H.: Flash Floods and Ripples: The spread of media content through the blogosphere. In: 3rd Int'l Conference on Weblogs and Social Media, ICWSM 2009 (2009)
8. Figueiredo, F., Belém, F., Pinto, H., Almeida, J., Gonçalves, M., Fernandes, D., Moura, E., Cristo, M.: Evidence of quality of textual features on the Web 2.0. In: 18th Conference on Information and Knowledge Management, CIKM 2009. ACM, New York (2009)
9. Garcia Esparza, S., O'Mahony, M.P., Smyth, B.: Towards tagging and categorization for micro-blogs. In: 21st National Conference on Artificial Intelligence and Cognitive Science, AICS 2010 (2010)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: An update. ACM SIGKDD Exp. 11(1) (2009)
11. Irani, D., Webb, S., Pu, C., Li, K.: Study of trend-stuffing on Twitter through text classification. In: 7th Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS 2010 (2010)
12. Jansen, B.J., Zhang, M., Sobel, K., Chowdury, A.: Twitter power: Tweets as electronic word of mouth. J. Am. Soc. Inf. Sci. 60(11) (2009)
13. Kinsella, S., Passant, A., Breslin, J.G.: Using hyperlinks to enrich message board content with Linked Data. In: 6th Int'l Conference on Semantic Systems, I-SEMANTICS 2010. ACM, New York (2010)
14. Kinsella, S., Passant, A., Breslin, J.G.: Topic classification in social media using metadata from hyperlinked objects. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Murdock, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 201–206. Springer, Heidelberg (2011)
15. Mendoza, M., Poblete, B., Castillo, C.: Twitter under crisis: Can we trust what we RT? In: 1st Workshop on Social Media Analytics, SOMA 2010. ACM, New York (2010)
16. Qi, X., Davison, B.: Classifiers without borders: Incorporating fielded text from neighboring web pages. In: 31st Int'l SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008. ACM, New York (2008)
17. Sergey, B., Lawrence, P.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30(1-7), 107–117 (1998)
18. Stankovic, M., Rowe, M., Laublet, P.: Mapping tweets to conference talks: a goldmine for semantics. In: 3rd Int'l Workshop on Social Data on the Web, SDoW 2010 (2010), CEUR-WS.org
19. Sun, A., Suryanto, M.A., Liu, Y.: Blog classification using tags: An empirical study. In: Goh, D.H.-L., Cao, T.H., Sølvberg, I.T., Rasmussen, E. (eds.) ICADL 2007. LNCS, vol. 4822, pp. 307–316. Springer, Heidelberg (2007)
20. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: Fourth Int'l Conference on Web Search and Data Mining, WSDM 2011. ACM, New York (2011)
21. Yin, Z., Li, R., Mei, Q., Han, J.: Exploring social tagging graph for web object classification. In: 15th SIGKDD Int'l Conference on Knowledge Discovery and Data Mining, KDD 2009. ACM, New York (2009)

Predicting Discussions on the Social Semantic Web

Matthew Rowe, Sofia Angeletou, and Harith Alani

Knowledge Media Institute, The Open University, Milton Keynes, United Kingdom
{m.c.rowe,s.angeletou,h.alani}@open.ac.uk

Abstract. Social Web platforms are quickly becoming the natural place for people to engage in discussing current events, topics, and policies. Analysing such discussions is of high value to analysts who are interested in assessing up-to-the-minute public opinion, consensus, and trends. However, we have a limited understanding of how content and user features can influence the amount of response that posts (e.g., Twitter messages) receive, and how this can impact the growth of discussion threads. Understanding these dynamics can help users to issue better posts, and enable analysts to make timely predictions on which discussion threads will evolve into active ones and which are likely to wither too quickly. In this paper we present an approach for predicting discussions on the Social Web, by (a) identifying seed posts, then (b) making predictions on the level of discussion that such posts will generate. We explore the use of post-content and user features and their subsequent effects on predictions. Our experiments produced an optimum F_1 score of 0.848 for identifying seed posts, and an average measure of 0.673 for Normalised Discounted Cumulative Gain when predicting discussion levels.

1 Introduction

The rise of the Social Web is encouraging more and more people to use these media to share opinions and ideas, and to engage in discussions about all kinds of topics and current events. As a consequence, the rate at which such discussions are growing, and new ones are initiated, is extremely high. The last few years have witnessed a growing demand for tools and techniques for searching and processing such online conversations to, for example; get a more up-to-date analysis of public opinion in certain products or brands; identify the main topics that the public is interested in at any given time; and gauge popularity of certain governmental policies and politicians. Furthermore, governments and businesses are investing more into using social media as an effective and fast approach for reaching out to the public, to draw their attention to new policies or products, and to engage them in open consultations and customer support discussions.

In spite of the above, there is a general lack of intelligent techniques for timely identification of which of the countless discussions are likely to gain more momentum than others. Such techniques can help tools and social media analysts overcome the great challenge of scale. For example, more than 7.4 million tweets on Wikileaks[1] were posted in just a few weeks. As early as 1997, Goldhaber [6] introduced the concept of *attention economics* as a way to stress the importance of engaging user attention in the new information era. However, there is a very limited understanding of the role that certain user-characteristics and content-features play in influencing the amount of response and attention generated on these social medias. Understanding the impact of such features can support interested parties in building more effective strategies for engaging with the public on social media.

In this work we are interested in identifying the characteristics of content posted on the Social Web that generate a high volume of attention - using the microblogging platform Twitter as our data source. In particular, we explore the attributes of posts (i.e., content and the author properties) that evolved into popular discussions and therefore received a lot of attention. We then investigate the use of such attributes for making predictions on discussion activity and their contributions. We also present a behaviour ontology, designed to model statistical features that our prediction techniques use from the Social Web in a common format. More precisely, we explore the following research questions: *Is it possible to identify discussion seed posts with high-levels of accuracy? What are the key features that describe seed posts?* And: *How can the level of discussion that a post will yield be predicted?* Investigation of these questions has lead to the following contributions in this paper:

1. *Identification of seed posts*: We present a classification-based method to identify discussion seed posts. Experiments are described using two corpora of tweets with features comprised from the posts and from the post authors. Analysis of such features identified key attributes that improve the likelihood of a post eliciting a response.
2. *Prediction of discussion activity levels*: We describe a ranking-based approach to predicting discussion activity levels, enabling our analysis to place posts in order of their expected discussion volume.

This paper is structured as follows: section 2 presents related work in the area of discussion and activity prediction on social media. Section 3 describes our ontology for modelling statistical features of users and posts. Section 4 presents our method for identifying discussion seed posts, and our experiments using two datasets of tweets. Section 5 describes our prediction method, the features used and our prediction experiments. Conclusions and future work are covered in section 6.

2 Background and Related Work

We define a discussion (conversation) as a chain of two or more posts, connected together to form a directed acyclic graph. Discussions begin with the publication of a *seed post*, following which users then post a *reply* to the seed, and then a

[1] http://www.guardian.co.uk/media/wikileaks

reply is posted in response, thus forming a *discussion chain*. Our work in this paper addresses the issue of predicting discussion activity on the Social Semantic Web. This entails the identification of discussion seeds as well as the response activity and volume of the conversation they initiate. To this end, the related literature can be studied in two partially overlapping research lines.

The first line concerns the topic of identifying high quality users and content on the Social Web, understanding the factors that initiate attention towards them and contribute to their popularity. Mishne and Glance [10] juxtapose the number of comments per weblog with the number of page views and incoming links, factors which constitute the weblog's popularity - where the number of comments is strongly correlated with popularity. Hsu et.al [8] present a method to identify good quality comments on Digg stories and rank them based on user features (e.g., number of posts, age in the system, number of friendships, number of profile views and topic of activity) and content features (e.g., length, informativeness, readability and complexity). Szabo and Huberman [14] use Digg and Youtube by exploiting the number of post votes as a feature to predict views of future content. Adamic et. al [1] and Bian et.al [2] use the Yahoo! question answering service to assess the quality of questions and answers and predict the best answer chosen by the question poster, where bespoke features are used (e.g., no. of best answers per user, no. of replies per question, answer length and thread length). Ratkiewicz et. al [11], study the attention towards Wikipedia pages prior and after certain events. They quantify their models using the number of clicks on a specific page. Work by Cha et. al [5] regards attention towards a user as an indicator of influence, they study users' influence on Twitter by measuring the number of followers, number of retweets and mentions. Our work extends the current state of the art by exploring new user and content features for identifying discussion seed posts. We focus on the use of Twitter as a source for our datasets, where the role of user reputation and content quality are linked together.

The second line of work concerns the identification of conversation activity on Twitter. Although a significant amount of work exists that analyses the phenomenon of microblogging in many levels, in this work we focus on the ones that study discussion activity. Only recent work by [12] has constructed conversation models using data obtained from Twitter by employing a state transition mechanism to label dialogue acts (e.g., reference broadcast, question, reaction, comment) within a discussion chain. The most relevant work to ours, by Suh et. al [13], explores the factors which lead to a post being *retweeted*, finding that the existence of a hashtag or URL in the original post does not affect its retweeting chance, while the number of followers and followed does. Our work differs from [13] by identifying discussions conducted, as opposed to identifying information spread, and unlike existing work by [12] we extend our scope to discussions, rather than merely considering interactions of replies between 2 parties. In doing so we identify discussion seed posts and the key features that lead to starting a discussion and generating attention. To the best of our knowledge there is no existing work that can suggest and predict if and to what extent a post will be responded to.

3 Behaviour Ontology

In the context of our work - predicting discussions - we rely on statistical features of both users and posts, which we describe in greater detail in the following section. No ontologies at present allow the capturing of such information, and its description using common semantics. To fill this gap we have developed a behaviour ontology,[2] closely integrated with the Semantically Interlinked Online Communities (SIOC) ontology [3], this enables the modelling of various user activities and and related impacts on a Social Networking Site (SNS). The behaviour ontology represents posts on social networking sites, the content of those posts, the sentiment of the content and the impact which a post has had on the site. It also extends existing SIOC concepts, in particular *sioc:UserAccount*, with information about the impact the user has had on the site by capturing the number of followers and friends the user has at a given time (represented with the Data property *CollectionDate*). For the sake of brevity and space restrictions Figure 1 shows a part of the ontology that is relevant to the representation of the information used in our approach.

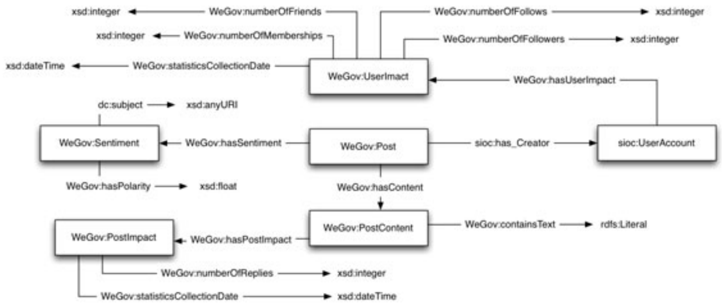


Fig. 1. The Behaviour Ontology

The two key classes presented in Figure 1 are: *PostImpact* and *UserImpact*. The former models the number of replies that a given post has generated, characterising the level of discussion that the post has yielded up until a given point in time. The latter class models the impact that the user has had within a given SNS. Capturing this information is crucial to predicting discussion activity, as according to [8,13] user reputation and standing within an online space is often a key factor in predicting whether content will generate attention or not.

4 Identifying Discussion Seeds

Identifying seed posts prior to such posts receiving a reply allows discussions to be pre-empted and tracked accordingly. We define a seed post as a post on a

[2] `http://people.kmi.open.ac.uk/rowe/ontologies/UserOnto_0.23.rdf`

given Social Web platform that will yield at least one reply - within this paper we concentrate on the use of tweets, therefore a seed post is regarded as the initial tweet that generates a reply. Features which describe seed posts can be divided into two sets: *user features* - attributes that define the user making the post; and, *content features* - attributes that are based solely on the post itself. We wish to explore the application of such features in identifying seed posts, to do this we train several machine learning classifiers and report on our findings. However we first describe the features used.

4.1 Feature Extraction

The likelihood of posts eliciting replies depends upon popularity, a highly subjective term influenced by external factors. Properties influencing popularity include user attributes - describing the reputation of the user - and attributes of a post's content - generally referred to as content features. In Table 1 we define user and content features and study their influence on the discussion "continuation".

Table 1. User and Content Features

	User Features	
In Degree:	Number of followers of U	#
Out Degree:	Number of users U follows	#
List Degree:	Number of lists U appears on. *Lists* group users by topic	#
Post Count:	Total number of posts the user has ever posted	#
User Age:	Number of minutes from user join date	#
Post Rate:	Posting frequency of the user	$\frac{PostCount}{UserAge}$
	Content Features	
Post length:	Length of the post in characters	#
Complexity:	Cumulative entropy of the unique words in post p λ of total word length n and pi the frequency of each word	$\frac{\sum_{i\in[1,n]} pi(\log\lambda - \log pi)}{\lambda}$
Uppercase count:	Number of uppercase words	#
Readability:	Gunning fog index using average sentence length (ASL) and the percentage of complex words (PCW).	[7] $0.4(ASL + PCW)$
Verb Count:	Number of verbs	#
Noun Count:	Number of nouns	#
Adjective Count:	Number of adjectives	#
Referral Count:	Number of @user	#
Time in the day:	Normalised time in the day measured in minutes	#
Informativeness:	Terminological novelty of the post wrt other posts The cumulative tfIdf value of each term t in post p	$\sum_{t\in p} tfidf(t,p)$
Polarity:	Cumulation of polar term weights in p (using Sentiwordnet[3] lexicon) normalised by polar terms count	$\frac{Po+Ne}{\lvert terms\rvert}$

4.2 Experiments

Experiments are intended to test the performance of different classification models in identifying seed posts. Therefore we used four classifiers: discriminative classifiers Perceptron and SVM, the generative classifier Naive Bayes and the decision-tree classifier J48. For each classifier we used three feature settings: user features, content features and user+content features.

[3] http://sentiwordnet.isti.cnr.it/

Datasets. For our experiments we used two datasets of tweets available on the Web: Haiti earthquake tweets[4] and the State of the Union Address tweets.[5] The former dataset contains tweets which relate to the Haiti earthquake disaster - tagged with *#haiti* - covering a varying timespan. The latter dataset contains all tweets published during the duration of president Barack Obama's State of the Union Address speech. Our goal is to predict discussion activity based on the features of a given post by first identifying seed posts, before moving on to predict the discussion level.

Within the above datasets many of the posts are not seeds, but are instead replies to previous posts, thereby featuring in the discussion chain as a node. In [13] *retweets* are considered as part of the discussion activity. In our work we identify discussions using the explicit "*in reply to*" information obtained by the Twitter API, which does not include retweets. We make this decision based on the work presented in boyd et.al [4], where an analysis of retweeting as a discussion practice is presented, arguing that message forwards adhere to different motives which do not necessarily designate a response to the initial message. Therefore, we only investigate explicit replies to messages. To gather our discussions, and our seed posts, we iteratively move up the reply chain - i.e., from reply to parent post - until we reach the seed post in the discussion. We define this process as *dataset enrichment*, and is performed by querying Twitter's REST API[6] using the *in_reply_to_id* of the parent post, and moving one-step at a time up the reply chain. This same approach has been employed successfully in work by [12] to gather a large-scale conversation dataset from Twitter.

Table 2. Statistics of the datasets used for experiments

Dataset	Users	Tweets	Seeds	Non-Seeds	Replies
Haiti	44,497	65,022	1,405	60,686	2,931
Union Address	66,300	80,272	7,228	55,169	17,875

Table 2 shows the statistics that explain our collected datasets. One can observe the difference in conversational tweets between the two corpora, where the Haiti dataset contains fewer seed posts as a percentage than the Union dataset, and therefore fewer replies. However, as we explain in a later section, this does not correlate with a higher discussion volume in the former dataset. We convert the collected datasets from their proprietary JSON formats into triples, annotated using concepts from our above behaviour ontology, this enables our features to be derived by querying our datasets using basic SPARQL queries.

Evaluation Measures. The task of identifying seed posts is a binary classification problem: *is this post a seed post or not?* We can therefore restrict our

[4] http://infochimps.com/datasets/twitter-haiti-earthquake-data
[5] http://infochimps.com/datasets/tweets-during-state-of-the-union-address
[6] http://dev.twitter.com

labels to one of two classes: *seed* and *non-seed*. To evaluate the performance our method we use four measures: precision, recall, f-measure and area under the Receiver Operator Curve. Precision measures the proportion of retrieved posts which were actually seed posts, recall measures the proportion of seed posts which were correctly identified and fallout measures the proportion of non-seed posts which were incorrectly classified as seed posts (i.e., *false positive rate*). We use f-measure, as defined in Equation 1 as the harmonic mean between precision and recall, setting $\beta = 1$ to weight precision and recall equally. We also plot the Receiver Operator Curve of our trained models to show graphical comparisons of performance.

$$F_\beta = \frac{(1+\beta^2) * P * R}{\beta^2 * P + R} \tag{1}$$

For our experiments we divided each dataset up into 3 sets: a training set, a validation set and a testing set using a 70/20/10 split. We trained our classification models using the training split and then applied them to the validation set, labelling the posts within this split. From these initial results we performed *model selection* by choosing the best performing model - based on maximising the F_1 score - and used this model together with the best performing features, using a ranking heuristic, to classify posts contained within our test split. We first report on the results obtained from our model selection phase, before moving onto our results from using the best model with the top-k features.

Table 3. Results from the classification of seed posts using varying feature sets and classification models

(a) Haiti Dataset

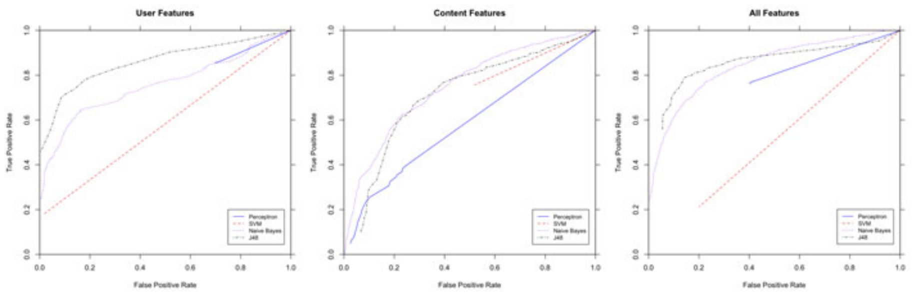
		P	R	F_1	ROC
User	Perc	0.794	0.528	0.634	0.727
	SVM	0.843	0.159	0.267	0.566
	NB	**0.948**	0.269	0.420	0.785
	J48	0.906	**0.679**	**0.776**	**0.822**
Content	Perc	**0.875**	0.077	0.142	0.606
	SVM	0.552	**0.727**	0.627	0.589
	NB	0.721	0.638	0.677	**0.769**
	J48	0.685	0.705	**0.695**	0.711
All	Perc	0.794	0.528	0.634	0.726
	SVM	0.483	**0.996**	0.651	0.502
	NB	**0.962**	0.280	0.434	**0.852**
	J48	0.824	0.775	**0.798**	0.836

(b) Union Address Dataset

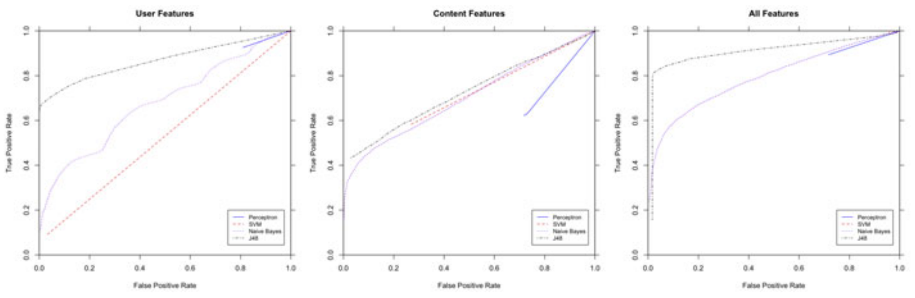
		P	R	F_1	ROC
User	Perc	0.658	0.697	0.677	0.673
	SVM	0.510	**0.946**	0.663	0.512
	NB	0.844	0.086	0.157	0.707
	J48	**0.851**	0.722	**0.782**	**0.830**
Content	Perc	0.467	**0.698**	0.560	0.457
	SVM	0.650	0.589	0.618	0.638
	NB	**0.762**	0.212	0.332	0.649
	J48	0.740	0.533	**0.619**	**0.736**
All	Perc	0.630	0.762	0.690	0.672
	SVM	0.499	**0.990**	0.664	0.506
	NB	0.874	0.212	0.341	0.737
	J48	**0.890**	0.810	**0.848**	**0.877**

4.3 Results

Our findings from Table 3 demonstrate the effectiveness of using solely user features for identifying seed posts. In both the Haiti and Union Address datasets training a classification model using user features shows improved performance over the same models trained using content features. In the case of the Union dataset we are able to achieve an F_1 score of 0.782, coupled with high precision, when using the J48 decision-tree classifier - where the latter figure (precision) indicates conservative estimates using only user features. We also achieve similar high-levels of precision when using the same classifier on the Haiti dataset. The plots of the Receiver Operator Characteristic (ROC) curves in Figure 2 show similar levels of performance for each classifier over the two corpora.When using solely user features J48 is shown to dominate the ROC space, subsuming the plots from the other models. A similar behaviour is exhibited for the Naive Bayes classifier where SVM and Perceptron are each outperformed. The plots also demonstrate the poor recall levels when using only content features, where each model fails to yield the same performance as the use of only user features. However the plots show that effectiveness of combining both user and content features.



(a) Haiti Dataset



(b) Union Address Dataset

Fig. 2. ROC Curves for Classification Models with differing Feature Sets

Experiments identify the J48 classifier as being our best performing model yielding optimum F_1 scores and by analysing the induced decision tree we observe the affects of individual features. Extremes of post polarity are found to be good indicators of seed posts, while posts which fall within this mid-polarity range are likely to be objective. One reason for this could be that the posts which elicit an emotional response are more likely to generate a reply. Analysis of the time of day identifies 4pm to midnight and 3pm to midnight as being associated with seed posts for the Haiti and Union Address datasets respectively.

Top-*k* Feature Selection. Thus far we have only analysed the use of features grouped together prompting questions: *which features are more important than others?* and *what features are good indicators of a seed post?* To gauge the importance of features in identifying seed posts we rank our features by their Information Gain Ratio (IGR) with respect to seed posts. Our rankings in Table 4 indicate that the number of lists that a user is featured in appears in the first position for both the Haiti and Union Address datasets, and the in-degree of the user also features towards the top of each ranking. Such features increase

Table 4. Features ranked by Information Gain Ratio wrt Seed Post class label. The feature name is paired within its IG in brackets.

Rank	Haiti	Union Address
1	user-list-degree (0.275)	user-list-degree (0.319)
2	user-in-degree (0.221)	content-time-in-day (0.152)
3	content-informativeness (0.154)	user-in-degree (0.133)
4	user-num-posts (0.111)	user-num-posts (0.104)
5	content-time-in-day (0.089)	user-post-rate (0.075)
6	user-post-rate (0.075)	user-out-degree (0.056)
7	content-polarity (0.064)	content-referral-count (0.030)
8	user-out-degree (0.040)	user-age (0.015)
9	content-referral-count (0.038)	content-polarity (0.015)
10	content-length (0.020)	content-length (0.010)
11	content-readability (0.018)	content-complexity (0.004)
12	user-age (0.015)	content-noun-count (0.002)
13	content-uppercase-count (0.012)	content-readability (0.001)
14	content-noun-count (0.010)	content-verb-count (0.001)
15	content-adj-count (0.005)	content-adj-count (0.0)
16	content-complexity (0.0)	content-informativeness (0.0)
17	content-verb-count (0.0)	content-uppercase-count (0.0)

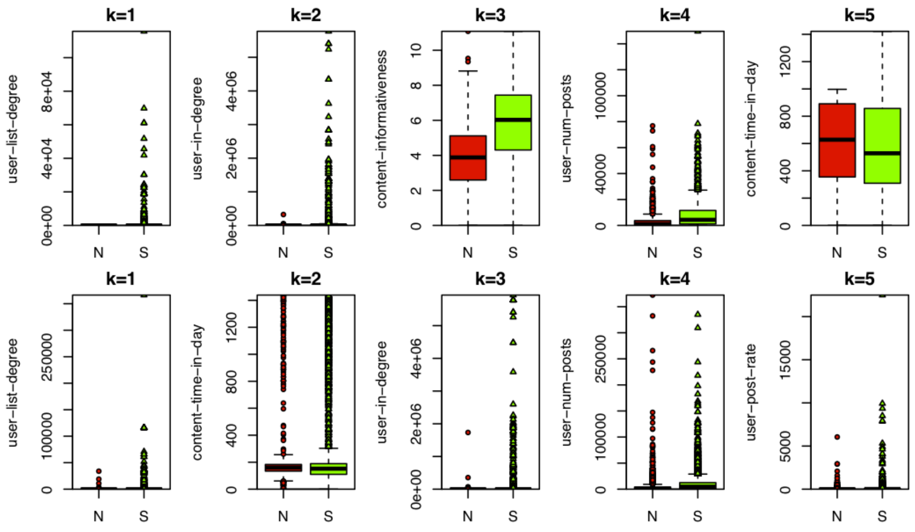
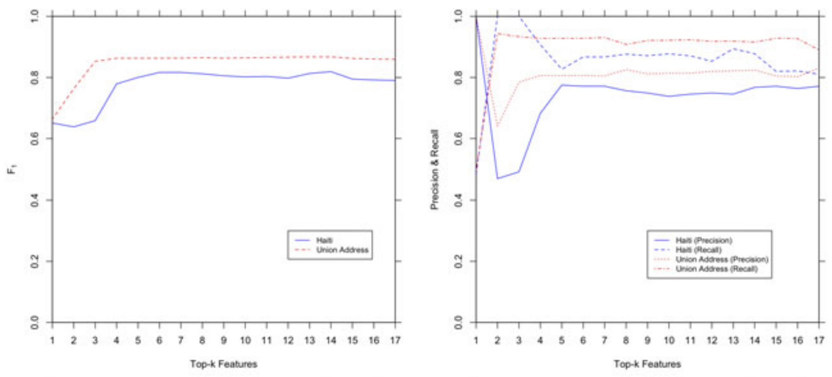


Fig. 3. Contributions of top-5 features to identifying Non-seeds (N) and Seeds (S). Upper plots are for the Haiti dataset and the lower plots are for the Union Address dataset.

the broadcast capability of the user, where any post made by the user is read by a large audience, increasing the likelihood of yielding a response. To gauge the similarity between the rankings we measured the Pearson Correlation Coefficient, which we found to be 0.674 indicating a good correlation between the two lists and their respective ranks.

The top-most ranks from each dataset are dominated by user features including the list-degree, in-degree, num-of-posts and post-rate. Such features describe a user's reputation, where higher values are associated with seed posts. Figure 3 shows the contributions of each of the top-5 features to class decisions in the training set, where the list-degree and in-degree of the user are seen to correlate heavily with seed posts. Using these rankings our next experiment explored the effects of training a classification model using *only* the top-k features, observing the affects of iteratively increasing k and the impact upon performance. We selected the J48 classifier for training - based on its optimum performance during the model selection phase - and trained the classifier using the training split from each dataset and only the top-k features based on our observed rankings. The model was then applied to the held out test split of 10%, thereby ensuring independence from our previous experiment.



(a) F_1 scores (b) Precision and Recall

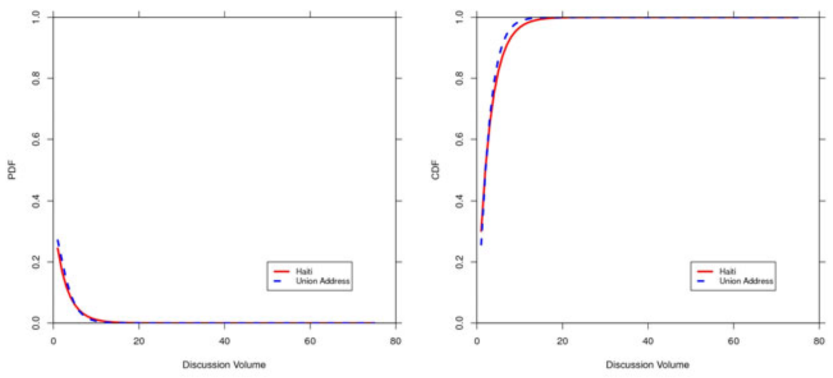
Fig. 4. Classification using J48 and top-k features ranked by Information Gain Ratio

Figure 4 shows the results from our experiments, where at lower levels of k we observe similar levels of performance - particularly when only the highest ranked feature is used (i.e., list-degree). As we increase k, including more features within our classification model, we observe improvements in F_1 scores for both datasets. The lower ranks shown in Table 4 are dominated by content features. As we include the lower-ranked features, our plots show a slight decrease in performance for the Haiti dataset due to the low IGR scores yielded for such features. For both datasets we yield 1 for precision when each model is trained using just the list-degree of the user, although J48 induces different cutoff points for the two datasets when judging the level of this nominal value. Using this feature provides a conservative estimate of seed posts, highlighted by the correlation in our training split in Figure 3. Such findings are consistent with work by [14] which found a *rich get richer* phenomena, where popular users - with a large audience of listeners/followers/observers - would yield feedback, and in doing so increase their in-degree: as the followers of their followers observed replies.

The results from our experiments have empirically demonstrated the affects of using user features in comparison with content features for identifying seed posts. Ranking features by their Information Gain Ratio also explains the importance of features, where we observe the effectiveness of including user features in identifying seed posts. However the role of content features also plays an important part, where the inclusion of the correct time of day to make a post impacts upon the likelihood of the post yielding a reply and starting a discussion.

5 Predicting Discussion Activity Levels

Identifying seed posts and the features which are commonly found within such posts, enables policy makers, and regular web users alike, to improve their posts or role in a community to ensure a reply and start a discussion. Identifying seed posts before they receive a reply also enables the tracking of key posts which are likely to yield a discussion - rather than monitoring a wide range of posts, some of which will lead to a discussion, while others may not. A natural progression of the identification of seed posts is it to predict the level of discussion that a seed post will generate. In doing so we can identify the optimum features that a given post, and its user, must have in order to maximise impact, generate a response and lead to a lengthy discussion. Predicting an exact number of replies that a given post will generate is not a straightforward task. Analysis of the training splits in our tweet datasets (Haiti and Union Address) identifies a large range in the discussion volumes and the distribution of these levels: in the Haiti dataset we found that the discussion volume ranged from 1 reply through to 74, and for the Union Address dataset we found similar levels: 1 post to 75.



(a) Probability distribution functions (b) Cumulative distribution functions

Fig. 5. Distribution functions of the Haiti and Union Address datasets

More in-depth analysis of the data is shown in Figure 5(a) and Figure 5(b), displaying the probability distributions and cumulative distributions respectively. For each dataset we used maximum likelihood estimation to optimise parameters for various distribution models, and selected the best fitting model using the Kolmogorov-Smirnov goodness-of-fit test against the training splits from our datasets. For the Haiti dataset we fitted the Gamma distribution - found to be a good fit at $\alpha = 0.1$, and for the Union dataset we fitted the Chi-squared distribution - however this was found to provide the minimum deviation from the data without satisfying any of the goodness of fit tests. The distributions convey, for both fitted datasets, that the probability mass is concentrated towards the head of the distribution where the volume of the discussion is at its lowest. The likelihood of a given post generating many replies - where many can be gauged as the mean number of replies within the discussion volume distribution - tends to 0 as the volume increases. Such density levels render the application of standard prediction error measures such as Relative Absolute Error inapplicable, given that the mean of the volumes would be used as the random estimator for accuracy measurement. A solution to this problem is instead to assess whether one post will generate a larger discussion than another, thereby producing a ranking, similar to the method used in [8].

To predict the discussion activity level we use a Support Vector Regression model trained using the three distinct feature set combinations that we introduced earlier in this paper: user features, content features and user+content features. Using the predicted values for each post we can then form a ranking, which is comparable to a ground truth ranking within our data. This provides *discussion activity levels*, where posts are ordered by their expected volume. This approach also enables contextual predictions where a post can be compared with existing posts that have produced lengthy debates.

5.1 Experiments

Datasets. For our experiments we used the same datasets as in the previous section: tweets collected from the Haiti crisis and the Union Address speech. We maintain the same splits as before - training/validation/testing with a 70/20/10 split - but without using the test split. Instead we train the regression models using the seed posts in the training split and then test the prediction accuracy using the seed posts in the validation split - seed posts in the validation set are identified using the J48 classifier trained using both user+content features. Table 5 describes our datasets for clarification.

Table 5. Discussion volumes and distributions of our datasets

Dataset	Train Size	Test Size	Test Vol Mean	Test Vol SD
Haiti	980	210	1.664	3.017
Union Address	5,067	1,161	1.761	2.342

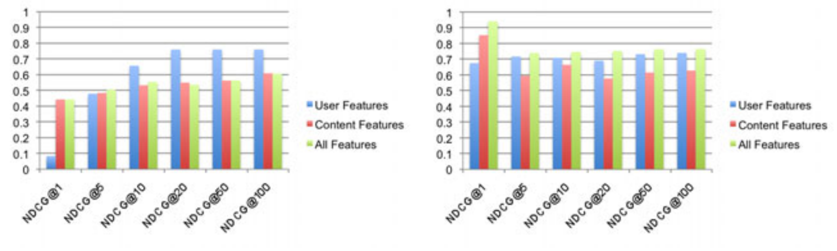
Evaluation Measures. To assess the accuracy of learnt regression models we compare the predicted rank from each model against the actual rank within our datasets - given that we have measured the discussion volume when collecting the data. Our intuition is that certain posts will attract a larger discussion volume than others, where the preference between posts based on such volume measures will enable a comparison of the ranking against a ground truth ranking based on the actual discussion volumes. To compare rankings we use the Normalised Discounted Cumulative Gain measure [9] for the top-k ranked elements, defined as $NDCG_k = DCG_k / iDCG_k$. This divides the Discounted Cumulative Gain (DCG) derived from the predicted rank against the actual rank defined as iDCG above. DCG is an empirical measure that is tailored towards rewarding rankings where the top-most elements in the ground truth are found within the same position in the predicted rank. This is motivated by web search scenarios where end users wish to find the most important documents on the first page of search results. We have a similar motivation given that we wish to identify those seed posts which yield the largest discussions and should therefore appear at the top of our ranking. We formalise DCG as:

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(1+i)} \tag{2}$$

In order to define rel_i we use the same approach as [8]: $rel_i = N - rank_i + 1$, where $rank_i$ is the ground truth rank of the element at index i from the predicted ranking. Therefore when dividing the predicted rank by the actual rank, we get a normalised value ranging between 0 and 1, where 1 defines the predicted rank as being equivalent to the actual rank. To provide a range of measures we calculated $NDCG@k$ for 6 different values where $k = \{1, 5, 10, 20, 50, 100\}$, thereby assessing the accuracy of our rankings over different portions of the top-k posts. We learnt a Support Vector Regression model for each dataset using the same feature sets as our earlier identification task: user features, content features and user+content features.

5.2 Results

Figure 6 shows the ranking accuracy that we achieve using a Support Vector Regression model for prediction over the two datasets, where we observe differing performance levels achieved using different feature set combinations. For the Haiti dataset the user features play a greater role in predicting discussion activity levels for larger values of k. For the Union Address dataset user features also outperform content features as k increases. In each case we note that content features do not provide as accurate predictions as the use of solely user features. Such findings are consistent with experiments described in [8] which found that user features yielded improved ranking of comments posted on Stories from Digg in comparison with merely content features.



(a) Haiti (b) Union Address

Fig. 6. Predicted seed post ranking using Support Vector Regression compared with ground truth ranking using *NCDG@k*

Following on from our initial rank predictions we identify the user features as being important predictors of discussion activity levels. By performing analysis of the learnt regression model over the training split we can analyse the coefficients induced by the model - Table 6 present the coefficients learnt from the user features. Although different coefficients are learnt for each dataset, for major features with greater weights the signs remain the same. In the case of the list-degree of the user, which yielded high IGR during classification, there is a similar positive association with the discussion volume - the same is also true for the in-degree and the out-degree of the user. This indicates that a constant increase in the combination of a user's in-degree, out-degree, and list-degree will lead to increased discussion volumes. Out-degree plays an important role by enabling the seed post author to see posted responses - given that the larger the out-degree the greater the reception of information from other users.

Table 6. Coefficients of user features learnt using Support Vector Regression over the two Datasets. Coefficients are rounded to 4 dp.

	user-num-posts	user-out-degree	user-in-degree	user-list-degree	user-age	user-post-rate
Haiti	-0.0019	+0.001	+0.0016	+0.0046	+0.0001	+0.0001
Union	-0.0025	+0.0114	+0.0025	+0.0154	-0.0003	-0.0002

6 Conclusions

The abundance of discussions carried out on the Social Web hinders the tracking of debates and opinion, while some discussions may form lengthy debate others may simply die out. Effective monitoring of high activity discussions can be solved by predicting which posts will start a discussion and their subsequent discussion activity levels. In this paper we have explored three research questions, the first of which asked *Is it possible to identify discussion seed posts with high-levels of accuracy?* We have presented a method to identify discussion seed posts achieving an optimum F_1 score of 0.848 for experiments over one dataset. Experiments with both content and user features demonstrated the importance of the user's reputation in eliciting a response. We sought further analyses of individual features - exploring *What are the key features that describe seed posts?* - and identified the importance of users' list-degree and in-degree: the former measuring the number of subscription channels that a given user has been added to and the latter defining the number of people subscribing to the user.

Following identification of seed posts, we then investigated: *How can the level of discussion that a post will yield be predicted?* Implementing a Support Vector Regression model produced a ranking of seed posts ordered by expected discussion activity from high to low, achieving an average measure of 0.673 for Normalised Discounted Cumulative Gain using user features. Creation of such rankings allows seed posts to be compared against other posts already generating lengthy discussions. Furthermore, through our use of regression models we are able to alter the predicted feature, enabling the time of day to publish a post to be predicted in order to maximise response.

The prediction techniques that we have outlined within this paper are designed to work over data obtained from disparate data sources. Our future work will explore the adaptation of induced models across domains and platforms and look to leverage overriding patterns for later application. Such a crossover however is not possible without a common understanding of information across such sources, therefore in this paper we have presented a behaviour ontology - built as an extension to SIOC - which allows the necessary features to be captured using common semantics from disparate domains and platforms. We will also explore the notion of *user-dependent post topic entropy*, where the specialisation of a user is captured through a topic distribution. Our intuition is that incurring followers is dependent on certain topics, where the publication of posts that cite such topics are more likely to elicit a response.

Acknowledgements

This work is funded by the EC-FP7 projects WeGov (grant number 248512) and Robust (grant number 257859).

References

1. Adamic, L.A., Zhang, J., Bakshy, E., Ackerman, M.S.: Knowledge sharing and Yahoo Answers: Everyone knows something. In: Proceedings of WWW 2008, pp. 665–674. ACM, New York (2008)
2. Bian, J., Liu, Y., Zhou, D., Agichtein, E., Zha, H.: Learning to Recognize Reliable Users and Content in Social Media with Coupled Mutual Reinforcement. In: 18th International World Wide Web Conference (WWW 2009) (April 2009)
3. Bojars, U., Breslin, J.G., Peristeras, V., Tummarello, G., Decker, S.: Interlinking the social web with semantics. IEEE Intelligent Systems 23, 29–40 (2008)
4. Boyd, D., Golder, S., Lotan, G.: Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In: Hawaii International Conference on System Sciences, pp. 1–10 (2010)

5. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.P.: Measuring User Influence in Twitter: The Million Follower Fallacy. In: Fourth International AAAI Conference on Weblogs and Social Media (May 2010)
6. Goldhaber, M.H.: The Attention Economy and the Net. First Monday 2(4) (1997)
7. Gunning, R.: The Technique of Clear Writing. McGraw-Hill, New York (1952)
8. Hsu, C.-F., Khabiri, E., Caverlee, J.: Ranking Comments on the Social Web. In: International Conference on Computational Science and Engineering, CSE 2009, vol. 4 (August 2009)
9. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. 20, 422–446 (2002)
10. Mishne, G., Glance, N.: Leave a Reply: An Analysis of Weblog Comments. In: Third annual workshop on the Weblogging ecosystem (2006)
11. Ratkiewicz, J., Menczer, F., Fortunato, S., Flammini, A., Vespignani, A.: Characterizing and modeling the dynamics of online popularity. Physical Review Letters (May 2010)
12. Ritter, A., Cherry, C., Dolan, B.: Unsupervised Modeling of Twitter Conversations. In: Proc. HLT-NAACL 2010 (2010)
13. Suh, B., Hong, L., Pirolli, P., Chi, E.H.: Want to be retweeted? Large scale analytics on factors impacting retweet in Twitter network. In: Proceedings of the IEEE Second International Conference on Social Computing (SocialCom), pp. 177–184 (August 2010)
14. Szabo, G., Huberman, B.A.: Predicting the popularity of online content. ACM Commun. 53(8), 80–88 (2010)

Mining for Reengineering: An Application to Semantic Wikis Using Formal and Relational Concept Analysis

Lian Shi[1], Yannick Toussaint[1], Amedeo Napoli[1], and Alexandre Blansché[2]

[1] LORIA CNRS – INRIA Nancy Grand-Est – Nancy Université, équipe Orpailleur, BP 70239, F-54506 Vandœuvre-lès-Nancy
`firstname.lastname@loria.fr`
[2] Laboratoire LITA, Université Paul Verlaine, Île du Saulcy F-57000 Metz
`alexandre.blansche@univ-metz.fr`

Abstract. Semantic wikis enable collaboration between human agents for creating knowledge systems. In this way, data embedded in semantic wikis can be mined and the resulting knowledge patterns can be reused to extend and improve the structure of wikis. This paper proposes a method for guiding the reengineering and improving the structure of a semantic wiki. This method suggests the creation of categories and relations between categories using Formal Concept Analysis (FCA) and Relational Concept Analysis (RCA). FCA allows the design of a concept lattice while RCA provides relational attributes completing the content of formal concepts. The originality of the approach is to consider the wiki content from FCA and RCA points of view and to extract knowledge units from this content allowing a factorization and a reengineering of the wiki structure. This method is general and does not depend on any domain and can be generalized to every kind of semantic wiki. Examples are studied throughout the paper and experiments show the substantial results.

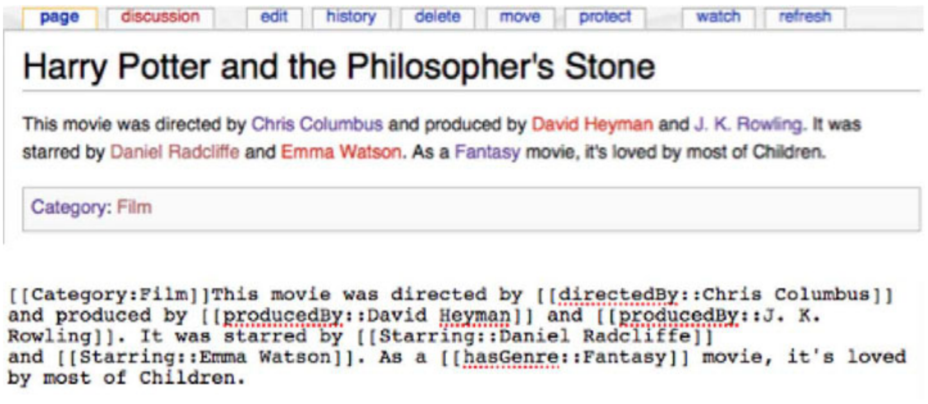
1 Introduction

Wikis provide friendly environments to create, modify and update a website, where different topics or pages are linked by hyperlinks, forming a large page network [9]. In the trend of semantic web, taking advantage of knowledge representation and ontology technologies, semantic wikis extend the capabilities of wikis by allowing annotations attached to elements in a wiki page [6]. Annotations refer to the introduction of a category for "typing" a page or a relation between an element of the page and another element in another page. Knowledge units in semantic wikis are usually represented within RDF Schema and OWL constructions, and can be queried on the fly using SPARQL for example. Therefore, semantic wikis enable communities to collaboratively produce both a large set of textual documents that human agents can easily browse thanks to semantic links and a formalized knowledge base readable by software agents. Moreover, a semantic wiki can be considered as a wide blackboard where human agents interact with software agents [3] for producing and completing knowledge. However, the collaborative and multi-user aspects introduce different perceptions of a domain and thus differences in knowledge organization. The incremental building over the time may also introduce lacks or over-definitions in the knowledge base.

Accordingly, learning techniques can be used to solve these kinds of problems by reengineering, i.e. a semantic wiki is considered as a base for discovering and organizing existing and new knowledge units. Furthermore, semantic data embedded in the semantic wikis are rarely used to enrich and improve the quality of semantic wikis themselves. There is a large body of potential knowledge units hidden in the content of a wiki, and knowledge discovery techniques are candidate for making explicit these units. The objective of the present work is to use knowledge discovery techniques –based on Formal Concept Analysis and Relational Concept Analysis– for learning new knowledge units such as categories and links between objects in pages for enriching the content and the organization of a semantic wiki. Thus, the present work aims at reengineering a semantic wiki for ensuring a well-founded description and organization of domain objects and categories, as well as setting relations between objects at the most appropriate level of description.

Reengineering improves the quality of a semantic wiki by allowing stability and optimal factorization of the category hierarchy, by identifying similar categories, by creating new categories, and by detecting inaccuracy or omissions. The knowledge discovery techniques used in this reengineering approach are based on Formal Concept Analysis (FCA) [5] and Relational Concept Analysis (RCA) [10]. FCA is a mathematical approach for designing a concept lattice from a binary context composed of a set of objects described by attributes. RCA extends FCA by taking into account relations between objects and introducing relational attributes within formal concepts, i.e. reifying relations between objects at the concept level. FCA and RCA are powerful techniques that allow a user to answer a set of questions related to the quality of organization and content of semantic wiki contents. The originality of this approach is to consider the semantic wiki content as a starting point for knowledge discovery and reengineering, applying FCA and RCA for extracting knowledge units from this content and allowing a completion and a factorization of the wiki structure (a first attempt in this direction can be found in [1]). The present approach is general and does not either depend on any domain or require customized rules or queries, thus can be generalized to every semantic wikis.

After defining some terminology about semantic wikis in Section 2, we introduce basics elements on Formal and Relational Concept Analysis and in Section 3. Section 4 gives details on the proposed approach for reengineering a semantic wiki. In Section 5, we propose an evaluation of the method based on experimental results, and we discuss the results and related issues (Section 6). After a brief review of related work, we conclude with a summary in Section 7.



Harry Potter and the Philosopher's Stone

This movie was directed by Chris Columbus and produced by David Heyman and J. K. Rowling. It was starred by Daniel Radcliffe and Emma Watson. As a Fantasy movie, it's loved by most of Children.

Category: Film

```
[[Category:Film]]This movie was directed by [[directedBy::Chris Columbus]]
and produced by [[producedBy::David Heyman]] and [[producedBy::J. K.
Rowling]]. It was starred by [[Starring::Daniel Radcliffe]]
and [[Starring::Emma Watson]]. As a [[hasGenre::Fantasy]] movie, it's loved
by most of Children.
```

Fig. 1. A wiki page titled "Harry Potter and the Philosopher's Stone". The upper half shows the content of the wiki page while the lower half is the underlying annotated form.

2 Problem Setting and Wiki Terminology

Throughout this paper, we use *wiki*(*s*) to refer to *semantic wiki*(*s*). Each wiki page is considered as a "source ontological element", including classes and properties [8]. Annotations in the page provide statements about this source element. For example, the page entitled "Harry Potter and the Philosopher's Stone" describes a movie and a set of annotations attached to this page (Figure 1).

Editors annotate an object represented by a wiki page with categories, data types, and relations, i.e. an object can be linked to other objects through relations. A category allows a user to classify pages and categories can be organized into a hierarchy. For example, the annotation `[[Category:Film]]` states that the page about "Harry Potter and the Philosopher's Stone" belongs to the category `Film`. The category `Fantasy` is a subcategory of `Film` as soon as the annotation `[[Category:Film]]` is inserted in the `Fantasy` page. Binary relationships are introduced between pages. For example, the annotation `[[Directed_by::Chris Colombus]]` is inserted in the "Harry Potter..." page for making explicit the `Directed_by` relation between "Harry Potter..." page and "Chris Colombus" page.

Some attributes are allowed to assign "values": they specify a relationship from a page to a data type such as numbers. Then `[[Duration::152min]]` give the the duration of the film "Harry Potter...".

Basically, all categories in a wiki are manually created by various editors, possibly introducing several sources of inconsistency and redundancy. The fact that the number of pages is continuously growing and that new categories are introduced is a major challenge for managing the category hierarchy construction. For keeping efficient the browsing and navigation within the wiki, the category hierarchy has to be periodically updated. Thus, a tool for automatically managing the category hierarchy in a wiki is of first importance. In the following, we show how this can be done using FCA and RCA, which are both detailed in the next section.

3 Introducing Formal Concept Analysis (FCA) and Relational Concept Analysis (RCA)

3.1 Formal Concept Analysis

The basics of FCA are introduced in [5]. Data are encoded in a formal context $\mathcal{K} = (\mathtt{G}, \mathtt{M}, \mathtt{I})$, i.e. a binary table where $\mathtt{G}$ is a set of objects, $\mathtt{M}$ a set of attributes, and $\mathtt{I} \subseteq \mathtt{G} \times \mathtt{M}$ an incidence relation. Two derivation operators, both denoted by $'$, formalize the sharing of attributes for objects, and, in a dual way, the sharing of objects for attributes:

$$' : \wp(\mathtt{G}) \longrightarrow \wp(\mathtt{M}) \text{ with } \mathtt{X}' = \{\mathtt{m} \in \mathtt{M} \mid \forall \mathtt{g} \in \mathtt{X}, \mathtt{gIm}\}$$
$$' : \wp(\mathtt{M}) \longrightarrow \wp(\mathtt{G}) \text{ with } \mathtt{Y}' = \{\mathtt{g} \in \mathtt{G} \mid \forall \mathtt{m} \in \mathtt{Y}, \mathtt{gIm}\}$$

$\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ respectively denote the powersets of $\mathtt{G}$ and $\mathtt{M}$; $\mathtt{gIm}$ states that object $\mathtt{g} \in \mathtt{G}$ is owning attribute $\mathtt{m} \in \mathtt{M}$. The two derivation operators $'$ form a *Galois connection* between $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$. Maximal sets of objects related to maximal set of attributes correspond to closed sets of the composition of both operators $'$ (denoted $''$), for $\wp(\mathtt{G})$ and $\wp(\mathtt{M})$ respectively. A pair $(\mathtt{X}, \mathtt{Y}) \in \wp(\mathtt{G}) \times \wp(\mathtt{M})$, where $\mathtt{X} = \mathtt{Y}'$ and $\mathtt{Y} = \mathtt{X}'$, is a *formal concept*, $\mathtt{X}$ being the *extent* and $\mathtt{Y}$ being the *intent* of the concept. The set $\mathcal{C}_{\mathcal{K}}$ of all concepts from $\mathcal{K}$ is ordered by extent inclusion, denoted by $\leq_{\mathcal{K}}$. Then, $\mathcal{L}_{\mathcal{K}} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ forms the *concept lattice* of $\mathcal{K}$.

For example, let us consider the two formal contexts $\mathcal{K}_{Films}$ and $\mathcal{K}_{Actors}$ given in Table 1. The context on the left provides descriptions for films while the context on the right is for actors. Each film or actor is introduced by its name and has a set of attributes. The two corresponding lattices, $\mathcal{L}_{InitFilms}$ and $\mathcal{L}_{InitActors}$ are given on Figure 2 and Figure 3.

A reduced labelling is used in the drawing of lattice: attributes are inherited from high level to low levels while objects are shared form low levels to high

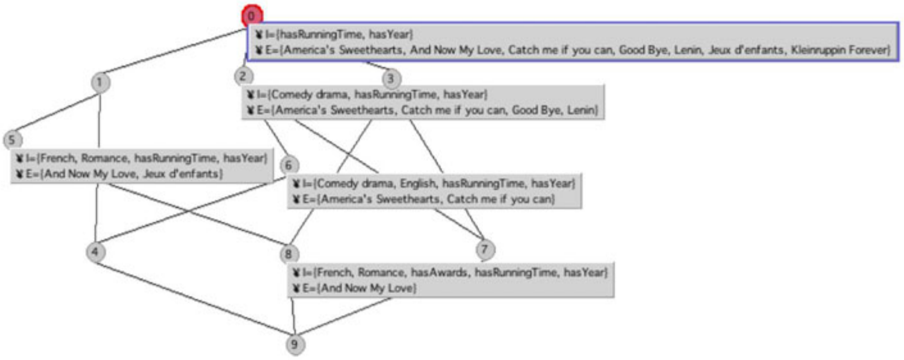

0
I={hasRunningTime, hasYear}
E={America's Sweethearts, And Now My Love, Catch me if you can, Good Bye, Lenin, Jeux d'enfants, Kleinruppin Forever}
1
2
3
I={Comedy drama, hasRunningTime, hasYear}
E={America's Sweethearts, Catch me if you can, Good Bye, Lenin}
5
I={French, Romance, hasRunningTime, hasYear}
E={And Now My Love, Jeux d'enfants}
6
I={Comedy drama, English, hasRunningTime, hasYear}
E={America's Sweethearts, Catch me if you can}
4
8
7
I={French, Romance, hasAwards, hasRunningTime, hasYear}
E={And Now My Love}
9

Fig. 2. The initial lattice of films $\mathcal{L}_{InitFilms}$

Table 1. The two binary contexts of films $\mathcal{K}_{Films}$ (left) and actors $\mathcal{K}_{Actors}$ (right)

	French	English	Germany	Romance	ComedyDrama	hasAwards	hasRunningTime	hasYear	American
Jeux d'enfants	x			x			x	x	
Good Bye, Lenin			x		x	x	x	x	
Catch me if you can		x			x		x	x	x
And now my love	x			x		x	x	x	
America's Sweethearts		x		x	x		x	x	x
Kleinruppin Forever				x			x	x	

	hasAwards	Female	Male	Age20	Age30
Guillaume Canet	x		x		x
Daniel Brühl	x		x		x
Leonardo DiCaprio	x		x	x	
Marthe Keller		x			
Tolias Schenke		x			
Julia Roberts	x	x			
Catherine Zeta Jones		x			
Anna Maria Muhe		x		x	

levels in a lattice. For example, concept `c5` in $\mathcal{L}_{InitActors}$ has for intent the set of attributes {`Female, hasAward`} (respectively from `c1` and `c3`). By contrast, concept `c3` in $\mathcal{L}_{InitActors}$ has for extent the set of individuals {`Julia Roberts, Leonardo DiCaprio, Daniel Brül, Guillaume Canet`} (respectively from `c5`, `c6`, and `c9`). When attributes are mentioned following reduced labelling, they will be said *local attributes*, otherwise *inherited attributes*. Attributes obey the following rules: when there are at least two local attributes $\mathtt{a}_1$ and $\mathtt{a}_2$ in the same intent, these attributes are equivalent, *i.e.* $\mathtt{a}_1$ appears as soon as $\mathtt{a}_2$ and reciprocally. For example, `hasRunninigTime` and `hasYear` are equivalent in $\mathcal{L}_{InitFilms}$ (see Figure 2). Moreover, local attributes imply inherited attributes. For example, `ComedyDrama` implies `hasRunninigTime` and `hasYear`.

3.2 Relational Concept Analysis

RCA was introduced and detailed in [10]. Data is described by a *relational context family* (RCF), composed of a set of contexts $\mathbf{K} = \{\mathcal{K}_\mathtt{i}\}$ and a set of binary relations $\mathbf{R} = \{\mathtt{r}_\mathtt{k}\}$. A relation $\mathtt{r}_\mathtt{k} \subseteq \mathtt{G}_\mathtt{j} \times \mathtt{G}_\ell$ connects two object sets, a *domain* $\mathtt{G}_\mathtt{j}$, i.e. $\mathtt{dom}(\mathtt{r}_\mathtt{k}) = \mathtt{G}_\mathtt{j}$, and a *range* $\mathtt{G}_\ell$, i.e. $\mathtt{ran}(\mathtt{r}_\mathtt{k}) = \mathtt{G}_\ell$. For example, the RCF corresponding to the current example is composed of the contexts $\mathcal{K}_{Films}$ and $\mathcal{K}_{Actors}$. The context $\mathcal{K}_{Starring}$ represents the relation `Starring` between films and actors (a film is starring an actor).

Hereafter, we briefly introduce the mechanism of RCA necessary for understanding the following (other details are given in [10]). RCA is based on a *relational scaling* mechanism that transforms a relation $\mathtt{r}_\mathtt{k}$ into a set

Table 2. The context $\mathcal{K}_{Starring}$ of the relation `Starring` between films and actors (films and actors are objects in the contexts $\mathcal{K}_{Films}$ and $\mathcal{K}_{Actors}$)

	Guillaume	Daniel Brühl	Leonardo DiCaprio	Marthe Keller	Tolias Schenke	Julia Roberts	Catherine Zeta Jones	Anna Maria Muhe
Jeux d'enfants	x							x
Good Bye, Lenin		x						
Catch me if you can			x					
And now my love				x		x		
America's Sweethearts						x	x	
Kleinruppin Forever				x	x			

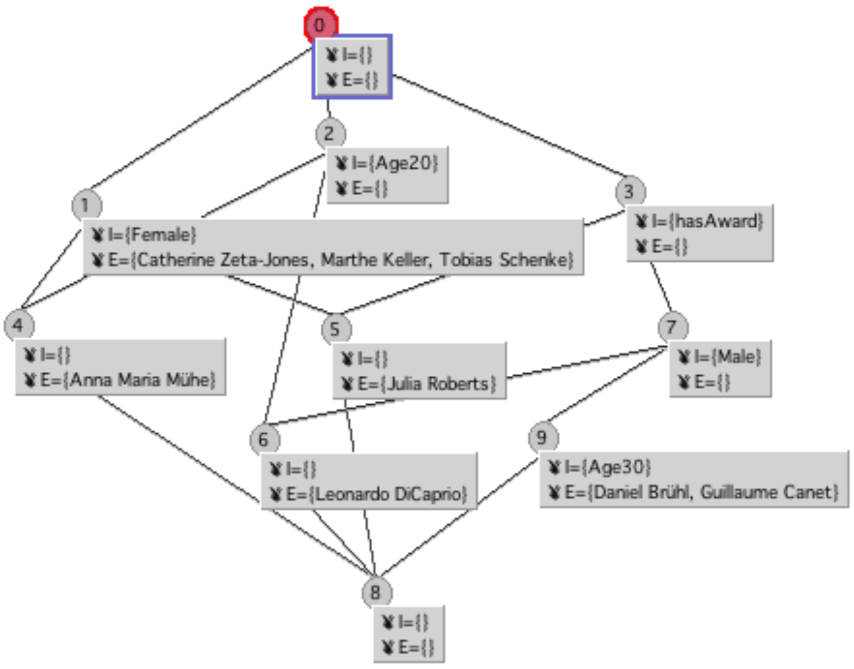


Fig. 3. The initial lattice of actors $\mathcal{L}_{InitActors}$

of *relational attributes* that are added to complete the "initial context" describing the object set $\mathtt{G_j} = \mathtt{dom}(\mathtt{r_k})$. For each relation $\mathtt{r_k} \subseteq \mathtt{G_j} \times \mathtt{G_\ell}$, there is an *initial lattice* for each object set, i.e. $\mathcal{L}_\mathtt{j}$ for $\mathtt{G_j}$ and $\mathcal{L}_\ell$ for $\mathtt{G_\ell}$. For example, the two initial lattices for the relation `Starring` are $\mathcal{L}_{InitFilms}$ (Figure 2) and $\mathcal{L}_{InitActors}$ (Figure 3).

Given the relation $\mathtt{r_k} \subseteq \mathtt{G_j} \times \mathtt{G_\ell}$, the RCA mechanism starts from two initial lattices, $\mathcal{L}_\mathtt{j}$ and $\mathcal{L}_\ell$, and builds a series of intermediate lattices by gradually completing the initial context $\mathtt{G_j} = \mathtt{dom}(\mathtt{r_k})$ with new "relational attributes". For that, relational scaling follows the DL semantics of role restrictions. Given the relation $\mathtt{r_k} \subseteq \mathtt{G_j} \times \mathtt{G_\ell}$, a relational attribute $\exists\mathtt{r_k} : \mathtt{c}$ –$\mathtt{c}$ being a concept and $\exists$ the existential quantifier– is associated to an object $\mathtt{g} \in \mathtt{G_j}$ whenever $\mathtt{r_k}(\mathtt{g}) \cap \mathtt{extent}(\mathtt{c}) \neq \emptyset$ (other quantifiers are available, see [10]). For example, let us consider the concept `c1` whose intent is `Starring` : `c3` in $\mathcal{L}_{FinalFilms}$, i.e. the final lattice of films on Figure 4. This means that all films in the extent of `c1` are related to (at least one or more) actors in the extent of concept `c3` in $\mathcal{L}_{FinalActors}$, i.e. the final lattice of actors, through the relation `Starring` (actors in the extent of `c3` are characterized by the `hasAward` attribute).

The series of intermediate lattices converges toward a "fixpoint" or "final lattice" and the RCA mechanism is terminated. This is why there is one initial and one final lattice for each context of the considered RCF. Here, $\mathcal{L}_{InitActors}$ is

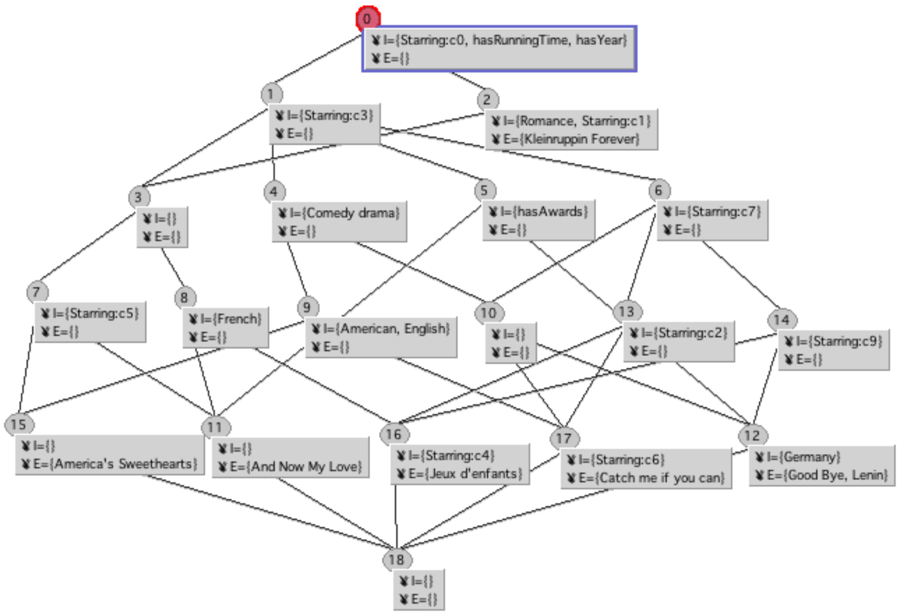


Fig. 4. The final relational lattice of films $\mathcal{L}_{FinalFilms}$

identical to $\mathcal{L}_{FinalActors}$ (Figure 3), and there are two different lattices for films, namely the initial $\mathcal{L}_{InitFilms}$ (Figure 2) and the final $\mathcal{L}_{FinalFilms}$ (Figure 4).

4 Methodology

In this section, we give details on the knowledge discovery approach used for wiki reengineering. We first explain how data are retrieved and how the different formal contexts and associated concept lattices are built. Then, we analyze the concepts and propose a new organization for the category hierarchy in the wiki.

4.1 Construction of the Relational Context Family

A complete RDF data export of a given wiki can be obtained by crawling the individual RDF export of each wiki page. According to the schema defined by Semantic MediaWiki[1] (SMW), a category is defined as a `owl:Class` and the object described in a page is exported as an instance defined by SWIVT ontology [7], which provides a basis for interpreting the semantic data exported by SMW.

The construction of a relational context family from RDF data export is based on the following correspondence. Objects in SMW are considered as objects in FCA,

[1] `http://ontoworld.org/wiki/SMW`

categories and datatype attributes in SMW are considered as attributes in FCA, and finally relations in SMW are considered as relations between objects in RCA.

We conduct SPARQL queries on the RDF dump to obtain a set of objects represented by pages ($\mathtt{O}$), a set of categories ($\mathtt{C}$), a set of datatype attributes ($\mathtt{A}$) and a set of relations ($\mathtt{R}$). Unlike the previous example in Section 3, here each RCF has only one set of objects $\mathtt{G}$ (i.e. all objects represented by wiki pages) and each relation $\mathtt{r_i} \in \mathtt{R}$ ($0 \leq \mathtt{i} \leq \mathtt{n}$) is defined on $\mathtt{G} \times \mathtt{G}$, where

- $\mathtt{G}$ consists of all objects obtained from $\mathtt{O}$.
- $\mathtt{M}$ is defined as $\mathtt{M} = \mathtt{C} \cup \mathtt{A}$.
- $\mathtt{I} \subseteq \mathtt{G} \times \mathtt{M}$. A pair $(\mathtt{g}, \mathtt{m}) \in \mathtt{I}$ iff $\mathtt{m} \in \mathtt{C}$ and object $\mathtt{g}$ belongs to this category, or $\mathtt{m} \in \mathtt{A}$ and $\mathtt{g}$ is annotated by this attribute. Additionally, the transitivity has to be explicitly stated in the context, i.e. $(\mathtt{g}, \mathtt{m}') \in \mathtt{I}$ iff $\mathtt{m}' \in \mathtt{C}$, $(\mathtt{g}, \mathtt{m}) \in \mathtt{I}$ and $\mathtt{m} \subseteq \mathtt{m}'$.
- $\mathtt{n}$ is the number of relations from $\mathtt{R}$.

By doing this, abstraction will be maximized and objects will be classified into formal concepts without any prior knowledge but RDF data,

4.2 Analyzing Formal Concepts of the Concept Lattice

Based on the two binary contexts $\mathcal{K}_{Films}$ and $\mathcal{K}_{Actors}$, and on the relational context family $\mathcal{K}_{Starring}$, we obtain a relational lattice of films shown in Figure 4. In this lattice, the intent of each concept can be divided into a set of categories, a set of attributes and a set of relational attributes. The analysis of formal concepts is driven by the following questions:

Question 1: Identifying equivalence between categories. Actually, categories appear as attributes and local attributes in a formal concept are equivalent [5]. For instance, the intent of concept `c9` in the lattice $\mathcal{L}_{FinalFilms}$ makes categories `American` and `English` equivalent, meaning that American and English movies are English speaking movies. This kind of redundancy is often due to the fact that a user may introduce a new category into the wiki without being well informed of existing ones.

Question 2: Creating new categories. A formal concept with no categories in its intent means that there is no category in the wiki for characterizing the set of objects in its extent. Therefore, a new category may be defined to capture this concept. For instance, in the lattice $\mathcal{L}_{FinalFilms}$, concept `c5` has the attribute `hasAward` that can be used for defining a new category, say "`Awarded`", which can classify objects having awards. Similarly, concept `c6` in lattice $\mathcal{L}_{FinalFilms}$ could be associated to a new category "movies starring male actors", because it has no category in its local intent but has the relational attribute `Starring:c7` and concept `c7` in the lattice $\mathcal{L}_{InitActors}$ has category `Male` as its local intent.

Question 3: Detecting category subsumption. Subsumption relations in the lattice infer subsumptions between existing categories and discovered categories. As a result, more optimized hierarchies can be defined and the organization of the wiki can be improved. In the lattice $\mathcal{L}_{FinalFilms}$, we assume that categories `English` and `American` are subcategories of `ComedyDrama` by observing concepts `c4` and `c9`. This sounds strange but is mainly due to the Closed-World Assumption of RCA, which collides with the openness of a semantic wiki and the reduced size of our running example (see Section 6).

Question 4: Defining categories. Definitions are quite rare in SMW despite they are essential. Nevertheless, definitions can help humans understanding of the purposes of categories and can be used for automatic classification by introducing necessary and sufficient conditions for an individual to belong to a category. As seen in question 1, elements in the local intent are substantially equivalent. Therefore, if a formal concept contains a category and one or more attributes in its local intent, then these attributes can be considered as a definition of that category. Moreover, any new introduced object to that category should be assigned these attributes. The case of equivalence between a category and a relational attribute is similar. For instance, concept `c2` has the category `RomanceMovie` in its intent. This category can be defined by the relational attribute `Starring:c1` where the intent of `c1` is `Female` (see lattice $\mathcal{L}_{InitActors}$). Then a romance movie would involve a female actor, and any new object in this concept should be related to at least one actress.

The result of all these is an RDF model that defines an OWL ontology containing both a TBox (new class definitions, subsumptions and equivalences) and an ABox (new instantiations). The final step is to contribute back with new knowledge to the original wiki. Our method acts as a wiki user suggesting updates. These changes, as any change from any other user, can be undone. Even if all the results are correct and consistent, some of them may be useless in practice. Therefore, in this approach it is the responsibility of a human expert or the wiki community to evaluate the reengineering proposals following the spirit of collaborative editing work for wikis. Moreover, discarded proposals should be remembered, so they are not proposed again in subsequent executions.

5 Experimental Results

We applied our method to several semantic wikis and defined criteria for evaluating the experimental results.

5.1 From Wikis to Lattices

RDF dumps from a number of semantic wikis were obtained using an *ad hoc* crawler. Seven wikis were selected for the experiments due to their representative characteristics, as summarized in Table 3. The selected wikis[2] include

[2] Retrieved by Dec 2, 2010, from `http://wiki.bioclipse.net/`, `http://hackerspaces.be/`, `http://referata.com/`, `http://geospatial.referata.com/`, `http://www.sharingbuttons.org/`, `http://tlcwiki.com/` and `http://vsk.wikia.com/`, respectively.

Table 3. Wiki characteristics in terms of the total number of pages (AP), the number of content pages (CP), the number of uncategorized content pages (UCP), the number of categories (CAT), the number of subsumptions (SUBCAT), the number of datatype attributes (DP), the number of relations (OP), the average cardinality of categories (CATSIZE), the average number of datatype attributes in content pages (DPS/CP) and the average number of relations in content pages (OPS/CF)

Semantic Wiki	AP	CP	UCP	CAT	SUBCAT	DP	OP	CATSIZE	DPS/CP	OPS/CF
Bioclipse	573	373	220	74	9	3	17	4.49	0.12	0.73
Hackerspace	821	564	312	11	0	0	14	26.00	0.00	2.56
Open Geospatial	131	120	14	4	0	0	8	26.50	0.00	4.97
Referata	316	155	121	13	0	0	48	2.85	0.00	0.88
Sharing Buttons	121	54	3	2	0	7	7	25.50	4.46	3.98
TLC	371	331	23	38	14	25	9	10.16	3.56	0.89
Virtual Skipper	820	226	43	109	106	41	7	4.17	2.42	0.28

Bioclipse, Hackerspace, Open Geospatial Encyclopedia, Referata, Sharing Buttons, Toyota Land Cruiser (TLC) and Virtual Skipper (VSK). The characteristics obtained by querying the RDF dumps using SPARQL differ from the statistics contained in the `Special:Statistics` page. These divergences were caused by the slightly different definitions for "content page" and the incompleteness of the RDF exports of some wikis.

Some of these wikis have a dense categorization (e.g., VSK has 109 categories for 226 concepts content pages, then roughly a 2:1 ratio of content pages to categories), while others are subject to inexistent or lightweight category hierarchies (e.g., Hackerspace ratio is 50:1). Ideally, we would expect to categorize some of the uncategorized pages listed under the UCP column in the table. Unfortunately, this is hampered by the fact that almost all of these unclassified pages are also lacking attributes or relations with other pages (for instance, in the Bioclipse wiki, only one out of 220 has attributes). Therefore, FCA/RCA is unable to discover categories because of inadequate information.

All objects, wiki categories, datatype attributes and relations were obtained by using Jena[3]. A custom Java script transformed each RDF model into an RCF described in XML. All lattices were produced by the Java-based Galicia[4] platform and exported to XML documents.

5.2 Results

Table 4 shows the topological characteristics of the lattices of all wikis. The number of formal concepts defines the size of the lattice. Apparently, this number is not always proportional to the size of the wiki. For instance, in spite of Bioclipse wiki being smaller than Hackerspace wiki in terms of pages, the lattice

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

[4] `http://sourceforge.net/projects/galicia/`

Table 4. Characteristics of the computed lattices

Semantic Wiki	Concepts	Edges	Depth	Connectivity	Width
Bioclipse	170	319	8	1.88	21.25
Hackerspace	21	34	5	1.62	4.20
Open Geospatial	67	120	5	1.79	13.40
Referata	24	139	4	1.63	6.00
Sharing Buttons	70	130	6	1.86	11.67
TLC	520	1059	9	2.04	57.78
Virtual Skipper	148	294	12	1.99	12.33

of Bioclipse has more concepts than Hackperspace one. In the lattice, each edge represents a subsumption relationship between concepts. Moreover, the *depth* of the lattice is defined by the longest path from the top concept down to the bottom concept, knowing that there are no cycles. The higher it is, the deeper the concept hierarchy is.

The connectivity of the lattice is defined as the average number of edges per concept. It is noteworthy that all lattices have a similar connectivity in a narrow range between 1.62 and 2.05. It seems that the characteristics of the wikis do not have a strong influence in the connectedness of the lattice. Finally, the last column gives the average number of concepts per level in the lattice. This value indicates the *width* of the lattice and it correlates to the size of the lattice. Consequently, the *shape* of a lattice is determined by both its depth and width.

Galicia produces XML files that represent the lattices as graphs, where concepts are labeled with their intent and extent. Using another custom Java application, we interpret these files and transform them into OWL/RDF graphs. Specifically, our application processes all concepts, and:

– generates `owl:equivalentClass` assertions for categories appearing as elements in the same intent of a concept,
– generates `rdf:type` assertions to express a membership of instances in the extent of a concept to the categories in the intent of the concept,
– generates `rdfs:subClassOf` assertions to subsumption relations between concepts,
– introduces new `owl:Classes` if the intent of a concept does not contain any category, and defines a new category by using the attributes in the intent.

After substracting the model given by FCA/RCA and removing trivial new categories, an RDF model is generated that contains concrete findings for reengineering the wiki. In Table 5 we report some metrics about the findings. It should be noticed that, in the case of VSK, the number of equivalences between originally existing categories (CAT-EQ) rises quickly due to the combinatorial effect and the symmetry of the equivalence (i.e. $A \equiv B$ and $B \equiv A$ count for two entries). Although some content pages are classified (CAT-CP), the lattice fails to classify originally uncategorized pages. The prime reason is that these pages often lack any attribute that can be used to derive a categorization.

Table 5. The result of analyzing the lattices in terms of the number of new proposed memberships of content pages to categories (CAT-CP), the number of proposed sub-categorizations (SUB-CAT'), the number of category equivalences between originally existing categories (CAT-EQ) and the number of proposed non-trivial categories (NEW-CAT-NT)

Semantic Wiki	CAT-CP	SUB-CAT'	CAT-EQ	NEW-CAT-NT
Bioclipse	781	194	8	73
Hackerspace	597	19	2	5
Open Geospatial	85	29	0	25
Referata	20	19	0	5
Sharing Buttons	156	26	0	18
TLC	1521	375	0	191
Virtual Skipper	181	161	58	72

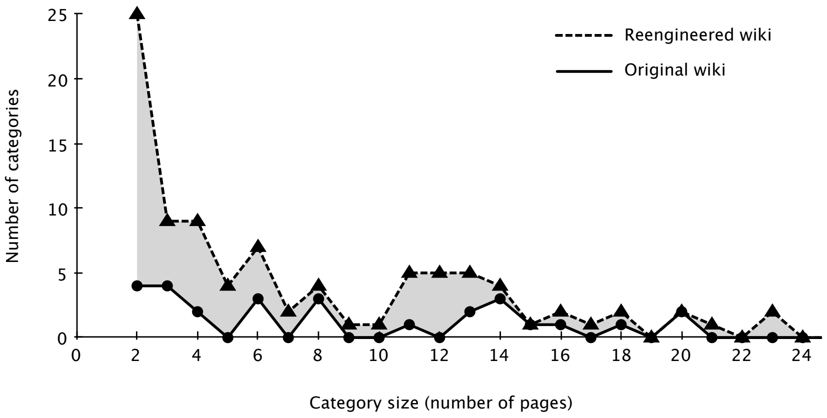


Fig. 5. Category size histogram of VSK wiki before and after FCA/RCA. Shaded area represents new proposed categories.

Figure 5 compares the category size histogram before and after the reengineering of VSK wiki. The shaded area amounts for the number of new discovered categories. The histogram clearly shows that most of the discovered categories are *small* in terms of their sizes.

The number of discovered subsumption relationships (SUB-CAT') seems to be more related to the number of discovered new categories than to the number of pre-existing ones. This indicates that in general the new categories are refinements of other ones; in other words, they have a "place" in the hierarchy.

Two of the studied wikis (Hackerspace and Referata) lead to only a few new categories. By looking into these two wikis, we found that they are already well organized and therefore provide less opportunities for reengineering, combined with the fact that these wikis do not use datatype properties.

Finally, we provide some of the generated findings to illustrate the kind of knowledge that can be obtained. For instance, in TLC wiki, the category `Has_specialty-is-Service+garage` is among the 191 proposed ones (the name is automatically generated and probably is not optimal). This new category is defined in the resulting OWL model with the Description Logic (DL) formula: $\texttt{Vendor} \sqcap \exists \texttt{has\_specialty}.\{\texttt{Service\_garage}\}$. Semantically, it can be interpreted as a subcategory of `Vendor` which aggregates those that have this precise specialty (actually, 89 out of 109 vendors).

Subsumption relations are also discovered among pre-existing categories. For instance, in the Bioclipse wiki, a rearrangement of categories into a hierarchy is proposed, with subsumptions such as $\texttt{Repositories\_maintained\_by\_Stefan\_Kuhn} \sqsubseteq \texttt{Repository}$. For some reason, this obvious subsumption link was missing in the wiki, and can be reconstructed afterwards by our method.

The discovered category equivalences and new restrictions based on attributes lead to definitions for the existing categories. Consider the discovered equivalence regarding TLC wiki: $\texttt{Diesel\_Engines} \equiv \exists \texttt{fuel}.\{\texttt{diesel}\}$. In the original wiki, only $\texttt{Diesel\_Engines} \sqsubseteq \texttt{Engine}$ is present. Therefore, the combination of both formulae provides a precise definition of the existing category, i.e. $\texttt{Diesel\_Engines} \equiv \texttt{Engine} \sqcap \exists \texttt{fuel}.\{\texttt{diesel}\}$.

6 Discussion

The experimental results show that our proposed method conduces to reengineering proposals that can go beyond what is obtained by DL-reasoning and querying on the original wiki. It is noteworthy to say that we do not compute the closure of our resulting model, which would produce an enlargement of the values in Table 5 but with little practical effect on the quality of the feedback provided to the wiki.

The method is suitable for any semantic wiki regardless of its topic or language. However, the computations are not linear with the size of the wiki (in terms of the number of pages). Precisely, the maximum size of the lattice is $2^{min(|G|,|M|)}$ with respect to FCA and $2^{min(|G|,2*|G|)}$ with respect to RCA. Therefore, processing large wikis can be a computational challenge.

FCA/RCA operates under the Closed-World Assumption (CWA), which diverges from the Open-World Assumption (OWA) of OWL reasoning. More importantly, CWA collides with the open nature of wikis. As a consequence, some of the results are counter-intuitive when they are translated back to the wiki, as it was exemplified in the previous section. However, the results are always consistent with the current data in the wiki, and the method can be repeated over time if the wiki changes. Recall that the process is “semi-automatic” and that an analysis is required.

A feedback analysis remains to be done. An approach for such an analysis is to provide results to human experts (e.g., wiki editors), who may evaluate the quality of the results based on their knowledge and experience. The quality can be measured in terms of correctness and usefulness. The latter will produce a subjective indication of the “insights” of the results, i.e., how much they go beyond the “trivial” and “irrelevant” proposals for reengineering.

7 Related Works and Conclusion

Krötzsch and co-authors [6] underlined that semantic wikis manage to disseminate semantic technologies among a broad audience and many of the emerging semantic wikis resemble small semantic webs. The knowledge model of a semantic wiki often corresponds to a small fragment of OWL-DL, but this fragment differs from one wiki to another as illustrated with SMW and IkeWiki [11]. Some extensions of SMW (like Halo Extension[5]) enable to introduce the domain and the range of a relation that are needed by FCA/RCA in order to abstract properties and relations between objects at the category level.

There exist several attempts to combine DL-based knowledge representation and FCA. One of the main works is the thesis of B. Sertkaya (see an overview in [12]). The objective is to use FCA for building a conceptualization of the world that could be encoded in an expressive DL. An “extended bottom-up approach” (computing Least Common Subsumers, Good Common Subsumers) is proposed for a bottom-up construction of a knowledge base, even including an existing knowledge base. However, SMW does not provide either disjunction or negation. Furthermore, concept lattices are mapped to the so called $\mathcal{FL}$ description logic in [10]. Then, the FCA/RCA combination provides substantial capabilities for working with wikis.

Among several domains of experiment, reengineering or refactoring UML models [4] is quite similar to the purpose of the present paper, i.e. wiki reengineering. The goal was to build abstractions in UML models.

Chernov et al. [2] defined a method for introducing semantics in “non-semantic” wikis. They attempt to define relations between categories looking at links between individuals. The method is essentially based on statistics: observing the number of links between pages in categories and computing Connectivity Ratio in order to suggest semantic connections between categories. Although there is no measure involved in the their approach, we are currently working at using numerical measures to deal with noise or omissions in the data.

Contrasting the previous work, the approach presented in [1] uses FCA on the semantic information embedded in wikis, however, authors did not distinguish attributes, relations and categories. In the present work, we go beyond by distinguishing semantical elements, in particular, datatype attributes, relations and categories, and we complete the work of FCA with the work of RCA on relations, giving better results on the reengineering of wikis.

In this paper, we proposed an approach for reengineering semantic wikis based on FCA and RCA. Our approach alleviates the human effort required to detect category redundancy, discover potential categories and identify membership between pages and category, subsumption and equivalence between categories. These objectives are achieved by analyzing formal concepts of lattices built on

[5] The Halo Project: `http://semanticweb.org/wiki/Project_Halo`

the semantic data contained in wikis. We argue that the use of FCA and RCA helps to build a well-organized category hierarchy. Our experiments show that the proposed method is adaptable and effective for reengineering semantic wikis. Moreover, our findings pose several open problems for future study.

References

1. Blansché, A., Skaf-Molli, H., Molli, P., Napoli, A.: Human-machine collaboration for enriching semantic wikis using formal concept analysis. In: Lange, C., Reutelshoefer, J., Schaffert, S., Skaf-Molli, H. (eds.) Fifth Workshop on Semantic Wikis – Linking Data and People (SemWiki-2010), CEUR Workshop Proceedings, vol. 632 (2010)
2. Chernov, S., Iofciu, T., Nejdl, W., Zhou, X.: Extracting semantic relationships between wikipedia categories. In: 1st International Workshop SemWiki2006 - From Wiki to Semantics, co-located with the ESWC 2006, Budva, (2006)
3. Cordier, A., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H., Toussaint, Y.: WikiTaaable: A semantic wiki as a blackboard for a textual case-based reasoning system. In: 4th Workshop on Semantic Wikis (SemWiki2009), held in the 6th European Semantic Web Conference (May 2009)
4. Dao, M., Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Improving generalization level in uml models iterative cross generalization in practice. In: ICCS, pp. 346–360 (2004)
5. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Berlin (1999)
6. Krötzsch, M., Schaffert, S., Vrandečić, D.: Reasoning in semantic wikis. In: Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.-L., Tolksdorf, R. (eds.) Reasoning Web. LNCS, vol. 4636, pp. 310–329. Springer, Heidelberg (2007)
7. Krözsch, M., Vrandečić, D.: Swivt ontology specification, `http://semantic-mediawiki.org/swivt/`
8. Krözsch, M., Vrandečić, D., Kolkel, M., Haller, H., Studer, R.: Semantic wikipedia. J. Web Sem., 251–261 (2007)
9. Leuf, B., Cunningham, W.: The Wiki Way: Quick Collaboration on the Web. Addison-Wesley Longman, Amsterdam (2001)
10. Rouane, M.H., Huchard, M., Napoli, A., Valtchev, P.: A proposal for combining formal concept analysis and description logics for mining relational data. In: Kuznetsov, S.O., Schmidt, S. (eds.) ICFCA 2007. LNCS (LNAI), vol. 4390, pp. 51–65. Springer, Heidelberg (2007)
11. Schaffert, S.: Ikewiki: A semantic wiki for collaborative knowledge management. In: 1st International Workshop on Semantic Technologies in Collaborative Applications (STICA 2006), Manchester, UK (2006)
12. Sertkaya, B.: Formal Concept Analysis Methods for Descriptions Logics. PhD thesis, Dresden university (2008)

SmartLink: A Web-Based Editor and Search Environment for Linked Services

Stefan Dietze, Hong Qing Yu, Carlos Pedrinaci, Dong Liu, and John Domingue

Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
{s.dietze,h.q.yu,c.pedrinaci,d.liu,j.b.domingue}@open.ac.uk

Abstract. Despite considerable research dedicated to Semantic Web Services (SWS), structured semantics are still not used significantly to annotate Web services and APIs. This is due to the complexity of comprehensive SWS models and has led to the emergence of a new approach dubbed *Linked Services*. Linked Services adopt Linked Data principles to produce simplified, RDF-based service descriptions that are easier to create and interpret. However, current Linked Services editors assume the existence of services documentation in the form of HTML or WSDL files. Therefore, we introduce SmartLink, a Web-based editor and search environment for Linked Services. Based on an easy-to-use Web form and a REST-ful API, SmartLink allows both humans as well as machines to produce light-weight service descriptions from scratch.

Keywords: Semantic Web Services, Linked Services, Linked Data, SmartLink.

1 Introduction

The past decade has seen a range of research efforts in the area of Semantic Web Services (SWS), aiming at the automation of Web service-related tasks such as discovery, orchestration or mediation. Several conceptual models, such as OWL-S [6], WSMO [3], and standards like SAWSDL [7] have been proposed, usually covering aspects such as service capabilities, interfaces and non-functional properties. However, SWS research has for the most part targeted WSDL or SOAP-based Web services, which are not prevalent on the Web. Also, due to the inherent complexity required to fully capture computational functionality, creating SWS descriptions has represented an important knowledge acquisition bottleneck and required the use of rich knowledge representation languages and complex reasoners. Hence, so far there has been little take up of SWS technology within non-academic environments.

That is particularly concerning since Web services – nowadays including a range of often more light-weight technologies beyond the WSDL/SOAP approach, such as RESTful services or XML-feeds – are in widespread use throughout the Web. That has led to the emergence of more simplified SWS approaches to which we shall refer here as “lightweight”, such as WSMO-Lite [9] SA-REST [7] and Micro-WSMO/hRESTs [4] which replace “heavyweight” SWS with simpler models expressed in RDF(S) which aligns them with current practices in the growing Semantic Web [1] and simplifies the creation of service descriptions. While the Semantic Web has successfully redefined itself as a Web of *Linked (Open) Data (LOD)* [2], the emerging *Linked Services* approach [7] exploits the established LOD principles for service description and publication, and is catering for exploiting the complementarity of the Linked Data and services to support the creation of advanced applications for the Web.

In order to support annotation of a variety of services, such as WSDL services as well as REST APIs, the EC-funded project SOA4ALL[1], has developed the Linked Services registry and discovery engine iServe[2]. iServe supports publishing service annotations as linked data expressed in terms of a simple conceptual model that is suitable for both human and machine consumption and abstracts from existing heterogeneity of service annotation formalisms: the Minimal Service Model (MSM). The MSM is a simple RDF(S) ontology able to capture (part of) the semantics of both Web services and Web APIs. While MSM [7] is extensible to benefit from the added expressivity of other formalisms, iServe allows import of service annotations following, for instance, SAWSDL, WSMO-Lite, MicroWSMO, or OWL-S. Once imported, service annotations are automatically published on the basis of the Linked Data principles. Service descriptions are thus accessible based on resolvable HTTP URIs by utilising content negotiation to return service instances in either plain HTML or RDF. In addition to a SPARQL endpoint, a REST API allows remote applications to publish annotations and to discover services through an advanced set of discovery strategies that combine semantic reasoning and information retrieval techniques. In order to support users in creating semantic annotations for services two editors have been developed: SWEET [5] (SemanticWeb sErvices Editing Tool) and SOWER (SWEET is nOt a Wsdl EditoR), which support users in annotating Web APIs and WSDL services respectively.

However, SWEET and SOWER build on the assumption that either HTML documentation of services/APIs (SWEET) or WSDL files (SOWER) are available as starting point for annotation. While that holds for a certain set of services, a number of services on the Web neither provide a WSDL nor an HTML documentation and hence, current Linked Services editors cannot be deployed in a range of cases. In addition, we would like to promote an approach were services documentation relies on structured RDF(S) and additional human-readable documentation is not provided manually but automatically generated to avoid redundancies. Therefore, we introduce and demonstrate *SmartLink*, an editing and search environment for Linked Services addressing the issues described above.

2 SmartLink: Linked Services Editor and Search Environment

In order to provide a Linked Services editor which allows the annotation of REST-ful services without any pre-existing documentation, a new services annotation and search tool was created, *SmartLink*[3] ("SeMantic Annotation enviRonmenT for Linked services"). SmartLink allows annotation of REST-ful services based on the MSM from scratch, that is, without any pre-existing services documentation such as WSDL or HTML files, as assumed by existing annotation tools (Section 1). SmartLink

[1] http://www.soa4all.eu/
[2] http://iserve.kmi.open.ac.uk
[3] http://smartlink.open.ac.uk & http://kmi.open.ac.uk/technologies/name/smartlink

operates on top of LOD stores such as iServe and is an open environment accessible to users simply via OpenID[4] authentication.

SmartLink exploits an extension of the MSM schema including a number of additional non-functional properties. These non-functional properties cover, for instance, contact person, developer name, Quality of Service (QoS), development status, service license, and WSMO goal reference. The latter property directly contributes to facilitate our approach of allowing MSM models to refer to existing WSMO goals which utilize the same service entity. MSM-schema properties are directly stored in iServe, while additional properties are captured in a complementary RDF store based on OpenRDF Sesame[5]. Due to the SmartLink-specific extensions to the MSM, we refer in the following to our Linked Services RDF store as *iServe+*. The following figure depicts the overall architecture of the SmartLink environment.

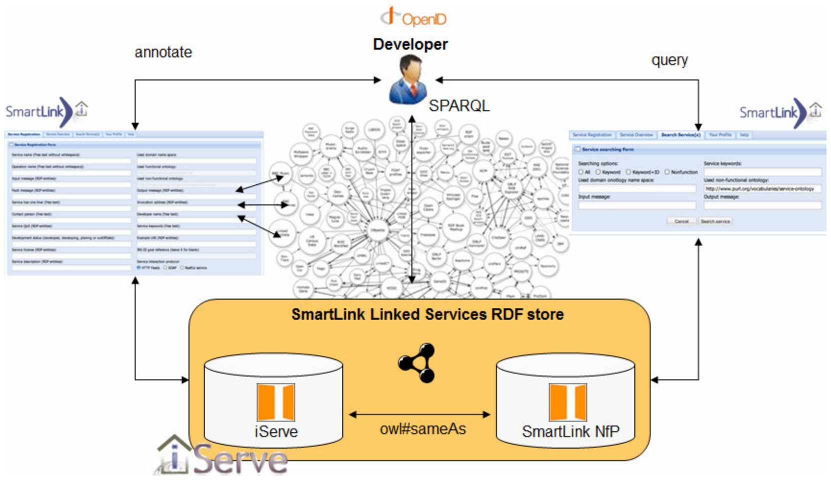


Fig. 1. SmartLink – overall architecture

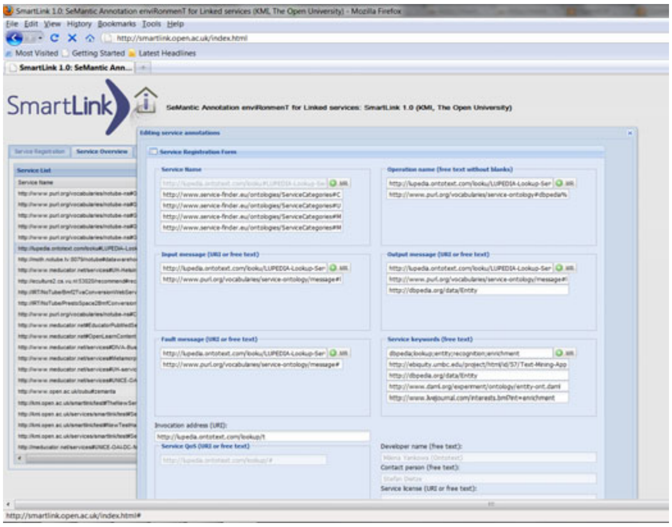
SmartLink allows developers to directly annotate existing RESTful services and APIs, which potentially also includes the annotation of WSMO goal requests – which in fact are RESTful services themselves – as MSM service instances. Figure 2 depicts the SmartLink Web interface showing the service editing form and the services library.

Being an LOD-compliant environment, one of the core features of the MSM is the capability to associate service descriptions with so-called *model references* which refer to RDF descriptions in external vocabularies defining the semantics of the service or its parts. That way, for instance, a particular *service response message* can be associated with an external RDF description which details and further describes the nature of the response. However, while this feature is useful and even necessary in order to provide meaningful service models, finding appropriate model references across the entire Web of data is a challenging task. Therefore, SmartLink uses established Linked Data

[4] http://openid.net/
[5] http://www.openrdf.org/

APIs – currently the WATSON[6] API - to identify and recommend suitable model references to the user.

Dedicated APIs allow machines and third party applications to interact with iServe+, e.g., to submit service instances or to discover and execute services. In addition, the Web application provides a search form which allows to query for particular services. Service matchmaking is being achieved by matching a set of core properties (input, output, keywords), submitting SPARQL queries, and a dedicated set of APIs.


SmartLink 1.0: SeMantic Annotation enviRonmenT for Linked services (KMI, The Open University) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://smartlink.open.ac.uk/index.html
Most Visited Getting Started Latest Headlines
SmartLink 1.0: SeMantic Ann...
SmartLink
SeMantic Annotation enviRonmenT for Linked services: SmartLink 1.0 (KMI, The Open University)
Service Registration
Service Overview
Service List
Service Name
Editing service annotations
Service Registration Form

Service Name
http://lupedia.ontotext.com/lookup#LUPEDIA-Lookup-Ser
http://www.service-finder.eu/ontologies/ServiceCategories#C
http://www.service-finder.eu/ontologies/ServiceCategories#U
http://www.service-finder.eu/ontologies/ServiceCategories#M
http://www.service-finder.eu/ontologies/ServiceCategories#M

Operation name (free text without blanks)
http://lupedia.ontotext.com/looku/LUPEDIA-Lookup-Ser
http://www.purl.org/vocabularies/service-ontology#dbpedia%

Input message (URI or free text)
http://lupedia.ontotext.com/looku/LUPEDIA-Lookup-Ser
http://www.purl.org/vocabularies/service-ontology/message#I

Output message (URI or free text)
http://lupedia.ontotext.com/looku/LUPEDIA-Lookup-Ser
http://www.purl.org/vocabularies/service-ontology/message#
http://dbpedia.org/data/Entity

Fault message (URI or free text)
http://lupedia.ontotext.com/looku/LUPEDIA-Lookup-Ser
http://www.purl.org/vocabularies/service-ontology/message#

Service keywords (free text)
dbpedia;lookup;entity;recognition;enrichment
http://ebiquity.umbc.edu/project/html/id/57/Text-Mining-App
http://dbpedia.org/data/Entity
http://www.daml.org/experiment/ontology/entity-ont.daml
http://www.livejournal.com/interests.bml?int=enrichment

Invocation address (URI):
http://lupedia.ontotext.com/lookup/t

Service QoS (URI or free text)
http://lupedia.ontotext.com/lookup/#

Developer name (free text):
Milena Yankova (Ontotext)

Contact person (free text):
Stefan Dietze

Service license (URI or free text):

http://smartlink.open.ac.uk/index.html#

Fig. 2. SmartLink – Web interface

SmartLink currently provides mechanisms which enable the export of particular (MSM) service instances as RDF or human-readable HTML. In order to facilitate service model transformation between MSM and other SWS formalisms, current research deals with the establishment of an export mechanism of MSM services.

3 Discussion and Conclusion

Current work deals with a first exploitation of SmartLink in the context of the NoTube project[7] where the ultimate goal is to develop a network of services, connected through the use of semantics, to personalize consumption of digital (IP)TV content. NoTube adopts the Linked Services approach by utilising the iServe+ and SmartLink tools. In addition, we have devised a functional classification for services specific to the NoTube domain, stored and exposed via our iServe+ environment. From our initial use case, a few observations have been made which will shape our

[6] http://watson.kmi.open.ac.uk/WatsonWUI/
[7] http://www.notube.tv

future efforts. For instance, the recommendation of LOD model references via open APIs proved very useful to aid SmartLink users when annotating services. However, due to the increasing number of LOD datasets – strongly differing in terms of quality and usefulness – it might be necessary in the future to select recommendations only based on a controlled subset of the LOD cloud in order to reduce available choices.

While SmartLink proved beneficial when creating light-weight service annotations, the lack of service automation and execution support provided by our extended MSM models, and, more importantly, the current tool support, made it necessary to transform and augment these models to into more comprehensive service models (WSMO). Due to the lack of overlap between concurrent SWS models, transformation is a manual and costly process. Hence, our current research and development deals with the extension of the MSM by taking into account execution and composition oriented aspects and the development of additional APIs, which allow the discovery, execution and semi-automated composition of Linked Services, and make the exploitation of additional SWS approaches obsolete.

Acknowledgments

This work is partly funded by the European projects NoTube and mEducator. The authors would like to thank the European Commission for their support.

References

[1] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American Magazine (2001) (retrieved March 29, 2009)

[2] Bizer, C., Heath, T., et al.: Linked data - The Story So Far. Special Issue on Linked data. International Journal on Semantic Web and Information Systems, IJSWIS (2009)

[3] Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer, Heidelberg (2007)

[4] Kopecky, J.; Vitvar, T.; and Gomadam, K. MicroWSMO. Deliverable, Conceptual Models for Services Working Group (2008), `http://cms-wg.sti2.org/TR/d12/v0.1/20090310/d12v01_20090310.pdf`

[5] Maleshkova, M., Pedrinaci, C., Domingue, J.: Supporting the creation of semantic restful service descriptions. In: 8th International Semantic Web Conference on Workshop: Service Matchmaking and Resource Retrieval in the Semantic Web, SMR2 (2009)

[6] 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. Member submission, W3C. W3C Member Submission, November 22 (2004)

[7] Pedrinaci, C., Domingue, J.: Toward the Next Wave of Services: Linked Services for the Web of Data. Journal of Universal Computer Science 16(13), 1694–1719 (2010)

[8] Sheth, A.P., Gomadam, K., Ranabahu, A.: Semantics enhanced services: Meteor-s, SAWSDL and SA-REST. IEEE Data Eng. Bull. 31(3), 8–12 (2008)

[9] Vitvar, T., Kopecky, J., Viskova, J., Fensel, D.: Wsmo-lite annotations for web services. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, Springer, Heidelberg (2008)

ViziQuer: A Tool to Explore and Query SPARQL Endpoints

Martins Zviedris and Guntis Barzdins

Institute of Matematics and Computer Science, University of Latvia, Raina bulv. 29, Riga LV-1459, Latvia
Martins.Zviedris@LUMII.lv, Guntis.Barzdins@LUMII.lv

Abstract. The presented tool uses a novel approach to explore and query a SPARQL endpoint. The tool is simple to use as a user needs only to enter an address of a SPARQL endpoint of one's interest. The tool will extract and visualize graphically the data schema of the endpoint. The user will be able to overview the data schema and use it to construct a SPARQL query according to the data schema. The tool can be downloaded from http://viziquer.lumii.lv. There is also additional information and help on how to use it in practice.

Keywords: SPARQL endpoint, Visualization, Query, Ontology.

1 Introduction

SPARQL endpoints take vital role in the Semantic Web as they provide access to actual data for end-users and software agents. Thus, it is important for developers and end-users to understand structure of the underlying data in a SPARQL endpoint to use the available data efficiently. Nowadays a problem arises as there is not enough fundamental work on the SPARQL endpoint schema definition or underlying data documentation. There is ongoing work [1] on an endpoint description, but it is in early development phase and not in actual use.

Until now SPARQL endpoints have been developed just as an access point for SPARQL queries to collect semantically structured data. There is not enough work on the SPARQL endpoint management process to document and overview the underlying data. For example, in SQL like database systems a database programmer can easily extract and view the underlying data schema, thus making much easier to develop a system that works with the database data. It is obvious that for a SPARQL endpoint user it would be also much faster and easier to work with the unknown endpoint data if the user would have more information about the actual data schema (ontology) according to which instance data in the endpoint is organized.

Use of existing query tools [2, 3, 4] is mostly like black box testing of a SPARQL endpoint, because they do not provide overview of a SPARQL endpoint data schema. Existing approaches are mostly based on faceted querying meaning that a user gets overview of only a small part of the actual ontology. The user is like a real explorer without knowledge of what will be encountered in next two steps and in which direction to browse for the needed data. Alternatively, a programmer could construct a SPARQL query by looking at some ontology and hope that a SPARQL endpoint contains the needed data and that correct URI namespaces are used. Note that most SPARQL endpoints nowadays do not provide any ontology or data schema at all as documentation, thus it is quite hard and time-consuming for a programmer to guess their data structure. There exist tools that can show all classes and all properties found in a SPARQL endpoint but not the structure of the underlying data.

Also there exists an approach [5] that can visualize ontologies as diagram graphs to aid better understanding of an ontology, but these applications are standalone and cannot connect to a SPARQL endpoint to view or edit underlying ontology. Unlike [5] we do not intend to show all possible ontology details as, but we rather concentrate on providing a simplified ontology derived from the actual data in a SPARQL endpoint.

2 Overview of the Tool

We have created a tool – ViziQuer[1] that supports a connection to a SPARQL endpoint that contains typed data. By typed data we mean that objects have the class they belong to defined by the relation rdf:type. The tool can be used to explore, understand and query SPARQL semantic database endpoint of ones interest. Thus, the ViziQuer consists of three parts. Firstly, the tool connects and extracts underlying data schema from a SPARQL endpoint. Secondly, the tool allows an end-user to graphically inspect the extracted data schema. Thirdly, an end-user can use the tool to construct a SPARQL query accordingly to the underlying data schema and execute it on the SPARQL endpoint.

The connection process begins by entering a SPARQL endpoint address that an end-user wants to connect and explore. Using predefined sequence of SPARQL queries ViziQuer extracts simple data schema from the SPARQL endpoint. Extraction process can take a while since schema retrieval depends on ontology size and speed of the SPARQL endpoint. We already mentioned that we only support typed data, as it is the basic feature required to extract at least part of the underlying data schema such as classes, their attributes and relations. SPARQL endpoints without rdf:type definitions can be explored only by RDF data browsers, for example, Tabulator [6].

There is no formal limit on data schema size; still we do not recommend trying the DBpedia[2] endpoint and similar ones. Firstly, because we make an assumption that a SPARQL endpoint does not have any limit to answer size, while DBpedia limits every answer to 1000 rows. Secondly, we do not recommend a DBpedia like endpoint as it would be very slow process to extract schema and would not give enough advantage as it would not be easy to explore the endpoint that consists of more than 1000 classes, that is a bit too much for a normal end-user to grasp without a higher ontology structuring, for example [7].

When extracting data schema from the typed underlying data, rather than from the given ontology, one is faced with the lack of definitions for subclass relations between object classes. Without subclass definitions a problem arises, as we need to

[1] http://viziquer.lumii.lv/

[2] http://DBpedia.org/sparql

depict part of relations more than once – as each logical, but not formally defined subclass might have the same outgoing relation present in data resulting in explosion of duplicate relations as one can grasp in the view of naïve schema visualization in Fig. 1 where depicted is schema visualization result of the Semantic Web Dog Food endpoint[3]. We use UML like graphical notation similar to one proposed in [5].

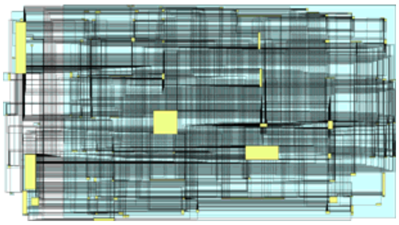


Fig. 1. The Semantic Web Dog Food endpoint full schema

As one can easily see in Fig 1, the naïve data schema visualization is opaque because subclass relations are not defined and each relation is drawn multiple times. To make it more comprehensive currently we use a hack and display each relation only once (see Fig.2) even if the relation actually occurs between more than one pair of classes. Thus, we get a comprehensive picture that is not fully semantically correct, but understandable for an end-user. In the third step when an end-user composes a SPARQL query based on the extracted schema all relation occurrences are taken into account. A semantically correct way to solve the duplicate relations problem would be to allow end-user to manually define missing subclass relations (based on meaningful names of classes) and then automatically "lift" duplicate relations to the most abstract superclass and thus making each relation to appear only once.

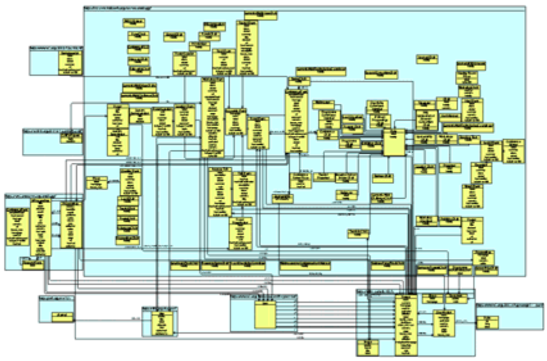


Fig. 2. The Semantic Web Dog Food with namespace and limited associations

Last important feature of the ViziQuer tool is ability to semi-automatically construct SPARQL queries according to the extracted data schema (ontology). In the

[3] http://data.semanticweb.org/sparql

ViziQuer we have implemented a subset of the GQL graphic query language [8] that provides basic features for selection of a subset of data. As queries are based on the ontology then querying is more like constructing or drawing a subset of the ontology corresponding to data that an end-user is interested in for further analysis.

A query construction is possible by two paradigms. First, a user can construct a query by selecting one class and browsing further like in faceted browsing where a user selects a possible way further from already selected classes. Second, a user can use a connection-based query construction. This means that a user selects two classes of interest by adding them to a query diagram and by drawing a line between them indicates that both classes should be interconnected. The tool uses an advanced algorithm to propose suitable paths how these two classes can be interconnected and a user just needs to select a path that best fits to desired path. Thus, when a user wants to connect some classes that are a bit further one from another, a selection of an acceptable path between classes is needed rather than a guess by manual browsing that can be quite hard in the faceted browsing paradigm if a user is not very well familiar with the ontology structure. We should also mention that both paradigms could be used in parallel when one constructs a SPARQL query.

We will briefly explain the GQL by an example query. Main idea in the GQL is selection of an interconnected subset of classes, attributes, and associations that at some point can be viewed as an ontology subset. Additionally it is possible to restrict the subset by some basic filters on class attributes. In Fig. 3 is depicted the example constructed for the Semantic Web Dog Food endpoint.

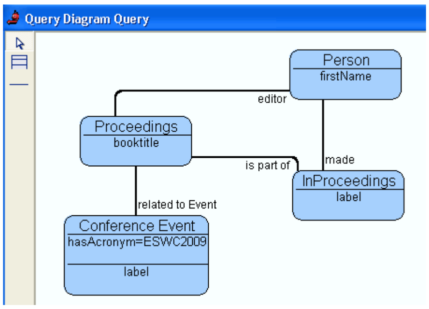


Fig. 3. Example query of the GQL based on the Semantic Web Dog Food endpoint

The query formulated in Fig. 3 could be rephrased as to “select those authors that have edited some proceedings and also have some paper in edited proceedings that are related to some conference event”. We add restriction that conference acronym is ESWC2009. We set the answer to contain a persons first name, a name of published paper, a year when proceedings where published and also a name of the conference. For limited space reasons we do not show this query translated into SPARQL.

The ViziQuer implementation is based on the Model Driven Architecture technologies that allow it to be very flexible and to connect to any SPARQL endpoint. We used the GrTP platform [9] as environment for the tool development. GrTP allows to easy manipulating graphical diagrams that is most needed to construct graphical queries and visualize a SPARQL endpoint underlying data schema. Main drawback for the platform is that it supports only the Windows operating systems, thus the ViziQuer currently works only in the Windows operating system environment.

3 Results and Conclusions

We have developed a tool that simplifies work with unknown SPARQL endpoints. The tool allows an end-user not just to build SPARQL queries, but also to get an overview of the underlying data. Still, as quality of formal ontology definitions in SPARQL endpoints is often incomplete further improvements must be made in schema management, for example, ability to manually add subclass relations between classes. Also it would be wise to consider a case when data is logically typed, but formal type definitions are missing.

Acknowledgments

This work has been partly supported by the European Social Fund within the project «Support for Doctoral Studies at University of Latvia».

References

1. SPARQL endpoint description, http://esw.w3.org/SparqlEndpointDescription
2. Heim, P., Ertl, T., Ziegler, J.: Facet Graphs: Complex Semantic Querying Made Easy. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 288–302. Springer, Heidelberg (2010)
3. Longwell RDF Browser, SIMILE (2005), http://simile.mit.edu/longwell/
4. List of SPARQL faceted browsers, http://esw.w3.org/SemanticWebTools#Special_Browsers
5. Barzdins, J., Barzdins, G., Cerans, K., Liepins, R., Sprogis, A.: OWLGrEd: a UML Style Graphical Notation and Editor for OWL. In: Clark, K., Sirin, E. (eds.) Proc. 7th International Workshop OWL: Experience and Directions, OWLED-2010 (2010)
6. Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Prud'ommeaux, E., Schraefel, M.C.: Tabulator Redux: Browsing and writing Linked Data. In: Proc. WWW 2008 Workshop: LDOW (2008)
7. Zviedris, M.: Ontology repository for User Interaction. In: d'Aquin, M., Castro, A.G., Lange, C., Viljanen, K. (eds.) ORES-2010 Workshop on Ontology Repositories and Editiors for the Semantic Web, CEUR (2010), http://CEUR-WS.org/Vol-596/
8. Barzdins, G., Rikacovs, R., Zviedris, M.: Graphical Query Language as SPARQL Frontend. In: Grundspenkis, J., Kirikova, M., Manolopoulos, Y., Morzy, T., Novickis, L., Vossen, G. (eds.) Local Proceedings of 13th East-European Conference (ADBIS 2009), pp. 93–107. Riga Technical University, Riga (2009)
9. Barzdins, J., Zarins, A., Cerans, K., Kalnins, A., Rencis, E., Lace, L., Liepins, R., Sprogis, A.: GrTP:Transformation Based Graphical Tool Building Platform. In: Proceedings of the MoDELS 2007 Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI-2007), CEUR Workshop Proceedings, vol. 297 (2007)

EasyApp: Goal-Driven Service Flow Generator with Semantic Web Service Technologies

Yoo-mi Park[1], Yuchul Jung[1], HyunKyung Yoo[1], Hyunjoo Bae[1], and Hwa-Sung Kim[2]

[1] Service Convergence Research Team, Service Platform Department, Internet Research Laboratory, ETRI, Korea
{parkym,jyc77,hkyoo,hjbae}@etri.re.kr
[2] Dept. of Electronics & Communications Eng., KwangWoon Univ., Korea
hwkim@kw.ac.kr

Abstract. EasyApp is a goal-driven service flow generator based on semantic web service annotation and discovery technologies. The purpose of EasyApp is to provide application creation environment for software programmers to make new application semi-automatically by enabling the semantic composition of web services on the Web. In this demo, we introduce key technologies of EasyApp to overcome the problems of the previous work on semantic web service technologies. Demonstration of use case 'hiring process' shows that EasyApp helps software developers make easily a service flow with key technologies: ontology-based goal analysis, semantic service annotation, semantic service discovery, and goal-driven service flow generation.

Keywords: goal-driven service flow generation, semantic web service annotation, semantic web service discovery, semantic web service composition.

1 Introduction

Semantic Web Service is a new paradigm to combine the Semantic Web and Web Services. It is expected to support dynamic computation of services as well as distributed computation. Ultimately, on Web Service, it leads to goal-based computing which is fully declarative in nature [1-3].

The previous researches on Semantic Web Services are OWL-S [4,6] and WSMO [5,6]. They suggested new service models and description languages with ontology for goal-based computing. However, these semantic web service approaches using the new models and languages require expertise on ontology and lots of manual work even for experts. In addition, they dealt with WSDL-based web services rather than RESTful web services which are commonly used in the industry recently. These limitations make it difficult to respond fast to the dynamically changing web service world that exist more than 30,000 services [8,9].

In this demo paper, we introduce a goal-driven service flow generator (i.e., EasyApp) with novel semantic web service technologies that can be applied to the currently existing web services without any changes. Towards automatic service composition, especially, we have considered semi-automatic service annotation, goal-driven semantic service discovery, and automatic service flow generation based on our goal ontology as its key enabling technologies. The key technologies can be applied for both WSDL-based services and RESTful services. In the use case 'hiring process', we show that the software developer makes service flows which satisfy his/her goals on EasyApp where our novel semantic web service technologies are embedded.

2 EasyApp

2.1 System Overview

EasyApp is a goal-driven service flow generator based on semantic web service annotation and discovery technologies. It provides application creation environment for a user to make new application by enabling semantic service composition of web services on the Web. Potential users of EasyApp are both expert software programmers and beginners.

Fig.1 shows service architecture of EasyApp and its logical functional entities that are 'goal analyzer', 'semantic service discovery', 'service flow generator', and 'source code generator'. On EasyApp, a user inputs 'user goal' and finally gets a service flow satisfied with the given goal. The user goal is parsed by referencing 'goal pattern library' and decomposed into corresponding sub-goals by the 'goal analyzer'. Each sub-goal is a criterion of the 'semantic service discovery'. The 'semantic service discovery' discovers and ranks relevant services among semantically annotated services stored in the registry. With the result of semantic service discovery, the 'service flow generator' makes up a basic service flow. After that, the user can refine the service flow manually and generate a template code in Java and XML by 'source code generator' automatically. EasyApp is strongly devoted by semantic service annotator, which is explained in the next section.

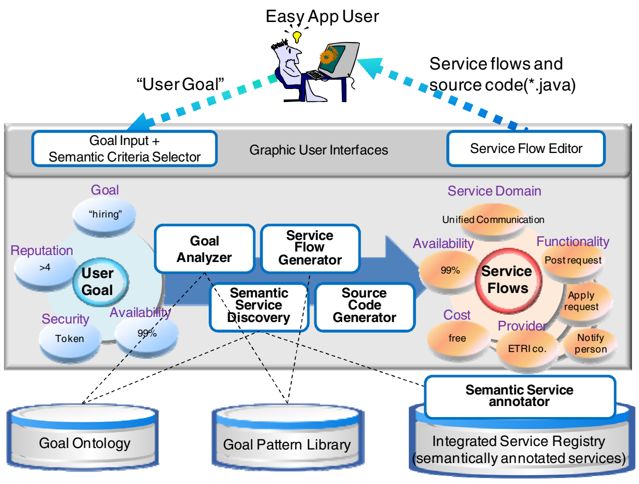


Fig. 1. Service Architecture of EasyApp

2.2 Key Technologies

We address the key technologies of this demo as follows:

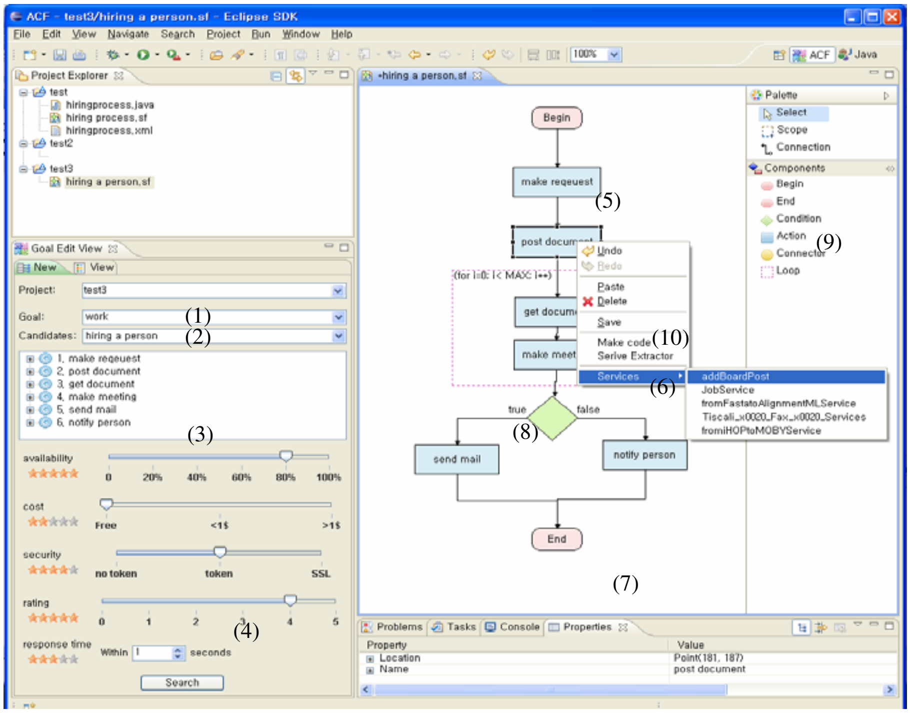
- Semantic service annotation: It involves tagging of services with additional semantic information so that service annotation becomes more meaningful and clear. We focus on the semantic annotation of web services, especially by considering their data semantics, functional semantics, and non-functional semantics about a target service as follows:
  - Data semantics : input and output parameters
  - Functional semantics: name, category, provider, location, description, and country
  - Non-functional semantics: availability, cost, security level, user rating, and response time

  We provide a semi-automatic semantic service annotation environment by supporting automatically crawled meta-information of each web service and a step-wise annotation on web user interface.
- Ontology-based goal analysis: It generates a set of sub-goals which can fulfill the user-specified goal with goal ontology and goal pattern library. In case of goal ontology, the goals widely used in the communication field are selected and constructed as OWL ontology. The goal ontology covers synonyms, hypernyms, and hyponyms. The goal pattern library stores a set of sequential sub-goals that can meet high-level goals such as "hiring a person", "making group meeting", and "planning business trip".
- Semantic service discovery: It performs a goal-driven matchmaking on semantically annotated services. In case of semantic service discovery, we employ the goal-driven service matchmaking which considers textual similarity, functional similarity [7], and constraints of non-functional properties (NFP) simultaneously. It can effectively deal with various types of service description and dynamically changing QoS of web services according to the user-specified goal.
- Goal driven service flow generator: With the result of semantic service discovery, it composes the discovered web services automatically without user's intervention. It can be achieved by referring to goal pattern library. The goal pattern library is built with well-structured goal patterns which are composed of goal and relevant sub-goals.

3 Use Case – Hiring Process

Suppose that a software programmer wants to develop a 'hiring process for a company' using some web services in the internet and intranet. The 'hiring process for a company' should handle three party communication channels among requester, applicant, and broker. Even in a small company, there is a person who requests to hire a new employee, and a person who is responsible for hiring a person, and applicants who want to apply the company. Fig. 2 shows a process of making new 'hiring process' application on EasyApp with a simple goal.

First of all, a developer catches a keyword in mind describing his/her goal, which can be ‘job’, ‘work’, or ‘hiring’. He/she inputs the keyword in the goal box (1). Based on the given keyword, EasyApp extends keywords (e.g. ‘vocation’, ‘occupation’, ‘career’, ‘recruiting’, ‘resume’, etc) as its substitutes from goal ontology and then looks up the relevant goals with the extended keywords from goal pattern library. After goal analysis, EasyApp suggests several candidate goals which concretize the user’s keyword to the developer. The suggested goals are ‘hiring a person’, ‘recruiting people’, ‘offering a job’, ‘getting a job’, and ‘making a resume’. He/she can select an appropriate goal, which is ‘hiring a person’ among them (2). Then, he/she can select additional criteria (non-functional semantics mentioned in Section 2.2) (3) to make his/her requirement clearer. When he/she clicks ‘search’ button (4), EasyApp decomposes the goal into sub-goals using goal pattern library and makes up a basic service flow that is composed of sequential sub-goals. Then, EasyApp discovers relevant services which satisfy sub-goals through semantic service discovery. During the semantic service discovery, top-k services are ranked based on a weighted sum of the textual similarity score given by keyword search and the functional similarity that represents the goal achievability of the given service. The top-k services are re-ranked by the weighted sum of each NFP’s weight and its importance. After the discovery, a service flow is displayed in the service flow editor (5).



Project: test3
Goal: work
Candidates: hiring a person

1. make reqeuest
2. post document
3. get document
4. make meeting
5. send mail
6. notify person

availability: 0, 20%, 40%, 60%, 80%, 100%
cost: Free, <1$, >1$
security: no token, token, SSL
rating: 0, 1, 2, 3, 4, 5
response time: Within 1 seconds

Property	Value
Location	Point(181, 187)
Name	post document

Fig. 2. Making service flow for ‘Hiring a person’ on EasyApp

In this use case, a service flow consists of the following sub-goals: 'make request' (for requester to make hiring request to the broker in company) → 'post document' (for requester to upload a required specification) → 'get document' (for broker to get the required specification) → 'make meeting' (for requester to meet the applicants) → 'notify person' (for requester to send result to the applicants). The sub-goals include finally ranked top-k services as a result of service discovery.

The developer can choose the most appropriate service (6) by referring to service properties represented in the 'property' view (7). He/she can modify the service flow in the editor when he/she wants by dragging & dropping activity icons (8) on palette (9). After the developer finishes the selection of services and the modification of service flows, he/she can obtain java code from the service flow (10). Finally, the developer gets a service flow for 'hiring process' on EasyApp.

4 Conclusions

In this demo, we present EasyApp, which is a novel and practical semantic web service composition environment. With EasyApp, software developer can make service flows that match the targeting goal regardless of web service programming proficiency. Further work is to employ semantic service mediation technology for on-the-fly service discovery and composition of web services.

Acknowledgments. This work was supported by the IT R&D program of KCC/KCA [KI002076, Development of Customer Oriented Convergent Service Common Platform Technology based on Network].

References

1. Fensel, D., Kerrigan, M., Zaremba, M.: Implementing Semantic Web Services. Springer, Heidelberg (2008)
2. Preist, C.: A Conceptual Architecture for Semantic Web Services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 395–409. Springer, Heidelberg (2004)
3. Cabral, L., Domingue, J., Motta, E., Payne, T.R., Hakimpour, F.: Approaches to Semantic Web Services: an Overview and Comparisons. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 225–239. Springer, Heidelberg (2004)
4. Web Service Modeling Ontology (WSMO), `http://www.w3.org/Submission/WSMO/`
5. Semantic Markup for Web Services (OWL-S), `http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/`
6. Lara, R., Roman, D., Polleres, A., Fensel, D.: A Conceptual Comparison of WSMO and OWL-S. In: European Conference on Web Services, pp. 254–269 (2004)
7. Shin, D.-H., Lee, K.-H., Suda, T.: Automated Generation of Composite Web Services based on Functional Semantics. Journal of Web Semantics 7(4), 332–343 (2009)
8. Seekda, `http://webservices.seekda.com/`
9. ProgrammableWeb, `http://www.programmableweb.com/`

Who's Who – A Linked Data Visualisation Tool for Mobile Environments

A. Elizabeth Cano[1,⋆], Aba-Sah Dadzie[1], and Melanie Hartmann[2]

[1] OAK Group, Dept. of Computer Science, The University of Sheffield, UK
[2] Telecooperation Group, Universität Darmstadt, Germany
{a.cano,a.dadzie}@dcs.shef.ac.uk, melanie@tk.de

Abstract. Reduced size in hand-held devices imposes significant usability and visualisation challenges. Semantic adaptation to specific usage contexts is a key feature for overcoming usability and display limitations on mobile devices. We demonstrate a novel application which: (i) links the physical world with the semantic web, facilitating context-based information access, (ii) enhances the processing of semantically enriched, linked data on mobile devices, (iii) provides an intuitive interface for mobile devices, reducing information overload.

Keywords: linked data, semantic web, visualisation, mobile devices.

1 Introduction

Mobile devices are increasingly becoming an extension of the lives of humans in the physical world. The popularity of these devices simplifies *in-situ* management of the ordinary end user's information needs. Specifically, smart phones' embedded devices (e.g., built-in cameras) allow to build an abstraction of the user's environment. Such abstraction provides contextual information that designers can leverage in adapting a mobile interface to the user's information needs. Further, context can act as a set of parameters to query the Linked Data (LD) cloud. The cloud connects distributed data across the Semantic Web; it exposes a wide range of heterogeneous data, information and knowledge using URIs (Uniform Resource Identifiers) and RDF (Resource Description Framework) [2,6]. This large amount of structured data supports SPARQL querying and the *follow your nose* principle in order to obtain facts. We present *Who's Who*, a tool that leverages structured data extracted from the LD cloud to satisfy users' information needs ubiquitously. The application provides the following contributions:

1. **Exploiting contextual information:** *Who's Who* facilitates access to the LD cloud by exploiting contextual information, linking the physical world with the virtual.
2. **Enhanced processing of Linked Data on mobile devices:** *Who's Who* enables processing of semantic, linked data, tailoring its presentation to the limited resources of mobile devices, e.g., reducing latency when querying semantic data by processing triples within a mobile browser's light-weight triple store.
3. **Mobile access to Linked Data:** *Who's Who* uses novel visualisation strategies to access LD on mobile devices, in order to overcome the usability challenges arising from the huge amount of information in the LD cloud and limited mobile device display size. This visualisation also enables intuitive, non-expert access to LD.

⋆ To whom correspondence should be addressed.

2 Application Scenario

To illustrate our work consider the following scenario:

Bob is a potential masters student invited to an open day at a university. He will tour different departments to get to know their facilities and research. To make the best of the open day, Bob will use his mobile phone as a location guide, allowing him to retrieve information (encoded as LD) about each department he visits, with the aid of visual markers distributed around the university. This will allow him to identify researchers he would like to meet and potential projects to work on.

We demonstrate the approach taken in *Who's Who* to realise this scenario, i.e., to support user- and context-sensitive information retrieval from the LD cloud using a mobile device. We exemplify this using the *Data.dcs* [6] linked dataset, which describes the research groups in the Department of Computer Science at the University of Sheffield.

3 The *Who's Who* Application

Who's Who facilitates entry into the LD cloud by exploiting context (section 3.1), reduces potential latency due to resource restrictions on even advanced mobile devices (section 3.2) and enables seamless management of information load (section 3.3).

3.1 Exploiting Context to Augment Physical Entities

The digital augmentation of physical entities has become a popular feature of many outdoor and indoor mobile applications [4]. For example, outdoors, a user's GPS (Global Positioning System) provides a contextual parameter for retrieving nearby points of interest. However, GPS is not suited for indoor use, due to among others, poor signal strength and range accuracy. Alternatives to location-based augmentation for indoor environments are visual detection and RFID (Radio Frequency IDentification) tags, which can be used to attach hyperlinks to real-world objects. In contrast to RFID readers, visual markers enable widespread use since they can be easily produced by regular printers, and read by standard cameras, which are now integrated into most mobile phones.

To bridge the gap between a physical entity's representation and existing information regarding it in the LD cloud, we follow the approach illustrated in Fig. 1, which consists of the following process: (1) a URI encoded in a visual marker represents a physical entity – in our scenario (section 2), a research group housed in the building at the location in question; (2) this URI translates to a server-side request which queries the LD cloud to enrich the information related to this entity – in our scenario, with research group members, their collaborations and publications; (3) this information is processed on the mobile device and presented to the end user.

3.2 Processing of Linked Data on Mobile Devices

Increments in memory capacity and processing power in mobile devices, particularly in smart phones, allow semantic processing of large numbers of triples (e.g., the Apple

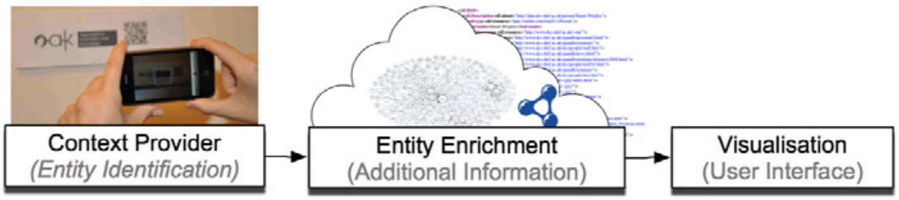


Fig. 1. The *Who's Who* application in a nutshell

iPhone 3GS has a 600Mhz CPU and 256MB RAM, and the HTC Hero has a 528MHz CPU and 288MB RAM; see also benchmarks for semantic data processing in small devices in [3]). Although it is possible to handle triples in local RDF stores in Android-based mobiles, this is not possible in other platforms such as the iPhone. An alternative is to use existing lightweight developments such as rdfQuery[1], which runs on web browsers, and HTML 5 features for persisting storage.

However, processing and rendering of semantic resources on mobile web browsers is still limited by low memory allocation (e.g., 10-64MB in Webkit and Firefox mobile on iPhone and Android phones). Leaving entirely the processing and rendering of semantic resources to the mobile client improves the user experience by reducing latency due to multiple requests. However, memory allocation restrictions make this a sub-optimal option. On the other hand, executing the semantic analysis and data processing entirely on the server-side results in the execution of continuous calls to the server, which translates to high data latency and a degradation of the responsiveness of the user interface and interactivity. There must be a compromise between the number of triples handled by a (mobile device) web browser and visualisation flow performance.

Who's Who follows the mobile and server-side based architecture in Fig. 2. Based on the parameters encoded in a visual marker, *Who's Who* queries *Data.dcs*. The *Data.dcs* triples are loaded in-memory via Jena on the server-side, following which SPARQL queries are executed. The triples retrieved are encoded with JSON – a lightweight data-interchange format – using JSONLib[2], and returned with a JavaScript callback to the mobile device. On the *Who's Who* mobile-side, the triples are loaded into an rdfQuery lightweight triple store. Interaction with the visualisation triggers local SPARQL queries that further filter the information.

The advantages of adopting this approach are that: 1) users need not download the application in advance (as is the case with applications relying on local RDF storage); 2) users need not know the URI corresponding to the physical entity they want to enrich, as contextual information is pre-encoded in the visual markers; 3) there is a balance between the triple load handled by the server- and mobile-sides, which translates to more responsive user interfaces; 4) the mobile-side triple store allows semantic filtering on the views exposed to the user, reducing latency and improving the interface's usability.

[1] rdfQuery: `http://code.google.com/p/rdfquery`
[2] JSONLib: `http://json-lib.sourceforge.net`

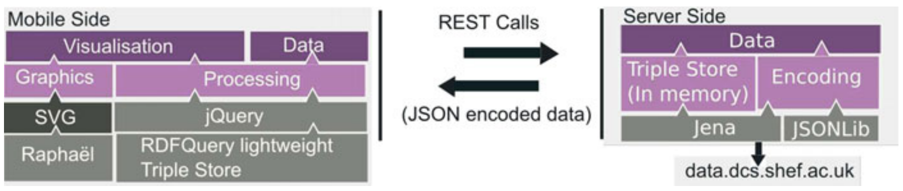
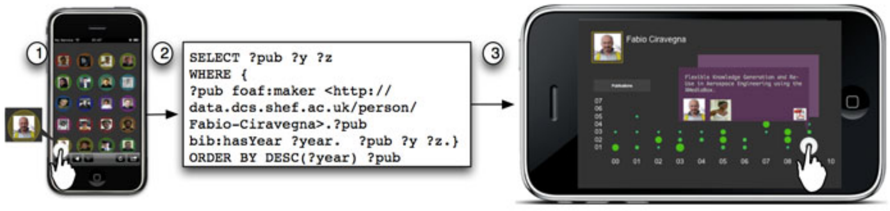


Fig. 2. Mobile- and server-side interaction in Who's Who architectural design

3.3 Visualisation

Who's Who supports the user in retrieving information stored in the LD cloud with visualisations tailored to the application domain. User requests are automatically translated to SPARQL queries executed on the lightweight triple store on the mobile device itself. If required, additional triples are retrieved from the *Who's Who* server. Fig. 3 describes the interaction flow for retrieving publications: 1) the user is presented a list of researchers corresponding to the physical entity encoded in the scanned visual marker; 2) when the user taps on a researcher – in this case *Fabio Ciravegna* – a SPARQL query is executed; 3) the publication view is presented, providing an additional filtering layer.

The publication view shows a graph containing the triples resulting from the SPARQL query – the number of publications per year and the number of collaborators involved in each publication. In Fig. 3 (3), the user has tapped on the graph bubble corresponding to the year *2009*, which links to two collaborators. The publications are arranged in a "card deck", where the first publication appears in the foreground. The user can traverse through the publications – where there are multiple – by selecting the next in the deck.



```
SELECT ?pub ?y ?z
WHERE {
?pub foaf:maker <http://
data.dcs.shef.ac.uk/person/
Fabio-Ciravegna>.?pub
bib:hasYear ?year.  ?pub ?y ?z.}
ORDER BY DESC(?year) ?pub
```

Fig. 3. (1) After selecting a researcher; (2) a SPARQL query is executed; (3) the resulting triples are presented in the graph in the publication view

4 Related Work

Searching for information about entities and events in a user's environment is an oft-performed activity. The state of the art focuses on text-based browsing and querying of LD on desktop browsers, e.g., *Sig.ma* [8] and *Marbles* [1], targeted predominantly at technical experts (see also [2]). This excludes a significant part of the user population – non-technical end users – who make use of advanced technology embedded in everyday devices such as mobile phones. One of the best examples of a visual browser targeted at mainstream use is *DBPedia Mobile* [1]; which is a location-aware Semantic Web client that identifies and enriches information about nearby objects. However it relies on GPS sensors for retrieving context, which makes it unsuitable for our indoor scenario. Our approach improves on existing LD browsers for mobile devices in that *Who's Who*: 1) extracts contextual information encoded in visual markers; 2) hides explicit SPARQL filters from the user, increasing usability for especially non-technical users.

5 Summary

Who's Who was developed to support especially those end users who may have little to no knowledge about where to find information on nearby physical entities. It provides exploratory navigation through new environments, guided by the user's context. Studies (see, e.g., [5,7]) evaluating the utility and usability of tag-based interaction with mobile device applications illustrate the potential of lowering barriers to LD use.

We have demonstrated the use of a set of visual markers, corresponding to research groups in a university department, to explore the linked data exposed in *Data.dcs*, using a smart phone equipped with a camera and a QRcode scanner. We have also illustrated how the approach taken in *Who's Who* simplifies such tasks, by using visualisation of structured data to extract relevant context and manage information load, to reveal interesting facts (otherwise difficult to identify), and to facilitate knowledge extraction.

Acknowledgements. A.E. Cano is funded by CONACyT, grant 175203. A.-S. Dadzie and M. Hartmann are funded by the European Commission (EC) project SmartProducts (231204). This work was also supported by the EC project WeKnowIt (215453).

References

1. Becker, C., Bizer, C.: Exploring the geospatial semantic web with DBpedia Mobile. Journal of Web Semantics 7(4), 278–286 (2009)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems (2009)
3. d'Aquin, M., Nikolov, A., Motta, E.: How much semantic data on small devices? In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 565–575. Springer, Heidelberg (2010)
4. Fröhlich, P., Oulasvirta, A., Baldauf, M., Nurminen, A.: On the move, wirelessly connected to the world. ACM Commun. 54, 132–138 (2011)
5. Mäkelä, K., Belt, S., Greenblatt, D., Häkkilä, J.: Mobile interaction with visual and RFID tags: a field study on user perceptions. In: Proc. CHI 2007, pp. 991–994 (2007)
6. Rowe, M.: Data.dcs: Converting legacy data into linked data. In: Proc., Linked Data on the Web Workshop at WWW'10 (2010)
7. Toye, E., Sharp, R., Madhavapeddy, A., Scott, D., Upton, E., Blackwell, A.: Interacting with mobile services: an evaluation of camera-phones and visual tags. Personal and Ubiquitous Computing 11, 97–106 (2007)
8. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: live views on the web of data. In: WWW'10, pp. 1301–1304 (2010)

OntosFeeder – A Versatile Semantic Context Provider for Web Content Authoring

Alex Klebeck[1], Sebastian Hellmann[2], Christian Ehrlich[1], and Sören Auer[2]

[1] Ontos GmbH, Poetenweg 49, Leipzig, Germany
{alex.klebeck,christian.ehrlich}@ontos.com
http://ontos.com

[2] Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany
{hellmann,auer}@informatik.uni-leipzig.de
http://aksw.org

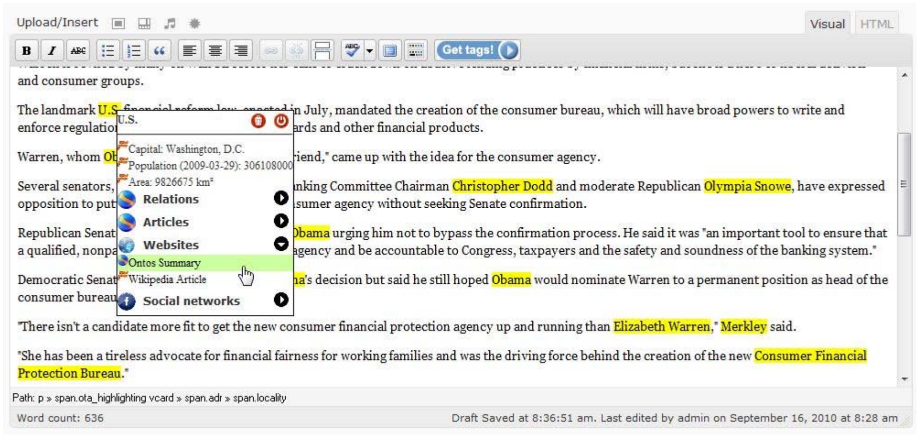
Abstract. As the amount of structured information available on the Web as Linked Data has reached a respectable size. However, the question arises, how this information can be operationalised in order to boost productivity. A clear improvement over the keyword-based document retrieval as well as the manual aggregation and compilation of facts is the provision of contextual information in an integrated fashion. In this demo, we present the *Ontos Feeder* – a system serving as context information provider, that can be integrated into Content Management Systems in order to support authors by supplying additional information on the fly. During the creation of text, relevant entities are highlighted and contextually disambiguated; facts from trusted sources such as *DBpedia* or *Freebase* are shown to the author. Productivity is increased, because the author does not have to leave her working environment to research facts, thus media breaks are avoided. Additionally, the author can choose to annotate the created content with *RDFa* or *Microformats*, thus making it "semantic-ready" for indexing by the new generation of search engines. The presented system is available as Open Source and was adapted for *WordPress* and *Drupal*.

1 Introduction

One of the routine tasks of a content author (e.g. a journalist) during the time of writing is researching for context information required for the intended article. Without proper tool support, the author has to resort to manual searching (e.g. Google) and skimming through available information sources. The availability of structured data on the Semantic Data Web allows to automate these routine activities by identifying topics within the article with the aid of Natural Language Processing (NLP) and subsequently presenting relevant context information by retrieving descriptions from the Linked Open Data Web (LOD).

We present the *Ontos Feeder*[1] – a system serving as context information provider, that can be integrated into Content Management Systems in order to

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



and consumer groups.

The landmark U.S. ... in July, mandated the creation of the consumer bureau, which will have broad powers to write and enforce regulatio... ...ards and other financial products.

Warren, whom Ob... ...iend," came up with the idea for the consumer agency.

Several senators, ...nking Committee Chairman Christopher Dodd and moderate Republican Olympia Snowe, have expressed opposition to put... ...sumer agency without seeking Senate confirmation.

Republican Senat... ...Obama urging him not to bypass the confirmation process. He said it was "an important tool to ensure that a qualified, nonpa... ...agency and be accountable to Congress, taxpayers and the safety and soundness of the banking system."

Democratic Senat... ...na's decision but said he still hoped Obama would nominate Warren to a permanent position as head of the consumer bureau...

"There isn't a candidate more fit to get the new consumer financial protection agency up and running than Elizabeth Warren," Merkley said.

"She has been a tireless advocate for financial fairness for working families and was the driving force behind the creation of the new Consumer Financial Protection Bureau."

Capital: Washington, D.C.
Population (2009-03-29): 306108000
Area: 9826675 km²

Path: p » span.ota_highlighting vcard » span.adr » span.locality

Word count: 636

Draft Saved at 8:36:51 am. Last edited by admin on September 16, 2010 at 8:28 am

Fig. 1. Entities are highlighted in the WYSIWYG editor of the CMS, Pop-ups allow to select further information

support authors by supplying additional information on the fly. Ontos Feeder uses the Ontos Web Service (OWS, see Section 3) to analyse the article text and retrieve *Ontos Entity Identifier* (OEI) URIs for relevant topics. These OEIs are interlinked with several data sources on the web and enriched with internal facts from the *Ontos Knowledge Base*. The Feeder is open-source and currently available for the CMS (Drupal[2] and Wordpress[3]. Additionally, the Feeder can automatically annotate the article with Microformats and RDFa annotations. These are increasingly utilized by search engines such as Google or Bing [4].

2 Feature Description and User Interface Walkthrough

The content creation process begins with the writing of an article in a supported CMS system. Having written the content, the author clicks on the `get tags` button to send the text to the OWS. The OWS analyses the text and returns disambiguated URIs for the found entities. Then the Ontos Feeder annotates the returned entities in the original text within the CMS and highlights them in the WYSIWYG editor (see Figure 1). In a **context information area** of the CMS an overview of the found entities is given in the form of thumbnails (see Figure 2). Now the author has several choices:

– View additional information about the entities and navigate recursively.
– Adapt the filter (e.g. by type) in the config options and remove some of the entities.
– Revise the text and resend the text to the OWS.
– Accept all annotated entities and publish them along with the text.

---

[2] http://sourceforge.net/projects/ontosfeeder

[3] http://wordpress.org/extend/plugins/ontos-feeder/

[4] http://ivan-herman.name/2009/12/12/rdfa-usage-spreading.../ and http://www.mahalo.com/rdfa

If an author requires additional information about a particular entity, pointing at each annotation or thumbnail results in showing an appropriate pop-up menu with further contextual information. Each entity type provides different context information depending on their source; some of them are gathered from LOD providers such as DBpedia or Freebase, and some are coming directly from the OWS itself. While the LOD providers are used to retrieve entity attributes like age, nationality or area, the OWS provides information from the Ontos Knowledge Base comprised of information about the relationships to other entities (persons or organisations), related news articles as well as a summarizing entity report. Clicking on the related persons or organisations link in the pop-up menu refreshes the context information area with the thumbnails of that entities, so that the author can navigate recursively through all the relationships.

The *Ontos Knowledge Base* contains aggregated information from various sources. The URIs assigned to the extracted entities by the Web Service are *Ontos Entity Identifiers* (OEI). OEIs are de-referencable identifiers and are connected via `owl:sameAs` links to other Linked Data entities. Therefore, additional information from other Linked Data providers such as *DBpedia* and *Freebase* is presented in the entity context as well.

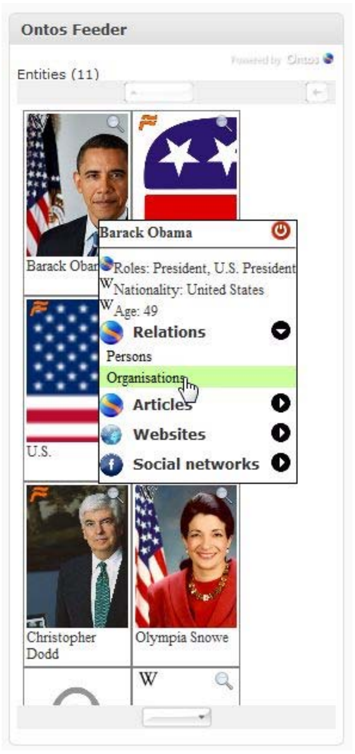


Fig. 2. The context information area is displayed next to the WYSIWYG editor and allows to navigate recursively to relevant contextual information from the Data Web

3 Architecture

While the server side consists of the OWS, the client side consists of the Core system and the CMS - adapters (see Figure 3). The core is CMS independent and can be embedded into a specific CMS by an appropriate adapter. Currently adapters for Drupal and WordPress are available.

Ontos Web Service (OWS) The Core system sends queries to the OWS. The Ontos Knowledge Base contains aggregated and refined information from around 1 million documents mainly from English online news. The Ontos Semantic Engine (NLP) extracts entities and their relationships from the text and disambiguates entities based on significance[1]. The significance is a complex measure based on the position of the occurrence in the text, the overall number of occurrences and

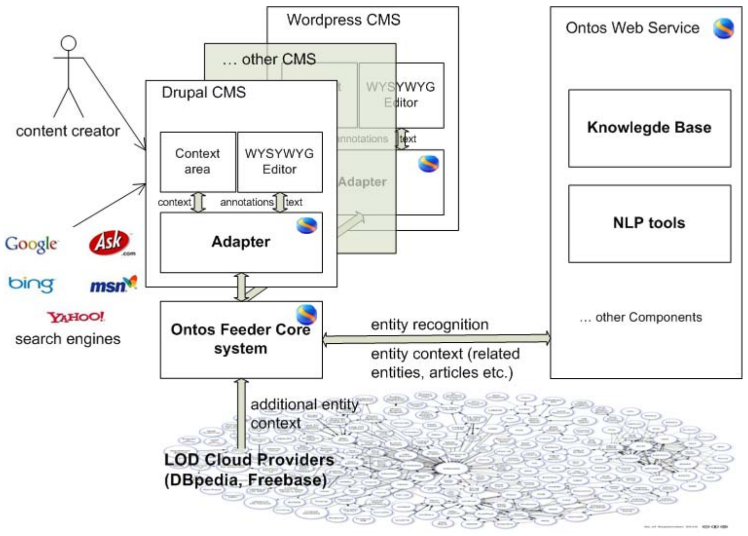


Fig. 3. Ontos Feeder overall architecture

the number of connected events and facts extracted from the text. The resulting information is returned to the Ontos Feeder.

Ontos Feeder. The Ontos Feeder currently supports requesting information for persons, organisations, locations and products, but can generally be extended to handle any of the entity types supported by the OWS. The user can configure, which types of entities the OWS should try to recognize in the provided text. The retrieval of each single piece of contextual information is encapsulated as a separate task by Ontos Feeder to increase the flexibility. The task engine supports task chaining, so if information could not be retrieved from a particular Linked Data source, it is requested from another one. The type of presented contextual information depends on the type of the recognized entity. The contextual information of a *Person* for example can consist of the age, nationality, status roles, connections to other persons and organisations, latest articles about this person, a Wikipedia article, a New York Times article, the personal homepage and a collection of different public social network profiles from Twitter or Facebook. Information about connections to other people and organisations, the status roles and the relevant articles are collected from the OWS. As every single information piece is requested by its own task, the variety of the presented contextual information can easily be adapted to personal needs.

4 Embedding Metadata

The OWS is able to annotate plain text as well as markup data such as HTML documents. The result is returned as a `stand-off annotation`, either in the form of start and end positions for text or an XPath expression for XML markup. A specialized annotation algorithm is used to: 1. highlight the annotations in the source HTML document in the editors. and 2. insert the annotations *inline* (as e.g. RDFa) into the HTML source of the article. Because all of the supported CMS WYSIWYG editors (currently FCKEditor and TinyMCE[5]) are capable of returning the current article as plain text, Ontos Feeder utilizes the Web Service in plain-text mode. As each of the editors have a different API, a special abstraction layer is put in front of the annotation algorithm to make it editor-independent. Furthermore, to make the annotation algorithm work faster for a plain-text document, all annotations are sorted in descended order and inserted bottom-up into the text. This avoids the recalculation of the annotation positions as compared to the top-down insertion. The annotation algorithm is capable of dealing with the entire supported semantic markup languages (RDFa and Microformats) and allows for annotation highlighting and on-the-fly binding of the contextual pop-up menu (see Figure 1).

5 Related Work and Summary

In recent years, several services have been published for suggesting annotations of tags to users. Among those services, OpenCalais and Zemanta are highly related to the Ontos Feeder as they also provide CMS integrations[6]. While Zemanta focuses on provide tag and link suggestions only, OpenCalais additionally extracts facts from the written text of the article. In contrast, the focus of the OWS is to provide disambiguated additional information, which is useful for the author. The data comes from the Ontos Knowledge Base and has been aggregated and fused from several sources. Also, the contribution of the Ontos Feeder, go well beyond the provision of a mere wrapper of a data service as it has a flexible, extensible architecture, is open-source and provides a context information area with recursive Linked Data navigation that aids the author. It transform stand-off annotations into inline RDFa and thus allows for a more fine-grained annotation method. Future work will be devoted to the area of co-referencing[2] for example by using OKKAM. Furthermore, it is planned, that users are able to define own vocabularies for named entity recognition, thus personalizing the annotation process.

References

1. Efimenko, I., Minor, S., Starostin, A., Drobyazko, G., Khoroshevsky, V.: Providing Semantic Content for the Next Generation Web. In: Semantic Web, pp. 39–62. InTech (2010)
2. 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] `http://ckeditor.com/` and `http://tinymce.moxiecode.com/`

[6] `http://drupal.org/project/[opencalais|zemanta]`

wayOU – Linked Data-Based Social Location Tracking in a Large, Distributed Organisation

Mathieu d'Aquin, Fouad Zablith, and Enrico Motta

Knowledge Media Institute, The Open University, Milton Keynes, UK
{m.daquin,f.zablith,e.motta}@open.ac.uk

Abstract. While the publication of linked open data has gained momentum in large organisations, the way for users of these organisations to engage with these data is still unclear. Here, we demonstrate a mobile application called wayOU (where are you at the Open University) which relies on the data published by The Open University (under data.open.ac.uk) to provide social, location-based services to its students and members of staff. An interesting aspect of this application is that, not only it consumes linked data produced by the University from various repositories, but it also contributes to it by creating new connections between people, places and other types of resources.

1 Introduction

The Open University[1] is a large UK University dedicated to distance learning. Apart from its main campus, it is distributed over 13 regional centres across the country. As part of the LUCERO project (Linking University Content for Education and Research Online[2]), the Open University is publishing linked open data, concerning people, publications, courses, places, and open educational material from existing institutional repositories and databases, under data.open.ac.uk.[3]

While the collection, conversion, exposure and maintenance of linked data in large organisations is slowly becoming easier, it is still an issue to get users of these organisations to engage with the data in a way suitable to them and that could also benefit to re-enforcing the data. Many 'applications' of linked data mostly concern the visualisation or exploration of available data for a particular purpose (see for example [1]), especially in mobile applications (see for example [2] or [3]), or the use of linked data to accomplish a specific task (e.g., recommendation in DBrec [4]). Our goal is to provide features to users that not only make use of linked data, but which usage would contribute in creating new connections in the data, including currently implicit relations between people and places.

We demonstrate wayOU (where are you at the Open University): a mobile application developed for the Google Android platform[4] that allows users of the

[1] http://www.open.ac.uk
[2] http://lucero-project.info
[3] http://data.open.ac.uk
[4] http://www.android.com

Open University (members of staff and students) to keep track of the places in which they have been on the main campus and the regional centres, and connect this information to other aspects of their interaction with the organisation (their workplace, meetings with colleagues, tutorials, etc.) This application relies on the data available through the SPARQL endpoint of data.open.ac.uk to get information regarding places (buildings, floors) and people (identifier, network), as well as the linked data published by the UK government regarding locations in the UK (especially, data.ordnancesurvey.co.uk[5]). More importantly, it generates and keeps track of information generated for each user regarding their current location, usual workplace, history of visits at the Open University and the reasons for these visits. In this way, it can be compared to the foursquare application[6] working in a specific 'corporate' environment and allowing to declare and expose more complex connections between users and places in this environment.

In the next section, we give a general overview of the structure of the wayOU application and in Section 3, we give more details about the way users interact with it. We discuss in Section 4 the future work and challenges we identified from our experience in building a social, mobile application based on linked data.

2 The wayOU Android Application

An overview of the architecture of the wayOU application is presented in Figure 1. The application can run on any Android enabled device (with at least Android OS 1.6 installed). This part of the application is mainly in charge of the interaction between the application and the user. It does not store any information locally (apart from basic settings such as the user's identifier). The information used and produced by the application is manipulated through an ad-hoc SPARQL client, accessing 2 SPARQL endpoints:

data.open.ac.uk provides an open SPARQL endpoint with information regarding the places in the Open University's campus and regional centres, especially regarding buildings and floors within buildings. It also contains information regarding members of the Open University (especially publications and projects), as well as about other resources such as courses, audio/video podcasts, open educational resources, etc. The SPARQL endpoint from data.ordnancesurvey.co.uk is used to obtain information regarding the location of buildings, based on their postcodes.[7]

A personal information store is used to store the information generated by the application for each user, and to query it. It deploys a service to update the store with RDF regarding the current location and activities of users and a SPARQL endpoint which identifies the appropriate graph to query depending on the identity of the user.

In addition, an option is available to export information from the private space of the personal information store, to the public one of data.open.ac.uk.

[5] `http://data.ordnancesurvey.co.uk`

[6] `http://foursquare.com`

[7] Postcode units in the UK identify small areas, often corresponding to a single street.

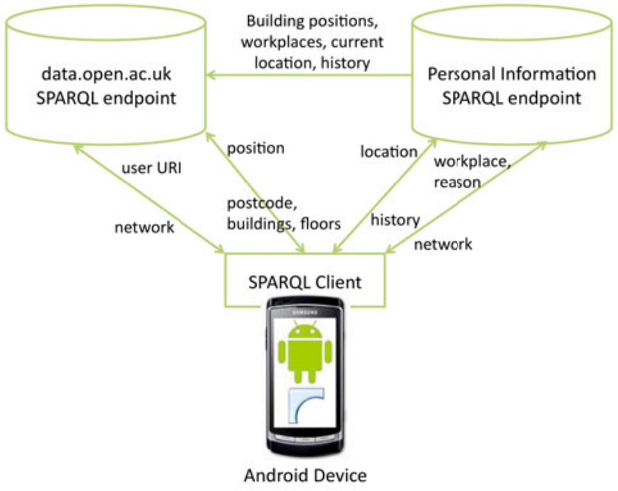
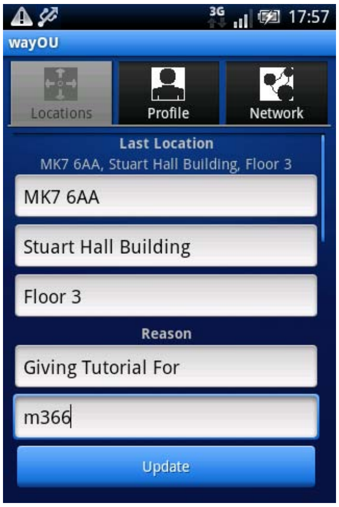

data.open.ac.uk SPARQL endpoint
Building positions, workplaces, current location, history
Personal Information SPARQL endpoint
user URI
network
position
postcode, buildings, floors
location
history
workplace, reason
network
SPARQL Client
Android Device

Fig. 1. Overview of the architecture of the wayOU Android application

3 Interface and Demonstration

In this section, we quickly go through the mobile interface of the wayOU application, as a way to explain its features and behaviour.

wayOU appears as a standard application on an Android-enabled mobile device. It does not require any setting other than the identifier of the user to work (the OUCU, Open University Computer Username, which is assigned to every member of staff and student of the University. See below).



Last Location
MK7 6AA, Stuart Hall Building, Floor 3

MK7 6AA

Stuart Hall Building

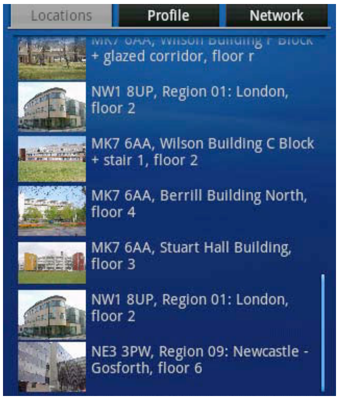
Floor 3

Reason

Giving Tutorial For

m366

The *Locations* tab is the main view of the application. It allows to enter the postcode, building name and floor level where the user is currently located. The postcode is automatically set to the closest postcode in which the Open University has buildings. Each field provides suggestions for auto-completion based on relevant data fetched from data.open.ac.uk: the building field suggests buildings in the chosen postcode and the floor field depends on the building field. In this view, it is also possible to provide a reason for attending a specific place. Predefined reasons are given with auto-completion, and include “Meeting with” somebody, “Giving a Tutorial” for a course and “Attending a tutorial” for a course. The second field provides values for auto-completion from the list of courses at The Open University or the list of people in the user’s network, depending on the chosen reason.



MK7 6AA, Wilson Building F Block + glazed corridor, floor r

NW1 8UP, Region 01: London, floor 2

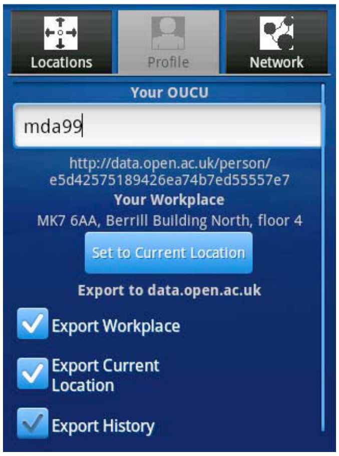
MK7 6AA, Wilson Building C Block + stair 1, floor 2

MK7 6AA, Berrill Building North, floor 4

MK7 6AA, Stuart Hall Building, floor 3

NW1 8UP, Region 01: London, floor 2

NE3 3PW, Region 09: Newcastle - Gosforth, floor 6



Your OUCU

mda99

http://data.open.ac.uk/person/e5d42575189426ea74b7ed55557e7

Your Workplace

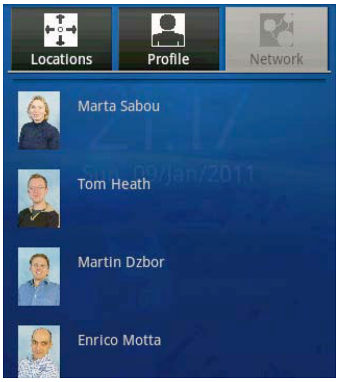
MK7 6AA, Berrill Building North, floor 4

Export to data.open.ac.uk

✓ Export Workplace

✓ Export Current Location

✓ Export History



Marta Sabou

Tom Heath

Martin Dzbor

Enrico Motta

The history of all the places attended by the user is also displayed as a list in the *Locations* tab. In this list, the user can quickly check whether he or she has already been in a given place, using the provided summary (postcode, building name, floor level) and the picture automatically attached to each item in the list. Each item links to a page describing the relationship between the user and this particular location, including the times he or she has attended it, whether or not it is his/her workplace, and reasons for attending otherwise.

The profile tab allows the user to connect the location information generated from the application to his/her identity at The Open University. The OUCU corresponds to the login of the user on the OU network, and is used to generate the URI identifying this particular user in the data.open.ac.uk linked data space. The user is also offered the possibility to declare his/her current location as his/her workplace, as well as to export the personal information generated by the application into the data.open.ac.uk space. This corresponds to exposing this information as public, so that it can be reused, for example, by the applications generating profile pages for members of staff of the University.

The notion of network is included in the application as a similar mechanism to the ones of 'friends' or 'contacts' in other social applications. Here, an initial list of members of the social network is populated from the list of collaborators of the user, derived from the co-authors of his/her papers, and people at the Open University working, or having worked, on the same projects. In addition, new members of the network can be derived based on the data generated by the application, because of users attending the same meetings or tutorials.

4 Conclusion, Future Work and Challenges

The wayOU application is currently being tested with selected users of the Open University. It will be made progressively available to more users before being fully opened, by the time of the conference. An interesting aspect of the wayOU application is that it both relies on and contributes to a base of linked data in a given organisation, and from which more information can be derived, possibly leading to new applications (e.g., analysing the movement of people within the organisation depending on their social network).

There are obvious challenges in building linked data-based mobile applications: while the mobile device is not required to realise complex computations, the overhead created by communicating with SPARQL endpoints represents a bottleneck. As the history of locations and the network of users grow, more and more time is needed to obtain and transfer the information. This and other challenges need be investigated in the future:

Dealing with complex privacy options. At the moment, the privacy options permitted by the application are rather simple: everything is private unless explicitly exported to data.open.ac.uk. More complex settings would be desirable, up to the possibility to define arbitrary complex access policies (for example: "give access to my current location on the campus, but not the regional centres, to people I have written papers with in the last 3 years").

Integrating with other social platforms. An obvious next step for this application is the integration with other social platforms, including for example exporting the current location to Twitter[8] or extending the user's network based on his/her Facebook[9] account. The Open University has developed several social network applications, including the Facebook *course profile* application[10] which could enhance and be enhanced by wayOU. Integrating such additional sources of information requires more effort as they are not based on linked data, and would make even more complex the privacy related issues described above.

A generic, reusable application. wayOU is developed specifically for users at the Open University and relies on linked data resources present at the Open University. However, adapting it for other organisations could be envisaged. Since it is designed to use the specific resources, vocabularies and URI Schemes of data.open.ac.uk, it is still unclear how much effort would be required.

References

1. Lehmann, J., Knappe, S.: DBpedia Navigator. In: ISWC Billion Triple Challenge (2008)
2. Becker, C., Bizer, C.: DBpedia mobile: A location-enabled linked data browser. In: Proceedings of Linked Data on the Web Workshop (LDOW 2008), Citeseer (2008)
3. van Aart, C., Wielinga, B., van Hage, W.R.: Mobile Cultural Heritage Guide: Location-Aware Semantic Search. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 257–271. Springer, Heidelberg (2010)
4. Passant, A., Decker, S.: Hey! Ho! Let's Go! Explanatory Music Recommendations with dbrec. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 411–415. Springer, Heidelberg (2010)

8 http://twitter.com
9 http://facebook.com
10 `http://www.facebook.com/apps/application.php?api_key=06d85b85540794e2fd02e9ef83206bf6`

SeaFish: A Game for Collaborative and Visual Image Annotation and Interlinking

Stefan Thaler[1], Katharina Siorpaes[1], David Mear[3],
Elena Simperl[1,2], and Carl Goodman[3]

[1] University of Innsbruck, STI-Innsbruck, Innsbruck, Austria
firstname.secondname@sti2.at
[2] Karlsruhe Institute of Technology, AIFB, Karlsruhe, Germany
firstname.secondname@kit.edu
[3] Pepper's Ghost Productions, London, UK
firstname.secondname@peppersghost.com

Abstract. Many tasks in semantic content creation, from building and aligning vocabularies to annotation or data interlinking, still require human intervention. Even though automatic methods addressing the aforementioned challenges have reached a certain level of maturity, user input is still required at many ends of these processes. The idea of human computation is to rely on the human user for problems that are impossible to solve for computers. However, users need clear incentives in order to dedicate their time and manual labor to tasks. The OntoGame series uses games to hide abstract tasks behind entertaining user interfaces and gaming experiences in order to collect knowledge. SeaFish is a game for collaborative image annotation and interlinking without text. In this latest release of the OntoGame series, players have to select images that are related to a concept that is represented by an image (from DBpedia) from a collection of images (produced by querying *flickr*^{TM} *wrappr* with the respective concept). The data collected by SeaFish is published as Linked Data on the Web. In this paper we outline the SeaFish game and demo.

Keywords: Games for semantic content creation, Image annotation, Linked Open Data, multimedia interlinking.

1 Motivation

The automated interpretation and description of images is still a major challenge. CAPTCHAs are still a prime example for a task that is trivial for a human user but impossible for a computer [6], allowing the distinction between a human user and a machine. The paradigm of human computation is also the foundation for "games with a purpose" [4] which aim to exploit human intelligence for solving computationally difficult tasks by masquerading them as games. Thereby, an abstract (knowledge acquisition or curation) task is not only hidden behind an attractive, easy-to-understand user interface but users are also given incentives to dedicate their time to solving the task. Playing, competition, and social recognition are core motivational mechanisms of games. The OntoGame series uses casual games in order to acquire human input for several semantic content creation tasks, such as ontology building (e.g. the OntoPronto game), video annotation (e.g. the OntoTube game), or ontology alignment (e.g. the SpotTheLink game).

In this paper we present the SeaFish game and describe its demo as planned for ESWC 2011. SeaFish is the latest release of the OntoGame series. SeaFish is a game for collaborative image annotation and interlinking without text. Players have to select images that are related to a concept that is represented by an image (from DBpedia) from a collection of images (produced by querying *flickr^{TM}wrappr*[1] with the respective concept). Players get points based on their degree of consensus with the player community. The data collected by SeaFish is published as Linked Data on the Web.

SeaFish does not involve any reading for annotation. It solely relies on images. This means that the game can be played in a fast pace as reading takes more time than just looking at a visual representation of a concept. Moreover, the game is not constrained by language or reading limitations. The resulting data is published as Linked Data and thus, SeaFish implements game-based multimedia interlinking as described by Buerger and colleagues[2].

Related work. We sketch related work focusing on games for the creation of image annotations. A comprehensive list of games for knowledge acquisition is available at the INSEMTIVES homepage[2]. Luis von Ahn introduced the ESP game[7] aka Google's Image Labeler for annotating images. Additioinally, he created other serious games that are published at the GWAP homepage[3]. PopVideo[4] aims to label videos and Waisda[5] allows players to tag live TV shows. Magic Bullet[9] evaluates the robustness of Captchas, TagCaptcha[3] produces image annotations as side product of the CAPTCHA process. Finally, Picture This [6] is another game for annotating images, however, unlike the ESP game users don't create tags but have to decide on the best matching tag.

2 Collaborative and Game-Based Image Annotation

SeaFish SeaFish is a single player game where players have to fish related images that are floating around. Users can play from the OntoGame website[7] as either registered user or as anonymous guest. Players may also login via Facebook by visiting the OntoGame Facebook page[8]. After they clicked the SeaFish! button

[1] *flickr^{TM}wrappr*,http://www4.wiwiss.fu-berlin.de/flickrwrappr/
[2] INSEMTIVES, http://www.insemtives.eu/games.php
[3] Games with a Purpose, www.gwap.com
[4] www.gwap.com/popvideo
[5] Waisda, http://waisda.nl/
[6] Picture This
[7] OntoGame, http://www.ontogame.org/games
[8] OntoGame Facebook page, http://apps.facebook.com/ontogame

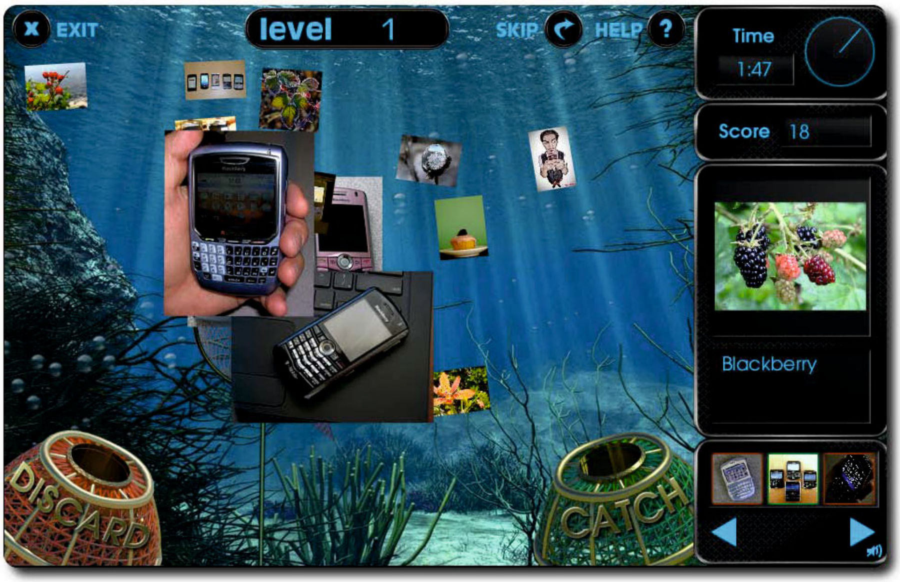
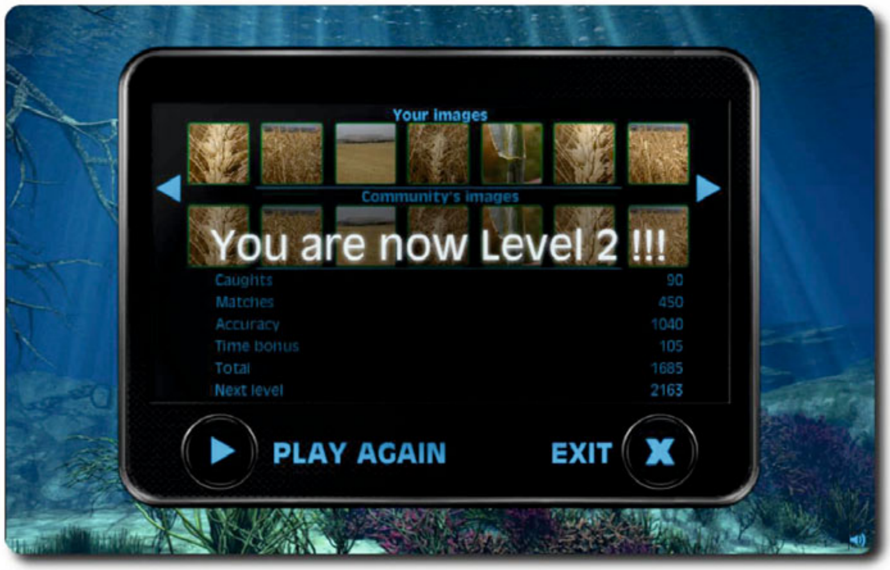


Fig. 1. SeaFish - in game screen

the game starts. From there players may either get playing instructions or start a game. Each game round is about a concept taken from DBpedia that is represented by an image. Players see this image on the right hand side. Additionally, players see the result of an search for the concept on $flickr^{TM}wrappr$, which are floating through the main screen (see Figure 1). Players have 2 minutes to mark those images as either related to the concept or unrelated to the concept. They can do so by catching the images with the fishing rod and dragging them either to the green basket (related) or the red basket (not related).

Players always obtain a small reward for catching an image. Generally, players get more points depending on whether their decision is consensual with decisions of the majority of other players. Additionally, the reward a player gets always depends on their reliability as well as their level. When an image is caught a new image appears in order to have less than ten images floating on the screen at the same time. The game stops when all images have been fished or the time is up. After each game round (see Figure 2) players can compare their answers with the communitys answers as well as view statistics on accuracy and time.

To give an example of Seafish: in a game round the player is shown an image of *Blackberry* (see Figure 1) on the right hand side (retrieved from DBpedia). As a result of the query *Blackberry* on $flickr^{TM}wrappr$, pictures are floating around on the main screen. The task of the player is to select those image that are related to the concept Blackberry(the berry) and those that are not by putting images in the "Discard" or "Catch" baskets on the bottom of the screen.



You are now Level 2 !!!

Caughts	90
Matches	450
Accuracy	1040
Time bonus	105
Total	1685
Next level	2163

Fig. 2. SeaFish - comparison screen

Data export. SeaFish exports annotations as RDF triples. In our first scenario we create annotations of the form <image><http://xmlns.com/foaf/spec/depiction><concept>as well as the inverse property. Our concepts are taken from DBpedia plus follow the four Linked Open Data principles[1]. This means that our data may be used to contribute to the Linked Open Data cloud without being modified.

Our annotations are generated from stored answers in the following manner: let s_i be an answer that states an image is related to a concept, i a number greater than six, r_i the reliability of the player giving the answer at the time of the answer and n the number of total answers stored about an image. A player's reliability is a measurement of how accurate this player played in previous games.

3 Discussion and Outlook

Currently, three months after the initial release of SeaFish we have collected feedback, implemented the proposed changes and re-released the game. A still open issue is the lack of returning player. To counter this we have integrated leveling mechanisms. We have integrated it on Facebook to profit from the social network effect. We are currently evaluating the outcome of these measures. Besides, we also evaluate the effect of tweaking the rewarding function on the quality of the generated annotations.

We also identified a possible extension of SeaFish: Ookaboo[9] is an API that links pictures to concepts from different Linked Open Data sources and makes them searchable. SeaFish could on the one hand be used to verify the data provided on Ookaboo. On the other hand it could also be used to provide new image tags.

Another possibility to attract more players is a planned feature to allow users to add their own category of concepts to play in the games.

4 Conclusion

At ESWC 2011 the audience of the demo players will be able to play the games of the OntoGame series (including Seafish) and can see how and which data is generated. In this paper, we have described the SeaFish game for collaborative image annotation. We are currently collecting massive user input in order to thoroughly evaluate the game by assessing the quality of generated data and the user experience.

Acknowledgments

The work presented has been funded by the FP7 project INSEMTIVES under EU Objective 4.3 (grant number FP7-231181).

References

1. Berners-Lee, T.: Linked data - design issues (2006)
2. Hausenblas, M., Troncy, R., Raimond, Y., Bürger, T.: Interlinking multimedia: How to apply linked data principles to multimedia fragments. In: Linked Data on the Web Workshop, LDOW 2009 (2009)
3. Morrison, D., Marchand-Maillet, S., Bruno, E.: Tagcaptcha: annotating images with captchas. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP 2009, pp. 44–45. ACM, New York (2009)
4. Von Ahn, L.: Games with a purpose. IEEE Computer 29(6), 92–94 (2006)
5. Von Ahn, L.: Peekaboom: A Game for Locating Objects in Images (2006)
6. Von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security (2003)
7. Ahn, L.v., Dabbish, L.: Labeling images with a computer game. In: CHI, pp. 319–326 (2004)
8. von Ahn, L., Ginosar, S., Kedia, M., Blum, M.: Improving Image Search with PHETCH. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007, vol. 4, pp. IV-1209 –IV-1212 (2007)
9. Yan, J., Yu, S.-Y.: Magic bullet: a dual-purpose computer game. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP 2009, pp. 32–33. ACM, New York (2009)

[9] Ookaboo, http://ookaboo.com/

The Planetary System: Executable Science, Technology, Engineering and Math Papers

Christoph Lange, Michael Kohlhase, Catalin David, Deyan Ginev, Andrea Kohlhase, Bogdan Matican, Stefan Mirea, and Vyacheslav Zholudev

Computer Science, Jacobs University Bremen, Germany
{ch.lange,m.kohlhase,c.david,d.ginev,a.kohlhase, b.matican,s.mirea,v.zholudev}@jacobs-university.de

Abstract. Executable scientific papers contain not just layouted text for reading. They contain, or link to, machine-comprehensible representations of the scientific findings or experiments they describe. Client-side players can thus enable readers to "check, manipulate and explore the result space" [1]. We have realized executable papers in the STEM domain with the Planetary system. Semantic annotations associate the papers with a content commons holding the background ontology, the annotations are exposed as Linked Data, and a frontend player application hooks modular interactive services into the semantic annotations.

1 Application Context: STEM Document Collections

The Planetary system [2] is a semantic social environment for document collections in Science, Technology, Engineering and Mathematics (STEM). STEM documents have in common that they describe concepts using mathematical formulæ, which are composed from mathematical symbols – operators, functions, etc. –, which have again been defined as more foundational mathematical concepts in mathematical documents. Thus, there is a dynamically growing ontology of domain knowledge. The domain knowledge is structured along the following, largely independent dimensions [3,4]: (i) logical and functional structures, (ii) narrative and rhetorical document structures, (iii) information on how to present all of the former to the reader (such as the notation of mathematical symbols), (iv) application-specific structures (e.g. for physics), (v) administrative metadata, and (vi) users' discussions about artifacts of domain knowledge.

We have set up Planetary instances for the following paradigmatic document collections: (i) a browser for the ePrint arχiv [5], (ii) a reincarnation of the PlanetMath mathematical encyclopledia [6] (where the name Planetary comes from), (iii) a companion site to the general computer science (GenCS) lecture of the second author [7,8], and (iv) an atlas of theories of formal logic [9]. This list is ordered by increasing machine-comprehensibility of the representation and thus, as explained below, by increasing "executability" of the respective papers. All instances support browsing and fine-grained discussion. The PlanetMath and GenCS collections are editable, as in a wiki[1], whereas the arχiv and Logic Atlas

[1] Planetary reuses technology of our earlier semantic wiki SWiM [10].

corpora have been imported from external sources and are presented read-only. We have prepared demos of selected services in all of these instances.

2 Key Technology: Semantics-Preserving Transformations

Documents published in PLANETARY become flexible, adaptive interfaces to a *content commons* of domain objects, context, and their relations. This is achieved by providing an integrated user experience through a set of interactions with documents based on an extensible set of client- and server side services that draw on explicit (and thus machine-understandable) representations in the content commons. We have implemented or reused ontologies for all structures of STEM knowledge ([4] gives an overview). Annotations of papers with terms from these ontologies act as hooks for local interactive services. By translation, PLANETARY makes the structural ontologies editable in the same way as the papers, so that the community can adapt and extend them to their needs.

The sources of the papers are maintained in LaTeX or the semantic mathematical markup language OMDoc [11]. For querying and information retrieval, and interlinking with external knowledge – including discussions about concepts in the papers, but also remote Linked Datasets –, we extract their semantic structural outlines to an RDF representation, which is accessible to external services via a SPARQL endpoint and as Linked Data [8]. For human-comprehensible presentation, we transform the sources to XHTML+MathML+SVG [8]. These papers gain their "executability" from embedded semantic annotations: Content MathML[2] embedded into formulæ [13], and an RDFa subgraph of the above-mentioned RDF representation embedded into XHTML and SVG.

The amount of semantic annotations depends on the source representation: (i) The arχiv corpus – 500+K scientific publications – has LaTeX sources, most of which merely make the section structure of a document machine-comprehensible, but hardly the fine-grained functional structures of mathematical formulæ, statements (definition, axiom, theorem, proof, etc.), and theories. We have transformed the papers to XHTML+MathML, preserving semantic properties like formula and document structure [5]. (ii) The PlanetMath corpus is maintained inside PLANETARY; it additionally features subject classification metadata and semi-automatically annotated concept links [14], which we preserve as RDFa. (iii) The GenCS corpus is maintained in sTeX, a semantics-extended LaTeX [15], inside PLANETARY. sTeX makes explicit the functional structures of formulæ, statements, and theories, narrative and rhetorical structures, information on notation, as well as – via an RDFa-like extensibility – arbitrary administrative and application-specific metadata. This structural markup is preserved as Content MathML and RDFa in the human-comprehensible output. In this translation, OMDoc, an XML language semantically equivalent to sTeX, serves as an intermediate representation. (iv) The Logic Atlas is imported into PLANETARY from an external OMDoc source but otherwise treated analogously to the GenCS corpus.

[2] Or the semantically equivalent OpenMath [12].

3 Demo: Interactive Services and the Planetary API

Our demo focuses on how PLANETARY makes STEM papers executable – by hooking interactive services into the annotations that the semantics-preserving translations put into the human-comprehensible presentations of the papers. Services are accessible locally via a context menu for each object with (fine-grained) semantic annotations – e.g. a subterm of a formula –, or via the "`InfoBar`", as shown in fig. 1. The menu has one entry per service available in the current context; the `InfoBar` indicates the services available for the information objects in each line of the paper. In the image on the right of fig. 1, we selected a subterm and requested to fold it, i.e. to simplify its display by replacing it with an ellipsis. The `FoldingBar` on the left, similar to source code IDEs, enables folding document structures, and the `InfoBar` icons on the right indicate the availability of local discussions. Clicking them highlights all items with discussions; clicking any of them yields an icon menu as shown in the center. The icon menu for the discussion service allows for reporting problems or asking questions using a STEM-specifically extended argumentation ontology [16]. The richer semantic markup of the GenCS and Logic Atlas collections enable services that utilize logical and functional structures – reflected by a different icon menu. Fig. 2 demonstrates looking up a definition and exploring the prerequisites of a concept. The definition lookup service obtains the URI of a symbol from the annotation of a formula and queries the server for the corresponding definition. The server-side part of the prerequisite navigation service obtains the transitive closure of all dependencies of a given item and returns them as an annotated SVG graph. Computational services make mathematical formulæ truly executable: The user can send a selected expression to a computer algebra web service for evaluation or graphing [17], or have unit conversions applied to measurable quantities [18]. Finally, besides these existing services, we will demonstrate the ease of realizing additional services – within the PLANETARY environment or externally of it. The API for services running as scripts in client-side documents is essentially defined

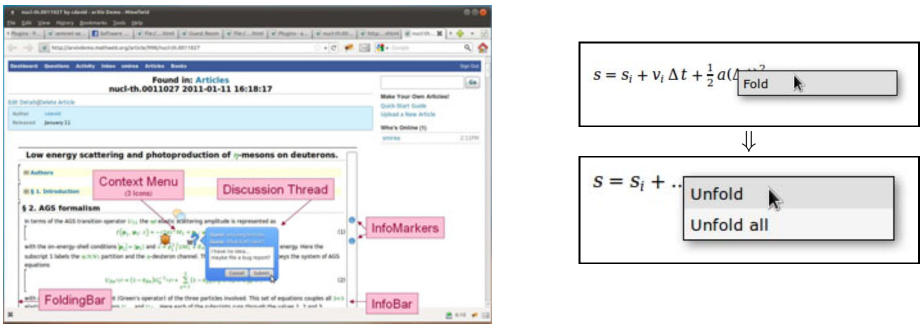
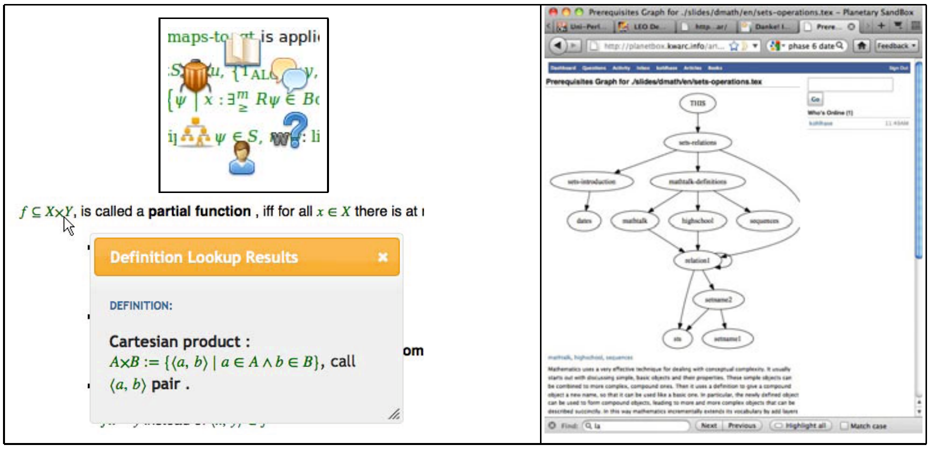

Low energy scattering and photoproduction of η-mesons on deuterons.
Context Menu
(3 Icons)
Discussion Thread
InfoMarkers
FoldingBar
InfoBar
$s = s_i + v_i \Delta t + \frac{1}{2} a(\Delta$
Fold
⇓
$s = s_i + ..$
Unfold
Unfold all

Fig. 1. Interacting with an arχiv article via `FoldingBar`, `InfoBar`, and localized discussions. On the right: localized folding inside formulæ


$f \subseteq X \times Y$, is called a **partial function** , iff for all $x \in X$ there is at

Definition Lookup Results

DEFINITION:

Cartesian product :
$A \times B := \{\langle a, b\rangle \mid a \in A \wedge b \in B\}$, call $\langle a, b\rangle$ **pair** .

Prerequisites Graph for ./slides/dmath/en/sets-operations.tex

mathtalk, highschool, sequences

Mathematics uses a very effective technique for dealing with conceptual complexity. It usually starts out with discussing simple, basic objects and their properties. These simple objects can be combined to more complex, compound ones. Then it uses a definition to give a compound object a new name, so that it can be used like a basic one. In particular, the newly defined object can be used to form compound objects, leading to more and more complex objects that can be described succinctly. In this way mathematics incrementally extends its vocabulary by add layers

Fig. 2. Definition Lookup and Prerequisites Navigation

by the in-document annotations, the underlying structural ontologies that are retrievable from the content commons, the possibility to execute queries against the content commons, and the extensibility of the client-side user interface.

4 Related Work

Like a **semantic wiki**, Planetary supports editing and discussing resources. Many wikis support LaTeX formulæ, but without fine-grained semantic annotation. They can merely *render* formulæ in a human-readable way but not make them executable. The Living Document [19] environment enables users to **annotate and share life science documents** and interlink them with Web knowledge bases, turning – like Planetary – every single paper into a portal for exploring the underlying network. However, life science knowledge structures, e.g. proteins and genes, are relatively flat, compared to the tree-like and context-sensitive formulæ of STEM. State-of-the-art **math e-learning systems**, including ActiveMath [20] and MathDox [21], also make papers executable. However, they do not preserve the semantic structure of these papers in their human-readable output, which makes it harder for developers to embed additional services into papers.

5 Conclusion and Outlook

Planetary makes documents executable on top of a content commons backed by structural ontologies. Apart from mastering semantic markup – which we alleviate with dedicated editing and transformation technology – document authors, as well as authors of structural ontologies, only need expertise in their own domain. In particular, no system level programming is necessary: The semantic representations act as a high-level conceptual interface between content authors and the system and service developers. Even developers can realize considerably new services as a client-side script that runs a query against the content commons. This separation of concerns ensures a long-term compatibility of the knowledge hosted in a Planetary instance with future demands.

References

1. Executable Paper Challenge, http://www.executablepapers.com
2. David, C., et al.: eMath 3.0: Building Blocks for a social and semantic Web for online mathematics & ELearning. In: Workshop on Mathematics and ICT (2010), http://kwarc.info/kohlhase/papers/malog10.pdf
3. Kohlhase, A., Kohlhase, M., Lange, C.: Dimensions of formality: A case study for MKM in software engineering. In: Autexier, S., Calmet, J., Delahaye, D., Ion, P.D.F., Rideau, L., Rioboo, R., Sexton, A.P. (eds.) AISC 2010. LNCS(LNAI), vol. 6167, pp. 355–369. Springer, Heidelberg (2010)
4. Lange, C.: Ontologies and Languages for Representing Mathematical Knowledge on the Semantic Web. Submitted to Semantic Web Journal, http://www.semantic-web-journal.net/underreview
5. arXMLiv Build System, http://arxivdemo.mathweb.org
6. PlanetMath Redux, http://planetmath.mathweb.org
7. Kohlhase, M., et al.: Planet GenCS, http://gencs.kwarc.info
8. David, C., Kohlhase, M., Lange, C., Rabe, F., Zhiltsov, N., Zholudev, V.: Publishing math lecture notes as linked data. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 370–375. Springer, Heidelberg (2010)
9. Logic Atlas and Integrator, http://logicatlas.omdoc.org
10. Lange, C.: SWiM – A semantic wiki for mathematical knowledge management. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 832–837. Springer, Heidelberg (2008)
11. Kohlhase, M.: OMDoc An open markup format for mathematical documents [Version 1.2]. LNCS (LNAI), vol. 4180. Springer, Heidelberg (2006)
12. Open Math 2.0. (2004), http://www.openmath.org/standard/om20
13. MathML 3.0., http://www.w3.org/TR/MathML3
14. Gardner, J., Krowne, A., Xiong, L.: NNexus: Towards an Automatic Linker for a Massively-Distributed Collaborative Corpus. IEEE Transactions on Knowledge and Data Engineering 21.6 (2009)
15. Kohlhase, A., Kohlhase, M., Lange, C.: sTeX – A System for Flexible Formalization of Linked Data. In: I-Semantics (2010)
16. Lange, C., et al.: Expressing Argumentative Discussions in Social Media Sites. In: Social Data on the Web Workshop at ISWC (2008)
17. David, C., Lange, C., Rabe, F.: Interactive Documents as Interfaces to Computer Algebra Systems: JOBAD and Wolfram|Alpha. In: CALCULEMUS, Emerging Trends (2010)
18. Giceva, J., Lange, C., Rabe, F.: Integrating web services into active mathematical documents. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) MKM 2009, Held as Part of CICM 2009. LNCS(LNAI), vol. 5625, pp. 279–293. Springer, Heidelberg (2009)
19. García, A., et al.: Semantic Web and Social Web heading towards Living Documents in the Life Sciences. In: Web Semantics 8.2–3 (2010)
20. ActiveMath, http://www.activemath.org
21. MathDox Interactive Mathematics, http://www.mathdox.org

Semantic Annotation of Images on Flickr*

Pierre Andrews, Sergey Kanshin, Juan Pane, and Ilya Zaihrayeu

Department of Information and Communication Technology
University of Trento Italy
{andrews,kanshin,pane,ilya}@disi.unitn.it

Abstract. In this paper we introduce an application that allows its users to have an explicit control on the meaning of tags they use when uploading photos on Flickr. In fact, this application provides to the users an improved interface with which they can add concepts to photos instead of simple free-text tags. They can thus directly provide semantic tags for their photos that can then be used to improve services such as search.

1 Introduction

The task of discovering the semantics of photos is still very difficult and hence, automatic annotation with concepts "is widely recognized as an extremely difficult issue" [1]. It is thus preferable to ask the creators of the photos to annotate them directly when they share them. As was demonstrated in [2], simple free-text annotations are not sufficient for performing good indexing and leveraging semantic search.

In this paper we discuss an extension to a popular open source photo uploader for the Flickr[1] website that allows the annotation of image files with semantic annotations without extra involvement from the user. One of the feature of this tool is the bootstrapping of semantic annotations by extraction of the intrinsic semantics contained in the context in which the images reside on the local computer of the user before uploading them to Flickr by using the technology described in [3]. The users can also manually provide semantic annotations through an extended interface. These semantic annotations are linked to their meaning in a knowledge organisation system such as Wordnet[2].

The source code for the tools described in this paper is available freely at `https://sourceforge.net/projects/insemtives/`.

2 Semantic Annotation Platform

The Semantic Annotation tool is built as an application on top of the INSEMTIVES platform. Interested readers are referred to [4] for details on this platform.

* This work has been partly supported by the INSEMTIVES project (FP7-231181, see http://www.insemtives.eu).

[1] `http://www.flickr.com`

[2] `http://wordnet.princeton.edu/`

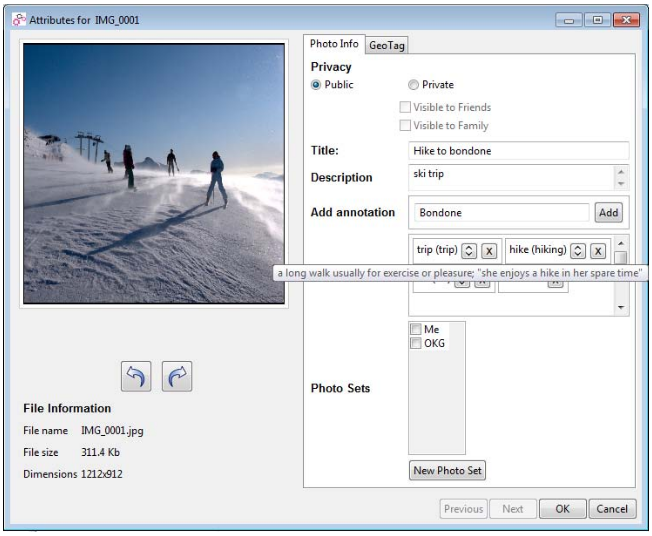

Attributes for IMG_0001

Photo Info | GeoTag

Privacy: [✓ Public] Private

Visible to Friends

Visible to Family

Title: Hike to bondone

Description: ski trip

Add annotation: Bondone

trip (trip) | hike (hiking)

a long walk usually for exercise or pleasure; "she enjoys a hike in her spare time"

Photo Sets: Me, OKG

File Information

File name IMG_0001.jpg

File size 311.4 Kb

Dimensions 1212x912

Fig. 1. Image properties with added concepts

The platform services are exposed through the INSEMTIVES platform API's, which can then be used by third party applications. The API's are based on a communication framework which represents a highly modular client-server architecture implemented using communication layer over JMS[3] messaging protocol as well as six other protocols, including REST[4] and Web Service Notification. The platform is divided in two main components: *Structured Knowledge* and *User and Communities*. The Structured Knowledge component stores all artifacts in RDF following the semantic annotation model defined in [5]. The semantic store relies on OWLIM[5] while the initial lexical resource is based on Wordnet and DBPedia. The User and Communities component contains user profiles and is responsible for maintaining provenance information of the annotations and resources in the platform.

3 Semantic Photo Annotation

The semantic annotation application has been developed as an extension to the JUploadr[6] uploading tool for Flickr. This tool has been chosen as it is open source and developed in Java which is the main development language for the INSEMTIVES platform and makes the tool easily multi-platform.

[3] Java Messaging Service, see `http://java.sun.com/products/jms/`
[4] `http://en.wikipedia.org/wiki/Representational_State_Transfer`
[5] `http://www.ontotext.com/owlim/`
[6] `http://www.juploadr.org/`

jUploadr allows the user to queue photos for uploading on Flickr and set their properties (see Figure 1). In particular, the tags, description and title, which are the main metadata stored and displayed by the Flickr website.

3.1 Semi-automatic Semantic Annotation

Concept Tagging The INSEMTIVES Uploadr tool will try to understand where the users have stored their photos and automatically add tags to them. For instance, if the users have stored their photos in the folder: *.../Pictures/Trips/Trentino/Hike to Bondone-10-01-10/IMG_0001.jpg* then the software will understand that these were photos of a trip in the Trentino's mountains and will automatically propose relevant *concepts* (see Figure 1).

When a new photo is added to the upload batch, the following processing takes place:

1. clean and split the photo path in *phrases*,
2. send the ordered set of *phrases* to the INSEMTIVES platform,
3. receive a set of *tokens* for each *phrase* with the corresponding *concepts*,
4. attach each identified *concept* to the photo to be uploaded as a semantic annotation.

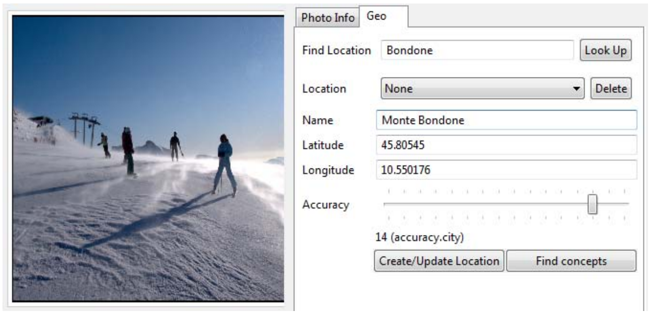
The cleaning of the path is performed to remove generic parts of the path that might not contain interesting semantic information for bootstrapping. The cleaning is specific to that particular type of media and the one applied for image files is the following: 1. if the path starts with the name of the user's home directory, drop the beginning. 2. if the path starts with /media or /mnt, drop the beginning. 3. if the path contains the word "picture(s)", "Media", "Photos" drop anything on the left of that word. Including that word. 4. treat _ as spaces.

Each section of the path (between path separators) is treated as a *phrase* – i.e., a short sentence made of related tokens. The set of phrases in the path is sent in an ordered way to the NLP service to be used as a context for disambiguation. For instance, "Hike to Bondone" will be treated in the context of "Trip" and thus "hike" will be disambiguated to "a long walk usually for exercise or pleasure" instead of "the amount a salary is increased"[7](see Figure 1).

These special tags have a "meaning" (or "semantics") attached to them, a summary of that meaning is always displayed next to the concept and if the user hovers the mouse over one of them, it's definition is displayed (see Figure 1).

Location Tagging. In the "geo" tab of the properties window for a photo, the user can specify where the photo was taken. Once a location is specified for a photo, the user can press the "Find concepts" button to ask the INSEMTIVES tool to find automatically new concepts that relate to the location where the photo was taken by looking at popular terms already used for this location on Flickr. These concepts will then be added to the list of semantic tags in the "Photo Info" tab, where the user can remove the ones that might not fit the photo.

[7] These senses are taken from WordNet.

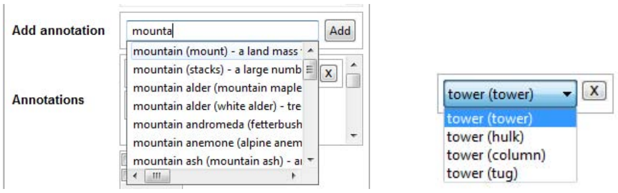


Find Location: Bondone
Location: None
Name: Monte Bondone
Latitude: 45.80545
Longitude: 10.550176
Accuracy: 14 (accuracy.city)

Fig. 2. Specifying Location of an Image

3.2 Manual Semantic Annotation

The user can also provide manual semantic annotations. When typing a free-text tag, the user is given a list of possible concepts from the platform knowledge base. Each concept is disambiguated by a summary word (see Figure 3.2a) and hovering it provides its definition.



mounta

mountain (mount) - a land mass
mountain (stacks) - a large numb
mountain alder (mountain maple
mountain alder (white alder) - tre
mountain andromeda (fetterbush
mountain anemone (alpine anem
mountain ash (mountain ash) - ai

tower (tower)

tower (tower)
tower (hulk)
tower (column)
tower (tug)

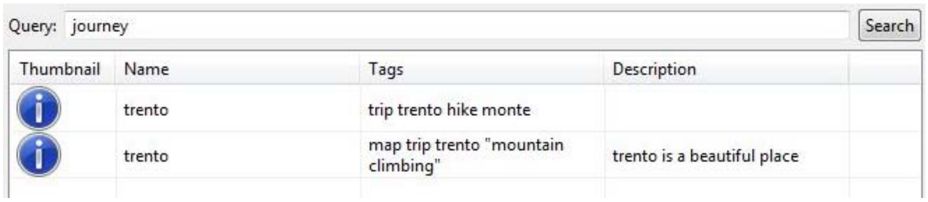
a) Concept Completion b) Concept Disambiguation

For the concepts that were automatically proposed by the services described earlier, the user can correct the disambiguation by selecting the right sense from a drop-down list. Here also, a summary is shown and the user can display the definition of the concept (see Figure 3.2b)) .

4 Semantic Search

On the Flickr website, photos can only be searched by keywords, hence, if a photo is tagged with "cat" and another one with "dog", these photos will not be found when searching for "animal".

However, if the photos were uploaded with the INSEMTIVES Uploadr and were assigned the *concepts* "cat" and "dog", then they can be retrieved with a semantic search for "animal". To do this, the INSEMTIVES Uploadr provides a specific search interface as shown in Figure 3.


Query: journey

Thumbnail	Name	Tags	Description
	trento	trip trento hike monte	
	trento	map trip trento "mountain climbing"	trento is a beautiful place

Fig. 3. Example of a Semantic Search. A search for "journey" found photos about a "trip".

5 Demonstration

The goal of the demonstration will be to show the extension of the standard open source jUploadr application with the specific semantic annotation tools. The demonstration will be split in two main scenarios:

Annotation of Photos the new tools for semantic annotation of images will be demonstrated in a scenario showing the use of automatic concept extraction from the local context, recommendation of concepts from the location and manual input of concepts to annotate a photo.

Semantic Search for Photos once some photos have been annotated and uploaded on Flickr, we will demonstrate the semantic search tool that is able to retrieve photos, no only on the tags attached to them, but also on the concepts used for annotation, thus finding related photos by mapping synonymous terms to the same concept and by reasoning about the subsumption relationship between concepts.

The visitors will also be shown how to download and install the tool for their own use with their Flickr accounts so that they can upload ESWC'11 photos with semantic annotations.

References

1. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40, 5:1–5:60 (2008)
2. Andrews, P., Pane, J., Zaihrayeu, I.: Semantic disambiguation in folksonomy: a case study. In: Bernardi, R., Chambers, S., Gottfried, B., Segond, F., Zaihrayeu, I. (eds.) Advanced Language Technologies for Digital Libraries. LNCS Hot Topic subline. Springer, Heidelberg (2011)
3. Zaihrayeu, I., Sun, L., Giunchiglia, F., Pan, W., Ju, Q., Chi, M., Huang, X.: From web directories to ontologies: Natural language processing challenges. 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. 623–636. Springer, Heidelberg (2007)
4. Siorpaes, K., Konstantinov, M., Popov, B.: Requirement analysis and architectural design of semantic content management platform. Technical report, Insemtives.eu (September 2009)
5. Andrews, P., Zaihrayeu, I., Pane, J., Autayeu, A., Nozhchev, M.: Report on the refinement of the proposed models, methods and semantic search. Technical report, Insemtives.eu (November 2010)

FedX: A Federation Layer for Distributed Query Processing on Linked Open Data

Andreas Schwarte[1], Peter Haase[1], Katja Hose[2],
Ralf Schenkel[2], and Michael Schmidt[1]

[1] fluid Operations AG, Walldorf, Germany
[2] Max-Planck Institute for Informatics, Saarbrücken, Germany

Abstract. Driven by the success of the Linked Open Data initiative today's Semantic Web is best characterized as a Web of interlinked datasets. Hand in hand with this structure new challenges to query processing are arising. Especially queries for which more than one data source can contribute results require advanced optimization and evaluation approaches, the major challenge lying in the nature of distribution: Heterogenous data sources have to be integrated into a federation to globally appear as a single repository. On the query level, though, techniques have to be developed to meet the requirements of efficient query computation in the distributed setting. We present FedX, a project which extends the Sesame Framework with a federation layer that enables efficient query processing on distributed Linked Open Data sources. We discuss key insights to its architecture and summarize our optimization techniques for the federated setting. The practicability of our system will be demonstrated in various scenarios using the Information Workbench.

1 Introduction

Motivated by the ongoing success of the Linked Open Data initiative and the growing amount of semantic data sources available on the Web, new approaches to query processing are emerging. While query processing in the context of RDF is traditionally done locally using centralized stores, recently one can observe a paradigm shift towards federated approaches which can be attributed to the decentralized structure of the Semantic Web. The Linked Open Data cloud - representing a large portion of the Semantic Web - comprises more than 200 datasets that are interlinked by RDF links. In practice many scenarios exist where more than one data source can contribute information, making query processing more complex. Contrary to the idea of Linked Data, centralized query processing requires to copy and integrate relevant datasets into a local repository. Accounting for the structure, the natural approach to follow in such a setting is federated query processing over the distributed data sources.

While there exist efficient solutions to query processing in the context of RDF for local, centralized repositories [7,5], research contributions and frameworks for distributed, federated query processing are still in the early stages. In practical terms the Sesame framework in conjunction with AliBaba[1] is one possible sample solution allowing for federations of distributed repositories and endpoints. However, benchmarks have shown poor performance for many queries in the federated setup due to the absence of advanced optimization techniques [6]. From the research community DARQ [9] and Networked Graphs [10] contribute approaches to federated SPARQL queries and federated integration. Since both require proprietary extensions to languages and protocols which are not supported by most of today's endpoints, they are not applicable in practical environments.

In this demonstration paper we present FedX, a practical framework for transparent access to data sources through a federation. The framework offers efficient query processing in the distributed setting, while using only protocols and standards that are supported by most of today's data sources.

In the following we will describe the FedX system and give a demonstration of its practical applicability in the Information Workbench. In section 2 we give some insights into the federation layer. Next, in section 3 we present the demonstration scenario. Finally, we conclude with some remarks on future work.

2 FedX - Design and System Overview

FedX[2] is being developed to provide an efficient solution for distributed query processing on Linked Open Data. It is implemented in Java and extends the Sesame framework with a federation layer. FedX is incorporated into Sesame as a `SAIL` (`S`torage `a`nd `I`nference `L`ayer), which is Sesame's mechanism for allowing seamless integration of standard and customized RDF repositories. The underlying Sesame infrastructure enables heterogeneous data sources to be used as endpoints within the federation. FedX implements the logics for optimization and efficient execution of the query in the distributed setting.

Figure 1 shows the architecture of an application built on top of FedX. The application layer provides the frontend to the query processing engine and is necessary for any kind of interaction with the federation. We decided to employ the Information Workbench [8] as such for our demonstration (see section 3). However, any other application can be used as well by utilizing the Sesame API.

The second layer is composed of the Sesame Framework and provides the basic infrastructure for the query processing engine. In particular this includes facilities for query parsing, Java mappings, I/O components, and the API for client interaction.

The federation layer is implemented as an extension to Sesame in form of a `SAIL` and constitutes FedX. FedX utilizes the basic Sesame infrastructure described above, and adds the necessary functionality for data source management, endpoint communication and - most importantly - optimizations for distributed query processing. Data sources can be added to a FedX federation in form of any repository mediator, where the latter means a supported Sesame repository implementation. Standard implementations are provided for local, native Sesame

[1] http://www.openrdf.org/doc/alibaba/2.0-beta4/
[2] FedX project page: http://iwb.fluidops.com/FedX

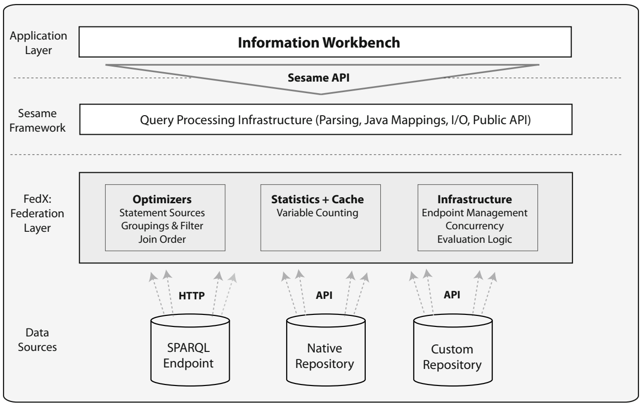


Fig. 1. FedX System Overview

repositories as well as for remote SPARQL endpoints. Furthermore custom mediators can be integrated by implementing the appropriate Sesame interface. With these mediators different types of federations are possible: purely local ones consisting of native, local Sesame repositories, endpoint federations or hybrid forms.

Federated query processing in FedX is comprised of the following steps. First, a global query is formulated against a federation of data sources. The global query is then parsed and optimized for the distributed setting. In particular it is split into local subqueries that can be answered by the individual data sources. Results of these local queries are merged in the federator and finally returned in an aggregated form. The whole process is transparent for the user, i.e. data appears to be *virtually integrated* in a single RDF graph.

Most crucial to the performance of such a query processing engine is the use of optimization techniques. Especially in the federated, distributed setting it is essential to apply new approaches to reduce the number of requests to the endpoints. Besides various generic techniques, FedX integrates some more sophisticated optimizations for the distributed environment. The combination of our applied join order optimization and the grouped subqueries reduce the number of intermediate results and requests tremendously, and are thus the major contributions for improving query performance in the distributed setting. The following listing gives an overview.

- **Statement sources:** Prior to query evaluation, all statements of the given SPARQL query are examined for their relevant data sources to avoid unnecessary communication during query processing.
- **Filter pushing:** SPARQL filter expressions are pushed down whenever possible to allow for early evaluation.
- **Parallel processing:** Concurrency is exploited by means of multithreaded execution of join and union computations.

- **Join order:** Join order tremendously influences performance since the number of intermediate results determines overall query runtime. In FedX the variable counting technique proposed in [3] supplemented with various heuristics is used to estimate the cost for each join. Following a greedy approach the joins are then executed in ascending order of cost.
- **Bound joins:** To reduce the number of requests and thus the overall runtime, joins are computed in a block nested loop join.
- **Groupings:** Statements which have the same relevant data source are co-executed in a single SPARQL query to push joins to the particular endpoint.

First benchmarks with FedBench[3] indicate a significant improvement of query performance compared to existing solutions[4]. For many queries proposed in [6] a performance gain of more than 90% can be achieved resulting in improvements of an order of magnitude, timeouts do not occur any longer. This is in particular due to the improved join order and the other above mentioned optimizations.

3 Demonstrating FedX in the Information Workbench

With the goal of illustrating the practicability of our system we provide a demonstration scenario using the previously discussed architecture. We employ the Information Workbench for demonstrating the federated approach to query processing with FedX. The Information Workbench is a flexible platform for Linked Data application development and provides among others frontend facilities for our UI as well as the integration with the backend, i.e. the query processing layers. In our demonstration we show a browser based UI allowing dynamic access and manipulation of federations at query time as well as ad hoc query formulation, then we execute the optimized query at the configured data sources using FedX, and finally we present the query results in the platform's widget based visualization components. The scenario steps from the user's point of view are summarized in the following and illustrated in figure 2.

1. **Linked Open Data discovery.** Data sources can be visually explored and discovered using a global data registry.
2. **Federation setup.** The federation is constructed and/or modified dynamically on demand using a browser based self-service interface. Discovered Linked Data repositories can be integrated into the federation with a single click.
3. **Query definition.** A query can be formulated ad hoc using SPARQL or selected from a subset of the FedBench queries. The predefined queries are designed to match the domain-specific data sources and produce results.

[3] FedBench project page: http://code.google.com/p/fbench/

[4] For an initial comparison we employed the AliBaba extension for the Sesame framework. To the best of our knowledge AliBaba provides the only federation layer available that does not require any proprietary extensions (e.g. SPARQL extensions).

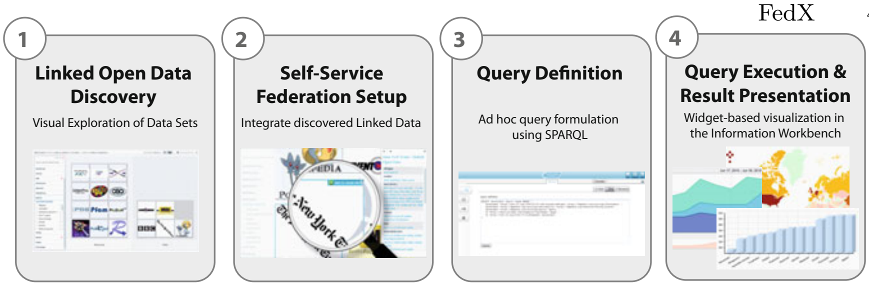

1
Linked Open Data Discovery
Visual Exploration of Data Sets
2
Self-Service Federation Setup
Integrate discovered Linked Data
3
Query Definition
Ad hoc query formulation using SPARQL
4
Query Execution & Result Presentation
Widget-based visualization in the Information Workbench

Fig. 2. Illustration of the Demonstration Workflow

4. **Query execution using FedX and result presentation.** The formulated query is submitted to the backend using the Sesame API and processed within FedX. After the query is parsed by Sesame, FedX applies its optimizations and evaluates the query at the data sources given by the dynamically configured federation. Finally, results are returned to the application layer for presentation in the widget based visualization components provided by the Information Workbench.

For the demonstration we use cross domain and lifescience datasets and queries as proposed in the FedBench benchmark. Those collections span a subset of the Linked Open Data cloud and are useful to illustrate practical applicability of query processing techniques such as those of FedX. Since FedX improves query response time compared to existing solutions, and moreover since the total runtime for most queries is in a range that is considered responsive, it is a valuable contribution for practical federated query processing.

4 Conclusion and Future Work

In this paper we have presented FedX and a practical demonstration within the Information Workbench. FedX provides a flexible federation layer integrated into the Sesame Framework which is suitable for practical application scenarios. First evaluation benchmarks have indicated that response time and query performance are such, that FedX in conjunction with a suitable application layer can be considered a highly valuable framework for federated query processing in today's settings. In future versions more advanced optimization techniques will be integrated into FedX to further improve query performance. Current ideas include the use of caching, integration of statistics (e.g. voiD [2]), and finally advanced grouping of subqueries to reduce the number requests.

References

1. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, p. 54. Springer, Heidelberg (2002)
2. Alexander, K., et al.: Describing Linked Datasets – On the Design and Usage of voiD. In: Proceedings of the Linked Data on the Web Workshop (2009)

3. Stocker, M., et al.: SPARQL basic graph pattern optimization using selectivity estimation. In: WWW, pp. 595–604. ACM, New York (2008)
4. Görlitz, O., et al.: Federated Data Management and Query Optimization for Linked Open Data. In: New Directions in Web Data Management (2011)
5. Erling, O., et al.: Rdf support in the virtuoso dbms. In: CSSW (2007)
6. Haase, P., Mathäß, T., Ziller, M.: An Evaluation of Approaches to Federated Query Processing over Linked Data. In: I-SEMANTICS (2010)
7. Neumann, T., Weikum, G.: Rdf-3X: a RISC-style engine for RDF. PVLDB 1(1) (2008)
8. Haase, P., et al.: The Information Workbench - Interacting with the Web of Data. Technical report, fluid Operations & AIFB Karlsruhe (2009)
9. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
10. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web. In: WWW (2008)

Reasoning in Expressive Extensions of the RDF Semantics

Michael Schneider

FZI Research Center for Information Technology, Karlsruhe, Germany
schneid@fzi.de

Abstract. The research proposed here deals with reasoning in expressive semantic extensions of the RDF Semantics specification, up to the level of OWL 2 Full. The work aims to conduct an in-depth study of the distinctive features and the degree of implementability of OWL Full reasoning. This paper describes the core problem, presents the proposed approach, reports on initial results, and lists planned future tasks.

Keywords: Semantic Web, Reasoning, RDF Semantics, OWL Full.

1 Problem Statement and State of the Art

This research deals with reasoning in expressive semantic extensions of the RDF Semantics specification [5]. The focus will specifically be on the ontology language OWL 2 Full [9], which has been standardized by the World Wide Web Consortium (W3C) in 2009 as an RDFS-compatible flavor of OWL that essentially covers all other members of the RDF and OWL language families. Several W3C languages have dependencies on OWL Full, including SKOS, RIF, and the current revision of SPARQL ("SPARQL 1.1"). So far, however, OWL Full has largely been ignored by the research community and no practically applicable reasoner has been implemented.

The current situation may have a variety of reasons. The most frequently heard technical argument against OWL Full reasoning is that OWL Full is *computationally undecidable* with regard to key reasoning tasks [7]. However, undecidability is a common theoretic problem in other fields as well, as for first-order logic reasoning, which still has many highly efficient implementations with relevant industrial applications [11]. Nevertheless, the undecidability argument and other arguments have led to strong reservations about OWL Full and have effectively prevented researchers from studying the distinctive features of the language and from searching for methods to realize at least useful *partial* implementations of OWL Full reasoning. But without a better understanding of OWL Full reasoning and its relationship to other reasoning approaches it will not even become clear what the added value of an OWL Full reasoner would be.

An OWL Full reasoner would make the full expressivity of OWL available to unrestricted RDF data on the Semantic Web. A conceivable use case for OWL Full reasoners is to complement RDF rule reasoners in reasoning-enabled applications operating on weakly-structured data to provide for significantly enhanced reasoning power. RDF rule reasoners, such as those implementing the OWL 2 RL/RDF rules [8], typically allow for efficient and scalable reasoning on arbitrary RDF data, but their reasoning capabilities are intrinsically limited, in particular with regard to terminological reasoning. OWL Full, on the other hand, offers very high reasoning expressivity, roughly comparable to that of OWL DL and even beyond. In a reasoning-enabled application, an RDF rule reasoner could rapidly produce the bulk of easily derivable results, while more thorough inferencing could be delegated to an OWL Full reasoner. Interoperability would often be warranted, since OWL Full applies to arbitrary RDF graphs as well, and is semantically fully compatible with RDFS and the OWL 2 RL/RDF rules. In contrast, OWL DL reasoners cannot generally be expected to work reliably in this use case due to the many syntactic restrictions and the differing semantics of OWL DL compared to the RDF Semantics.

As said, only little effort has been spent in the implementation of OWL Full reasoning so far. One general idea that can be found in the literature is to translate the native semantics of an ontology language into a first-order logic (FOL) axiomatisation and to use an automated theorem prover (ATP) for reasoning. An early application of this approach to a preliminary version of RDF and a precursor of OWL has been reported by Fikes et al. [1]. The focus of this work was, however, more on identifying technical problems in the original language specifications rather than on practical reasoning. Hayes [4] provides fairly complete translations of RDF(S) and OWL 1 Full into Common Logic, but does not report on any reasoning experiments. This gap is filled by Hawke's reasoner *Surnia* [3], which applies an ATP to an FOL axiomatisation of OWL 1 Full. For unknown reasons, however, Surnia performed rather poorly on reasoning tests [12]. Comparable studies have also been carried out for ATP-based OWL DL reasoning, which have shown more promising results [10].

This research is going to conduct an in-depth study of the features and the implementability of OWL Full reasoning. More precisely, the following research questions will be investigated: Firstly, *what are the distinctive features of OWL 2 Full compared to other approaches used for Semantic Web reasoning?* Secondly, *to which degree and how can OWL 2 Full reasoning be implemented?*

2 Proposed Approach and Methodology

The investigation of the two research questions will be carried out in the form of a *feature analysis* and an *implementability analysis.*

The goal of the **feature analysis** is to create a systematic and comprehensive *catalogue of distinctive pragmatic features* of OWL 2 Full. Both syntactic and semantic aspects of the language will be taken into account. A feature will be called *distinctive*, if it is not supported by either OWL 2 DL or by typical RDF rule reasoners, as those implementing the OWL 2 RL/RDF rules. For example, a distinctive *syntactic-aspect feature* might be the ability to assert a disjointness axiom between two annotation properties (as for SKOS lexical labels), while a distinctive *semantic-aspect feature* might be the ability to draw logical conclusions from such an axiom; neither is supported by OWL 2 DL. For each identified feature, a precise description, a motivation, an explanation for its distinctiveness, and one or more concrete examples will be given. Identification of the features may make use of any kind of source, including literature, ontologies, forum discussions, or technical aspects of the language. For each candidate feature, concrete evidence will be searched in order to support its validity.

The **implementability analysis** will concentrate on studying the *FOL translation approach*, as mentioned in Sec. 1. This approach has the advantage that it applies to arbitrary extensions of the RDF Semantics and it enjoys strong and mature reasoning tool support through existing ATPs. The idea is to create a corresponding FOL formula for every model-theoretic *semantic condition* of the OWL 2 Full semantics. For example, the semantic condition for class subsumption, as given in Sec. 5.8 of [9], can be translated into the FOL formula

$$\forall c, d : \mathrm{iext}(\mathrm{rdfs{:}subClassOf}, c, d) \Leftrightarrow \mathrm{ic}(c) \land \mathrm{ic}(d) \land \forall x : [\mathrm{icext}(c, x) \Rightarrow \mathrm{icext}(d, x)] \,.$$

An *RDF triple* '`:s :p :o`' is mapped to an atomic FOL formula '$\mathrm{iext}(p, s, o)$'. An *RDF graph* is translated into a conjunction of such 'iext' atoms, with existentially quantified variables representing blank nodes. An *entailment query* is represented by the conjunction of the OWL Full axiomatisation, the translation of the premise graph, and the negated translation of the conclusion graph. *Entailment checking* can then be performed by means of an ATP.

The implementability analysis will be based on a prototypical reasoner that is going to be built from an ATP, an FOL axiomatisation of the OWL 2 Full semantics, and a converter for translating RDF graphs into FOL formulas. The reasoner will then be evaluated based on the identified distinctive OWL Full features. This will be done by using the created concrete feature examples as test cases for conformance and performance testing of the parsing and the reasoning capabilities of the reasoner. Conversely, this method will help ensuring that the identified distinctive features will be technically valid. The evaluation results will be compared to those for OWL DL reasoners and RDF rule reasoners.

To the author's knowledge, the proposed research will be the first in-depth study of the features and the implementability of OWL 2 Full reasoning. No analysis of the distinctive pragmatic features of OWL 2 Full has been done so far. Also, there has been no rigorous analysis of OWL 2 Full reasoning based on the FOL translation approach yet.

3 Current Status and Initial Results

The **syntactic-aspect feature analysis** has been partially completed for the OWL 1 subset of OWL 2 Full. This work resulted in a catalogue of 90 features that have been grouped into 14 categories. The features were often motivated by data on the public Semantic Web. Example ontologies for all the features were used in the EU project *SEALS* (`http://seals-project.eu`) as test cases for the evaluation of ontology engineering tools. Project deliverable D10.3 reports on various problems that have been observed when using the OWL DL-centric OWL API (`http://owlapi.sourceforge.net`) with these test cases.

Table 1. Results for a test suite of 32 characteristic OWL Full conclusions

Reasoner / Mode	Success	Failure	Unknown	System Error
Pellet 2.2.2 / OWL-API 3.1	9	22	0	1
BigOWLIM 3.4 / owl2-rl	9	23	0	0
iProver 0.8 / all OWL Full axioms	28	0	4	0
iProver 0.8 / sufficient axioms only	32	0	0	0

For the **semantic-aspect feature analysis**, work on the development of several reasoning test suites has been started. The test suites are designed according to diverse criteria, such as language coverage, reasoning difficulty, and realism. One of these test suites consists of "characteristic" OWL 2 Full conclusions from e.g. meta-modeling, annotation properties, unrestricted use of complex properties, and "vocabulary reflection". They have been collected from forum discussions and other sources. The first two rows of Table 1 show the results of applying the test suite to the OWL 2 DL reasoner *Pellet* (`http://clarkparsia.com/pellet`) and the OWL 2 RL/RDF rule reasoner *OWLIM* (`http://ontotext.com/owlim`). Both reasoners succeeded on only a small fraction of the test suite, and they did so conjointly only on *two* of the test cases.

For the **implementability analysis**, an FOL axiomatisation has been created for a major fragment of the OWL 2 Full semantics, the main omission being support for datatype reasoning. The used FOL dialect is the language for encoding *TPTP* problems (`http://tptp.org`). A converter from RDF graphs to TPTP formulas has also been implemented. This enables the use of the OWL Full axiomatisation with a large number of existing ATPs.

The third row of Table 1 shows the results from applying the same test suite to the ATP *iProver* (`http://code.google.com/p/iprover`). When using the complete OWL Full axiomatisation, iProver succeeded on most of the test cases, but for a few test cases it did not terminate within a time limit of 300 seconds. Further analysis suggested that this issue may be due to the large number of complex OWL 2 Full axioms. The complete axiomatisation was then, separately for each test case, manually reduced to small sub-axiomatisations that were just sufficient to entail the expected result. This allowed iProver to succeed on *all* test cases (fourth row). In this scenario, reasoning typically finished within a few hundredth of a second on a standard personal computer, compared to often several seconds for the complete axiomatisation. These and additional results from experiments concerning OWL 2 Full language coverage, scalability and model-finding have been submitted to CADE 2011.

4 Conclusions and Future Work

This paper proposed a first in-depth study of the distinctive features and the degree of implementability of OWL 2 Full reasoning. First results indicate that one can use ATPs and an FOL axiomatisation of the OWL 2 Full semantics for genuine OWL Full reasoning beyond the capabilities of typical RDF rule and OWL DL reasoners. The approach is flexible enough to implement arbitrary extensions of the RDF Semantics or to add features such as rule-style reasoning.

It has been observed that acceptable reasoning performance can often only be achieved by reducing the whole OWL Full axiomatisation to a small sufficient sub-axiomatisation. A next step will therefore be to search for an automated method to *eliminate redundant axioms* with regard to the given input ontology.

So far, the implementability analysis was restricted to entailment and inconsistency checking. To fulfil the discussed use case of complementing RDF rule reasoners in reasoning-enabled applications, OWL Full reasoners should also support flexible *query answering* on arbitrary RDF data. This will specifically be needed to realize the *OWL 2 RDF-Based Semantics entailment regime* of SPARQL 1.1 [2]. Some ATPs have been reported to offer query answering on FOL knowledgebases [6]. It will be analyzed to what extent these capabilities can be exploited for OWL Full query answering.

For the *syntactic-aspect feature analysis*, which has already been finished for OWL 1 Full, the remaining work will be to extend the analysis to the whole of OWL 2 Full. The *semantic-aspect feature analysis* is in a less-complete state and still requires the development of a feature categorization similar to that of the syntactic-aspect feature analysis. The started work of building reasoning test suites will be continued and will eventually lead to a collection of concrete examples for the still to-be-identified semantic-aspect features.

References

1. Fikes, R., McGuinness, D., Waldinger, R.: A First-Order Logic Semantics for Semantic Web Markup Languages. Tech. Rep. KSL-02-01, Knowledge Systems Laboratory, Stanford University, Stanford, CA 94305 (January 2002)
2. Glimm, B., Ogbuji, C. (eds.): SPARQL 1.1 Entailment Regimes. W3C Working Draft (October 14, 2010)
3. Hawke, S.: Surnia (2003), `http://www.w3.org/2003/08/surnia`
4. Hayes, P.: Translating Semantic Web Languages into Common Logic (July 18, 2005), `http://www.ihmc.us/users/phayes/CL/SW2SCL.html`
5. Hayes, P. (ed.): RDF Semantics. W3C Recommendation (February 10, 2004)
6. Horrocks, I., Voronkov, A.: Reasoning Support for Expressive Ontology Languages Using a Theorem Prover. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 201–218. Springer, Heidelberg (2006)
7. Motik, B.: On the Properties of Metamodeling in OWL. Journal of Logic and Computation 17(4), 617–637 (2007)
8. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (October 27, 2009)
9. Schneider, M. (ed.): OWL 2 Web Ontology Language: RDF-Based Semantics. W3C Recommendation (October 27, 2009)
10. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using Vampire to Reason with OWL. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 471–485. Springer, Heidelberg (2004)
11. Voronkov, A.: Automated Reasoning: Past Story and New Trends. In: Proc. IJCAI 2003, pp. 1607–1612 (2003)
12. W3C WebOnt OWL Working Group: OWL 1 Test Results (March 9, 2004), `http://www.w3.org/2003/08/owl-systems/test-results-out`

Personal Semantics: Personal Information Management in the Web with Semantic Technologies

Salman Elahi

Knowledge Media Institute (KMi), The Open University, UK
s.elahi@open.ac.uk

Abstract. Every web user has several online profiles through which personal information is exchanged with many service providers. This exchange of personal information happens at a pace difficult to fully comprehend and manage without a global view and control with obvious consequences on data control, ownership and, of course, privacy. To tackle issues associated with current service-centric approaches, we propose a user-centric architecture where the interaction between a user and other agents is managed based on a global profile for the user, maintained in a profile management system and controlled by the user herself. In this PhD, we will investigate research issues and challenges in realizing such a system based on semantic technologies.

1 Research Problem

Web users maintain several online profiles across e-commerce websites, social networks and others, making it difficult for them to realize how much information is exchanged and what happens to that information. In other words, web interactions are happening in a 'one to many' mode, where different agents, with different statuses and relationships to the user take part and receive personal information. We refer to this phenomenon as the *fragmentation of personal data exchange*, where many different 'destinations' of the data receive various fragments of personal information over time, without the user having a global view of her own data.

The problem stems from the model these online interactions are based upon, i.e. a service-centric model, where everything is focused on the needs of a particular organization. Current research suggests a rather contrasting model to address these issues: a user-centric approach [1] where the interaction between a user and other agents is managed based on a global profile for the user, maintained in a profile management system and controlled by the user herself. Parts of this profile can be requested by various agents (websites), with the user being given the possibility to control these accesses and to keep track of them directly within the profile management system. In this PhD, we will investigate research issues and challenges in realizing a system based on user-centric profiles, and show how semantic technologies can support users in making sense, managing and controlling their exchanges of personal data on the Web with such a system.

2 State of the Art

Personal information management as defined in [2] is at the core of Semantic Desktops [3], which are concerned with the management of information produced and consumed by desktop applications. While there are significant overlaps, our approach focuses on personal information on the web and on the way this information is exchanged with other agents, rather than on the management of information in the close, local environment of the user.

Identity management shares a thin boundary with the aspects of personal information management we are looking at. The most popular approach for "user-centric" identity management is OpenID[1]. OpenID is a protocol that provides a unique ID to each user to be used over various services. It therefore supports users in reducing the distribution of their identity data. OAuth[2] coupled with OpenID provides secure exchange of data without disclosing users' credentials to third party websites. However, OpenID and OAuth are essentially concerned with the problem of authentication, not the management of personal information. A few initiatives are attempting to realize user-centric identity management frameworks, including Windows CardSpace[3], LibertyAlliance[4], Higgins I-Card[5], etc. These frameworks provide central places for storing and managing personal information (i.e., profiles), to which external websites are granted access. In this sense, they comply with our notion of user-centric personal information management system. While they still suffer from a number of limitations (e.g., in iCard based frameworks, personal data is still fragmented, in the sense that profile information is "boxed" according to the websites requesting it). [4,5] concern flexible user profiles in mash up environments. They focus on domain independent models while [6,7] also discuss user models but with a focus on retail domain. FOAF+SSL [8] is an authentication protocol to provide secure exchange of distributed information. It uses the SSL layer in modern day Web browsers to create a secure conduit of information between two parties. FOAF+SSL is also in an initial phase of development and has not been widely adopted yet.

3 Propose Approach and Methodology

A social context provides a very comfortable leverage to someone to represent and manage their persona in different real world interactions as compared to online interactions, where it becomes a tedious task to maintain one's identity (personal information) in one place because of the many different profiles a typical web user usually possesses with each service provider she interacts with. In this scenario, user-centric profiles seem to be a promising solution where a user can define different contexts to disclose certain aspects of her profile depending on what type of agents

[1] http://www.openid.net
[2] http://www.oauth.net
[3] http://www.microsoft.com/windows/products/winfamily/cardspace/default.aspx
[4] http://www.projectliberty.org/liberty/about/
[5] http://www.eclipse.org/higgins/

she is communicating with. However, the realization of a truly user-centric personal profile management system raises complex research issues and challenges, which we intend to tackle in this PhD, partly based on the use of semantic technologies. More precisely, we will focus on the technical and the user aspects [9]:

- *The technical aspects* include in particular issues related to user modelling in an open, web environment: how to integrate multiple profiles into one global profile? How to manage access policies and access rights on user information, including identifying policy conflicts between consumer and provider and suggesting ways to resolve them? How to make a user's interaction more sophisticated and friendly with new concepts? How to represent behavioural changes in the profile while maintaining consistency?
- *The user-related issues* regard the way to integrate this user-centric approach in the existing social environment of the user: How much of a profile can be constructed with already available user information? What interaction models are suitable to the management of large, global user profiles by web users? Will users be willing to invest time in such a system? What mechanisms are necessary to ensure trust in the profile providers?

In practice, we envisage the combination of user-centric profile management with semantic technologies as a system where the user is offered the ability to manage one, integrated profile containing information potentially relevant to different agents she is communicating with: a semantic model of her personal information. Such a semantic model of personal information can not only be used to negotiate terms of communications (access policies) with different service providers but can also be reasoned upon, enabling a user to have a global coherent model of her data, helping her in making sense, managing and giving access to them. Through this approach, we intend to answer the following three questions:

1. How to model a user? Different users would have different requirements to be able to flexibly structure, organize and represent their arbitrary data.
2. How to model semantic access control over semantic data? We will be looking at how access control, and especially definition of access rights in user-centric profile scenarios is realized in a way homogeneous to the description of its semantic data, taking benefit from the flexible data model and inference capabilities provided by the Semantic Web technologies.
3. How users would react to such a new approach? We will empirically study the impact of a user-centric profile management system on the practices of data exchange, based on user surveys and on monitoring usages.

The methodology to resolve these questions will consist of the following phases:

1. **Feasibility studies:** The idea of user-centric profile management has not yet been tested in the open environment of the web, where thousands of exchanges happen with hundreds of websites every day. In the next section, we give an overview of a first study where we reconstructed a global user profile from logs of a particular user's activities on the web [10]. Such feasibility studies help us

identifying specific technical issues related to the combination of semantic technologies with user-centric profile management in a web environment.

2. **Contrasting the user-centric approach with existing models:** In this phase, we focus on the aspect of managing the flexible exchange and access to personal information using an ontological model. We are conducting another experiment by creating a semantic representation of the current information and access model of a large organization (The Open University), in order to contrast it with a possible user-centric view of information and access. The Open University represents a perfect example of a large-scale distributed organization with multiple departments, a very large number of distance learning students and off campus employees. In this phase, we intend to demonstrate how a user-centric, semantic model of 'personal information' and 'access policies' provides added value through allowing users to manage, reason upon and control their own information in a more transparent way than in an organization centric model.
3. **Tackling specific technical issues and prototyping:** We propose an architecture consisting of four evolving sub-frameworks. The profile framework will provide intuitive interfaces for distributed management of user-centric profiles. The policy framework acts as a security grid based on the intelligence provided by the Semantic framework, which assesses user behaviour and provide recommendations to evolve the profile and keeps the cycle going. The policy framework derives security policies based on the rich semantic information about the agents the user is interacting with and defines access rights on the contents which are then used in the Content framework when a request comes in for certain information from a consuming agent [9].
4. **Empirical study and evaluation:** Along with the evaluation studies of phase 1 and 2 we will be realise empirical studies where the prototype system developed in the previous phase will be used within an organization instead of the usual information and access mechanisms. Through collecting both usage and survey data, we will be able to assess the benefit of the approach, and evaluate specific advantages gained through the use of semantic technologies.

4 Initial Results

In this section we discuss initial results obtained (during the first year of this part-time PhD). In [10], our objective was to collect all the fragments of personal information sent by a user during several weeks, and to try to reconstruct from these fragments a coherent global profile. From logs of HTTP traffics through specifically developed tools, the user managed to create 36 profile attributes mapped onto 1,108 data attributes. However, while this small and simple exercise proved to be satisfying the basic hypothesis, we also identified few research issues and challenges which will need to be addressed to create more complex and flexible user profiles [10], including the need for complex and sophisticated profile representations (with temporal and multi-faceted representations), the need to include multiple, external sources of information to enrich the profile (including information about the accessing agents) and the need for appropriate mechanisms for the definition of semantic access control model over semantic data. On the last point, we are currently investigating a prospective model for access control in a user-centric scenario, and applying it in the scenario of the Open University's information and access model. Through the use of an ontological model, and the possibility of employing inference upon such a model, we expect to obtain results showing how the user-centric approach can outsmart the organization-centric approach, providing better overviews of the access control aspects over the considered data and possibly detecting un-intended behaviours which normally remain hidden in an organization-centric view.

5 Future Work

The next major steps for this part-time PhD include completing the current work on Semantic policies and access rights in the next 4 months. The results obtained from this exercise will be used to develop a prototype profile management system based on the user-centric framework discussed earlier. This framework will employ semantic technologies to provide access control through semantic policies with evolvable profiles representing changing needs of a user in today's online interactions. This phase will investigate and try to address the technical issues mentioned above. Social issues will be investigated in the next phase with the deployment of this system in place of the Open University's usual information and access mechanisms to gather empirical evidence of how users interact with such a system and its impact on them. This empirical evidence will be used in the evaluation of the system.

References

1. Iannella, R.: Social Web Profiles. Position paper, SNI (2009)
2. Jones, W., Teevan, J.: Personal Information Management (2007); ISBN: 9780295987378
3. Sauermann, L., Bernardi, A., Dengel, A.: Overview and outlook on the semantic desktop. In: ISWC (2005)
4. Leonardi, E., Houben, G., Sluijs, K., Hidders, J., Herder, E., Abel, F., Krause, D., Heckmann, D.: User Profile Elicitation and Conversion in a Mashup Environment. In: FIWLIW (2009)
5. Abel, F., Heckmann, D., Herder, E., Hidders, J., Houben, G., Krause, D., Leonardi, E., Slujis, K.: A Framework for Flexible User Profile Mashups, AP-WEB 2.0 (2009)
6. Ghosh, R., Dekhil, M.: Mashups for semantic user profiles. Poster. In: International World Wide Web Conference, WWW (2008), Poster
7. Ghosh, R., Dekhil, M.: I, Me and My Phone: Identity and Personalization Using Mobile Devices. In: HPL 2007-184 (2007)
8. Story, H., Harbulot, B., Jacobi, I., Jones, M.: FOAF+SSL: RESTful Authentication for the Social Web. In: SPOT (2009)
9. Olesen, H., Noll, J., Hoffmann, M. (eds.): User profiles, personalization and privacy. WWRF Outlook series, Wireless World Research Forum (May 2009)
10. Elahi, S., d'Aquin, M., Motta, E.: Who Wants a Piece of Me? Reconstructing a User Profile from Personal Web Activity Logs. In: LUPAS, ESWC (2010)

Reasoning with Noisy Semantic Data

Qiu Ji[1], Zhiqiang Gao[1,⋆], and Zhisheng Huang[2]

[1] School of Computer Science and Engineering, Southeast University, Nanjing, China
{jiqiu,zqgao}@seu.edu.cn

[2] Department of Mathematics and Computer Science, Vrije University Amsterdam
huang@cs.vu.nl

1 Problem Statement

Based on URIs, HTTP and RDF, the Linked Data project [3] aims to expose, share and connect related data from diverse sources on the Semantic Web. Linked Open Data (LOD) is a community effort to apply the Linked Data principles to data published under open licenses. With this effort, a large number of LOD datasets have been gathered in the LOD cloud, such as DBpedia, Freebase and FOAF profiles. These datasets are connected by links such as owl:sameAs. LOD has gained rapidly progressed and is still growing constantly. Until May 2009, there are 4.7 billion RDF triples and around 142 million RDF links [3]. After that, the total has been increased to 16 billion triples in March 2010 and another 14 billion triples have been published by the AIFB according to [17].

With the ever growing LOD datasets, one problem naturally arises, that is, the generation of the data may introduce noise, thus hinders the application of the data in practice. To make the Linked Data more useful, it is important to propose approaches for dealing with noise within the data. In [6], the authors classify noise in Linked Data into three main categories: accessibility[1] and derefencability[2] w.r.t. URI/HTTP, syntax errors, and noise[3] and inconsistency[4] w.r.t. reasoning. In our work, we focus on dealing with the third category of noise, namely noise and inconsistency w.r.t. reasoning, but may also consider other categories of noise. We further consider one more noise in the logical level, that is, the logical inconsistency caused by ontology mapping.

2 State of the Art

In [6], a comprehensive study of various kinds of noise in Linked Data have been conducted over 149,057 URIs concluding 54,836 valid RDF/XML document. In

⋆ Corresponding author.

[1] The problem of accessibility here means some of the retrieved documents have no structured data or contain misreported content-types.

[2] Dereferencing means providing information about a resource lookup of its URI using HTTP.

[3] The noise can be atypical use of OWL/RDFS vocabularies, or use of undefined classes and properties, and so on.

[4] Inconsistency here means logical inconsistency in OWL ontologies.

[1], an approach was proposed to detect accidental syntactic errors or vocabulary misuse and then apply patches to produce OWL DL ontologies. As owl:sameAs has been heavily used in LOD to connect different data resources, it becomes more and more important to use it correctly. That is, any two URI references connected by owl:sameAs should be the same thing. But in reality, the correctness can not be ensured. Therefore, the authors in [5] explored the origins of this situation and developed an Similarity Ontology by systematizing various theoretically-motivated distinctions which are 'kind of close' to owl:sameAs.

Compared with other kinds of noise in Linked Data, there have been much work on dealing with logical contradictions in OWL ontologies (see [2] for a survey). Given an inconsistent ontology, one can either use an inconsistency-tolerant approach to reasoning with it (e.g. [7]) or repair it (e.g. [16]). To provide some additional information to deal with inconsistency, some researchers have proposed to measure inconsistency in an OWL ontology [10]. Logical inconsistency can also occur when mappings among ontologies are established [11].

In our work, we will mainly focus on noise w.r.t. reasoning in Linked Data. We will enhance the state of the art in the following aspects. First, we will consider learning expressive ontologies from noisy LOD datasets and provide methods to measure the noise. Second, we will enhance existing approaches to handle the inconsistency in learned ontologies by using some patterns and defining novel inconsistency-tolerant approaches. Third, we will propose measures to evaluate inconsistent mappings and novel methods to repair mappings. We explain each of the three aspects in detail in the following section.

3 Proposed Approach and Methodology

3.1 Statistical Reasoning with Noisy Linked Data

Developing an ontology is not an easy task and often introduces noise and incompleteness. This happens in LOD datasets as well. The ontologies in LOD datasets are generally inexpressive and may contain a lot of noise. For example, one of the most popular ontology, DBpedia ontology[5], is claimed as a shallow ontology. The TBox of this ontology mainly includes a class hierarchy.

To deal with the incomplete ontologies, we plan to use statistical relational learning(SRL) techniques to learn expressive ontologies from LOD datasets (more details can be seen in [19]).

Before learning ontologies, we will propose methods to measure the noise of a data, which can be defined according to the data quality assessment [13] in database area. For instance, an objective metric can be defined as the degree to which misused vocabularies from all vocabularies in an ontology. Such kind of measures provides a reference to decide whether a dataset needs to be cleaned or not. If it is necessary to do cleaning, we could apply various cleaning strategies to correct or remove the noise. For example, we can correct the misused

[5] http://wiki.dbpedia.org/Ontology

vocabularies manually with the help of an ontology editor like Protege[6]. After ontology learning, we can define the measure of ontology incompleteness which is the degree to which axioms are missing from the ontology.

3.2 Handling OWL Inconsistency in Linked Data

After learning expressive ontologies from LOD datasets and linking them with other datasets, we may confront the problem of inconsistency[7] handling. According to [6], it may be quite difficult to deal with this kind of noise. Due to the large scale of the data, it is hard to apply existing approaches for reasoning with inconsistent OWL ontologies to deal with OWL inconsistency in Linked Data.

In [6], the authors suggested that, to handle inconsistency in Linked Data, inconsistent data may be pre-processed with those triples causing inconsistencies dropped according to some heuristic measures. We fully agree with this proposal. One measure that we will consider is the inconsistency measure defined by four-valued semantics (see [10]). In the open philosophy of the Web, it may be not desirable to completely repair inconsistent ontologies. One reason, as suggested in [6], is that contradiction could be considered as a 'healthy' symptom of different opinion. Therefore, when we repair inconsistency in OWL ontologies in Linked Data, our goal is not to result in fully consistent ontologies, but to reduce the inconsistency degrees of those ontologies. After that, we can apply some inconsistency-tolerant approaches to reasoning with those inconsistent ontologies. To partially repair an inconsistent ontology, we will apply some patterns to efficiently detect the sources of inconsistency, such as patterns given in [18]. To provide inconsistency-tolerant reasoning services, we will further develop the idea of using selection functions [7] to reasoning with inconsistent ontologies. The idea is to propose specific selection functions for specific ontology languages.

3.3 Mapping Repair and Evaluation

As reported in [8], LOD datasets are well connected by RDF links on the instance level. But on the schema level, the ontologies are loosely linked. It is interesting to consider aligning these ontologies based on the plentiful resources of LOD datasets. A few such approaches have been proposed in [8,12].

With the mappings generated, we may confront the problem of dealing with inconsistency caused by mappings and ontologies if we interpret mappings with OWL semantics. We will first consider evaluating the inconsistent mappings[8] by defining a nonstandard reasoner. We will then consider mapping repair based on work in [14,11]. For example, we can apply some patterns to efficiently detect problematic correspondences in the mappings.

[6] `http://protege.stanford.edu/`

[7] A data is inconsistent iff it has no model.

[8] An inconsistent mapping means no concepts in $O_1 \cup O_2$ are interpreted as empty but there is such a concept in the union of O_1, O_2 connected by $\mathcal{M}$.

3.4 Evaluation

To evaluate our work, we will implement our proposed approaches and do evaluation over LOD datasets. Based on our previously developed system RaDON[9], which is a tool to repair and diagnose ontology networks, we will develop a system for reasoning with noisy Linked Data.

4 Results

We have studied repair and diagnosis in ontology networks and developed a tool, called RaDON, to deal with logical contradictions in ontology networks (see [9]). The functionalities provided by RaDON have been implemented by extending the capabilities of existing reasoners. Specifically, the functionalities include debugging and repairing an inconsistent ontology or mapping, and coping with inconsistency based on a paraconsistency-based algorithm.

In [15], we proposed possibilistic extension of OWL to deal with inconsistency and uncertainty in OWL ontologies. Some novel inference services have been defined and algorithms for implementing these inference services were given. We have implemented these algorithms and provided evaluations for their efficiency.

For an inconsistent mapping, the semantic precision and recall defined in [4] meet the trivialization problems. To resolve such kind of problems, we define the meaningfulness of an answer given by an inconsistency reasoner: Given two ontologies O_1 and O_2 and a mapping $\mathcal{M}$ between them, for a correspondence $c = \langle e, e', r, \alpha \rangle$, an answer provided by an inconsistency reasoner is meaningful iff the following condition holds: $\Sigma \Vdash t(c) \Rightarrow (\exists \Sigma' \sqsubseteq \Sigma)(\Sigma' \not\models e \sqsubseteq \bot \;\; and \; \Sigma' \not\models e' \sqsubseteq \bot \;\; and \; \Sigma' \models t(c))$. Here, e and e' are atomic concepts. r is a semantic relation like equivalence and α is a confidence value. t is a translation function to transfer a correspondence to a DL axiom. Σ is the union of O_1, O_2 and a set of axioms obtained by translating all correspondences in $\mathcal{M}$ to DL axioms. An inconsistency reasoner is regarded as meaningful iff all of the answers are meaningful. Based on this definition, we can redefine semantic measures in [4].

5 Conclusions

Reasoning with noisy Linked Data is a quite challenging and interesting work. In our work, we mainly consider the following work: (1) We will propose methods for measuring noisy LOD datasets like incompleteness and clean the noise in these datasets if necessary. Based on the plentiful LOD datasets, we will propose methods for learning expressive ontologies using SRL techniques. (2) To deal with logical inconsistency, we propose to partially repair an inconsistent ontology by considering some patterns to achieve good scalability for LOD datasets. Then we plan to apply some novel inconsistency-tolerant reasoning strategies like defining specific selection functions for specific ontology languages. (3) We will propose methods for evaluating inconsistent mappings and methods to repair inconsistent mappings by applying some patterns.

[9] `http://neon-toolkit.org/wiki/RaDON`

Acknowledgements

This paper is sponsored by NSCF 60873153 and 60803061.

References

1. 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)
2. Bell, D., Qi, G., Liu, W.: Approaches to inconsistency handling in description-logic based ontologies. In: Chung, S., Herrero, P. (eds.) OTM-WS 2007, Part II. LNCS, vol. 4806, pp. 1303–1311. Springer, Heidelberg (2007)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. In: IJSWIS, pp. 1–22 (2009)
4. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: IJCAI, Hyderabad, India, pp. 348–353 (2007)
5. 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)
6. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: LDOW, Raleigh, NC, USA (2010)
7. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: IJCAI, pp. 454–459. Morgan Kaufmann, San Francisco (2005)
8. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology alignment for linked open data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 402–417. Springer, Heidelberg (2010)
9. Ji, Q., Haase, P., Qi, G., Hitzler, P., Stadtmüller, S.: RaDON — repair and diagnosis in ontology networks. 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. 863–867. Springer, Heidelberg (2009)
10. Ma, Y., Qi, G., Hitzler, P.: Computing inconsistency measure based on paraconsistent semantics. Journal of Logic and Computation (2010)
11. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing ontology mappings. In: AAAI, pp. 1408–1413 (2007)
12. 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)
13. Pipino, L., Lee, Y.W., Wang, R.Y.: Data quality assessment. ACM Commun. 45(4), 211–218 (2002)
14. Qi, G., Ji, Q., Haase, P.: A conflict-based operator for mapping revision. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 521–536. Springer, Heidelberg (2009)
15. Qi, G., Ji, Q., Pan, J.Z., Du, J.: Extending description logics with uncertainty reasoning in possibilistic logic. International Journal of Intelligent System (to appear, 2011)

16. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. J. Autom. Reasoning 39(3), 317–349 (2007)
17. Vrandecic, D., Krotzsch, M., Rudolph, S., Losch, U.: Leveraging non-lexical knowledge for the linked open data web. Review of AF Transactions, 18–27 (2010)
18. Wang, H., Horridge, M., Rector, A.L., Drummond, N., Seidenberg, J.: Debugging OWL-DL ontologies: A heuristic approach. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 745–757. Springer, Heidelberg (2005)
19. Zhu, M., Gao, Z.: SRL based ontology learning from linked open data. In: ESWC PhD Symposium,Crete, Greece (to appear, 2011)

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

Roxane Segers

Department of Computer Science, VU University Amsterdam
r.h.segers@vu.nl

1 Research Context and Problem Statement

Currently, cultural heritage portals limit their users to search only for individual objects and not for objects related to some historical narrative. Typically, most museums select objects for an exhibition based on the story they want to tell the public, but in digital collections this context can currently not be made explicit as the historical context is not part of the object annotations.

From previous experiences with cultural heritage portals such as Europeana[1] and CHIP[2], we observed that adding historical events to object descriptions is valuable for the grounding of cultural heritage objects in their historical context as events represent important changepoints in time and form the basic units of the historical narrative. Further, historical event descriptions are comprised of actors, locations and timestamps that are in some cases already present as facets of the object annotation. As such, adding events to object descriptions can enhance browsing and searching of cultural heritage collections as the events unify otherwise unrelated but historical relevant facets of object annotations.

The **problem** motivating this research is threefold: (1) There is no standard practice in cultural heritage organizations to include events within the object annotation, e.g. there is neither a shared vocabulary for the (historical) event descriptions, nor for different historical terms and concepts. The creation of such vocabulary is important in order to ensure alignment between the descriptions of historical events, which typically vary over time. (2) A multitude of different historical perspectives and interpretations of the same historical event exist. As a result, a variety of expressions can be used to describe the same event which implies problems for creating thesauri and vocabularies. (3) There is no consensus on the elements that make up an event and which could provide meaningful relationships between the events and pertaining objects.

This PhD research is situated in the **context** of two projects: (1) The *Semantics of History* project[3] with focus on the modeling and extraction of events and their perspectives from historical text documents; (2) *Agora*[4] with focus on

[1] http://www.europeana.eu
[2] http://www.chip-project.org/
[3] http://www2.let.vu.nl/oz/cltl/semhis/
[4] http://www.agora.cs.vu.nl

searching, browsing and representing historical events online. Both projects use the digital collections of the Netherlands Institute for Sound and Vision[5] and the Rijksmuseum Amsterdam[6].

In my research, I focus on the following **research questions**:

1. What is an adequate event model to capture variances and invariances of historical events? Can we instantiate this event model (semi-)automatically?
2. What is an adequate organization of a historical ontology to be used for annotations for cultural heritage objects?
3. What is a suitable organization of a historical event thesaurus to allow for diverse interpretations and expressions for events?
4. What are relevant evaluation criteria for quality of the event model, the historical ontology and the thesaurus and their added value for exploration and search of cultural heritage objects?

Issues Related to the Research Questions

1. **The event model** provides the vocabulary for the event elements and their relations. However, the domain-specific requirements for modeling historical events in terms of *classes and properties* cannot be given beforehand. Additionally, the historical event model facilitates *relations between events*, e.g. causality, meronymy. However, these relations are not part of an event but exist as an interpretational element between two or more events and thus need to be modeled as a separate module.
2. **The historical ontology** serves as a semantic meta-layer to type historical events, independently of the expressions used. However, it is unknown to what degree an ontology can be used as an *unprejudiced meta-layer* for such typing as it might imply an interpretation. Ontologies typically represent a *time-fixed view on reality* which influences the modeling of objects that can only play a role in an event after a certain point in time. Additionally, the *expressivity and extensibility* of the ontology depends on the expressivity and extensibility of the event model and vice versa. It is critical to know how they interact, as incompatible properties can affect reasoning about events.
3. **The instantiation of the event model** needs to be based on different sources to capture the different perspectives and interpretations of events. Typically, event descriptions reside in unstructured text documents. Thus, portable information extraction techniques should be applied for detecting events and their elements in document collections of different style and topic.
4. **The event thesaurus** is a structured set of historical events used for event-based annotation of cultural heritage objects and for aligning different object collections. For the creation of such a thesaurus we need to know (1) how to identify and organize equal and similar event descriptions and (2) how to identify and structure multiple interpretations of the relations between events. Properties such as `hasSubevent` can become problematic for structuring the thesaurus, as some sources might only report temporal inclusion.

[5] http://portal.beeldengeluid.nl/

[6] http://www.rijksmuseum.nl/

2 State of the Art

This PhD work is related to four research areas. Here, we give a brief state of the art, a comprehensive overview of the related work can be found online[7].

Event models: Various event models exist, e.g. the Event Ontology[8], LODE [13], the F-Model [12], SEM [6] and CIDOC-CRM[9]. However, none were explicitly designed for historical events and each has various limitations concerning extending the model with domain-specific properties.

Model instantiation: Diverse information extraction (IE) techniques are used to instantiate models in a variety of domains, e.g. [1] and [4]. However, historical event extraction is emerging only recently.

Ontologies: Formal ontologies, e.g. DOLCE [10] and SUMO [11], and lexical databases, e.g. WordNet [5] exist that can partly be reused for historical ontology. Top-level ontologies pose modeling choices that may not compatible with the historical domain requirements. WordNet is language specific and not consistent in the modeling of synsets, which hampers the ontological soundness for an historical ontology.

Ontology learning can be seen as a subtask of information extraction that focuses on learning classes and relations between classes. Different techniques exist for ontology learning, e.g. [9], [2] but interpretational issues pertaining to historical events have not been addressed yet.

Related projects: A historical thesaurus [7] has been used in CultureSampo[10] to enhance searching and browsing of Finnish cultural heritage. It comprises event instances statically organized in a timeline and does not allow for various views on events. Modeling historical data and events has also been the focus of FDR/Pearl Harbor project [8] but no results have been published yet.

3 Approach

We propose the following novel approach for extracting and structuring knowledge of historical events from various text sources. First, we adapt an existing event model to meet the domain specific requirements. Next, we populate this model and learn a historical ontology using information extraction techniques.[11] For the creation of the event thesaurus we consider to use different reasoning techniques over both the instances and types of the modeled event descriptions. Following, we elaborate on the approach in relation to the research questions:

RQ1: We consider SEM[6] as a model to start from, as it is not domain-specific, represents a minimal set of event classes and includes placeholders for a foreign typing system.

[7] http://semanticweb.cs.vu.nl/agora/relatedwork
[8] http://motools.sf.net/event/event.html
[9] http://cidoc.ics.forth.gr/officialreleasecidoc.html
[10] http://www.kulttuurisampo.fi/
[11] see: http://semanticweb.cs.vu.nl/agora/experiments

RQ 2: We consider learning the ontology bottom up by using the facets of the extracted events as relevant terms in the domain[2]. WordNet is used as an external vocabulary to semantically organize the terms and determine the least common subsumer[3]. We consider to map the ontology to DOLCE to guarantee ontological soundness.

RQ 3: We consider to learn lexical patterns for extracting coarse-grained historical event descriptions from general Web documents and apply these to domain-specific text collections. These patterns are semantically rich and can be used to classify the extractions. The relevance scores for the patterns are used to determine the precision of the extractions. To boost the recall of the pattern-based extraction, we consider using the internal syntactic structure of the events as patterns.

RQ4: For the creation of the thesaurus, we consider temporal-spational reasoning methods to identify similar event descriptions. To identify explicit relations between events, we consider information extraction in the text documents. Further, implicit relations are inferred from the typing of the events.

4 Methodology

We apply the following iteration methodology in order to realize the approach in section 3, i.e. **Iteration I** is scoped on acquisition of basic models:

– Analysis of SEM classes for the information extraction process.
– Learn *patterns* to instantiate SEM classes, starting with the *event class*. Next, we extend to other classes and pertaining relations. We combine the results of three IE techniques: (1) pattern-based and (2) co-occurrancy based, both using Yahoo and Wikipedia and (3) lexical framing in newspaper collections. For each we evaluate the recall, precision and reusability.
– *Ontology*, version 1, based on the first extraction results.
– *Thesaurus*, version 1, with limited relations.
– Test and evaluate the *ontology* and *thesaurus* in the Agora demonstrator. We define new requirements from the evaluation.

In **Iteration II** we iterate all the RQs once again to extend the models with domain specific requirements:

– Extend the document collection to domain-specific texts, e.g. scopenotes with links to historical themes and historical handbooks. We scope the domain to two periods/themes of interest to the involved cultural heritage institutions. Apply the IE techniques and the extended event model. Creation of ontology version 2 with unprejudiced typing of events.
– Evaluate the ontology and thesaurus version 2 by applying the IE module and event model to another historical period/theme to ensure that the results are not over-fitting the data. Integrate the results in the Agora demonstrator.
– Define requirements for evaluating the thesaurus in the Agora demonstrator, e.g. added value in terms of links between objects (quantitative), added value in terms of relevant and coherent links (qualitative).

5 Achieved Results and Future Work

The PhD work is now entering the second year. Current work involves analysing the extracted events by the pattern-based IE. The results so far are:

- literature study on event models, requirements for an historical event model and best practices in the application of event models within different domains. (journal paper, accepted for JWS2010).
- study on historical events definition and modeling requirements; use case for event annotations of cultural heritage objects (Workshop Events2010).
- experiments with pattern-based event extraction (accepted abstract at CLIN'11).
- prototype of Agora portal for event-based searching and browsing of cultural heritage collections (demo accepted at Museums at the Web'11)

Future work will accomplish the steps in the approach. We also consider experiments on the portability of the results to other domains and languages.

References

1. Buitelaar, P., Cimiano, P., Magnini, B. (eds.): Ontology learning from Text: Methods, Evaluation and Applications. IOS Press, Amsterdam (2005)
2. Cimiano, P.: Ontology Learning and Population from Text Algorithms, Evaluation and Application. Springer, Heidelberg (2006)
3. Cohen, W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: Proceedings of AAAI 1992. AAAI Press, Menlo Park (1992)
4. de Boer, V.: Ontology Enrichment from Heterogeneous Sources on the Web. PhD thesis, VU University, Amsterdam, The Netherlands (2010)
5. Fellbaum, C. (ed.): Wordnet: An Electronical Lexical Database. MIT Press, Cambridge (1998)
6. van Hage, W., Malaisé, V., de Vries, G., Schreiber, G., van Someren, M.: Combining ship trajectories and semantics with the simple event model (sem). In: EiMM 2009, New York, NY, USA, pp. 73–80 (2009)
7. Hyvönen, E., Alm, O., Kuittinen, H.: Using an ontology of historical events in semantic portals for cultural heritage. In: ISWC 2007 (2007)
8. Ide, N., Woolner, D.: Historical ontologies. In: Words and Intelligence II: Essays in Honor of Yorick Wilks, pp. 137–152 (2007)
9. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems, 72–79 (2001)
10. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Schneider, L.: Wonderweb deliverable d18. Technical report, ISTC-CNR (2003)
11. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proceedings of FOIS 2001, pp. 2–9. ACM, New York (2001)
12. Scherp, A., Franz, T., Saathoff, C., Staab, S.: F—a model of events based on the foundational ontology dolce+dns ultralight. In: K-CAP 2009, Redondo Beach (2009)
13. Shaw, R., Troncy, R., Hardman, L.: LODE: Linking open descriptions of events. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 153–167. Springer, Heidelberg (2009)

Enriching Ontologies by Learned Negation
Or How to Teach Ontologies Vegetarianism

Daniel Fleischhacker

KR & KM Research Group, University of Mannheim, Germany
daniel@informatik.uni-mannheim.de

1 Problem Statement

Ontologies form the basis of the semantic web by providing knowledge on concepts, relations and instances. Unfortunately, the manual creation of ontologies is a time-intensive and hence expensive task. This leads to the so-called knowledge acquisition bottleneck being a major problem for a more widespread adoption of the semantic web. Ontology learning tries to widen the bottleneck by supporting human knowledge engineers in creating ontologies. For this purpose, knowledge is extracted from existing data sources and is transformed into ontologies. So far, most ontology learning approaches are limited to very basic types of ontologies consisting of concept hierarchies and relations but do not use large amounts of the expressivity ontologies provide.

Negation is of great importance in ontologies since many common ideas and concepts are only fully expressible using negation. An example for the usefulness of negation is the notion of a *vegetarian* who is characterized by *not* eating meat. It is impossible to fully formalize this notion without applying negation at some level. Not stating these additional information on vegetarians would severely limit the possibilities to deduce new knowledge on vegetarians from the ontology by doing reasoning. Furthermore, negation is of great significance for assessing the quality of ontologies. Without it, ontologies may never get incoherent or inconsistent which is an important quality criterion. Additionally, with negations contained in ontologies, it is possible to use ontology debugging approaches more effectively.

Given all these points, we consider it important to put effort into a more elaborate research of automatic or semi-automatic learning of negation for enriching ontologies.

2 State of the Art

There is a large number of possible data sources all of them exposing different properties with respect to their structure and content. To handle these different inputs, ***ontology learning*** makes use of approaches from many different research areas which leads to a wide spectrum of different ontology learning methods [2].

Regarding the learning of negation there is little work so far. An example being the ***extraction of concept disjointness*** as a special case of negation. Haase and Völker [6] use lexico-syntactic patterns and their work is extended by Völker et al. [16] applying classification approaches on a number of different lexical and structural features in their LeDA tool. However, these approaches focus on the generation of disjointness of *atomic* classes and are not directly applicable for generating axioms containing complements of *complex* class expressions. Even if most negation axioms, i.e., axioms containing explicit negation, may be represented by disjointness, the representation in data (e.g., vegetarian as *someone who does not eat meat*) not necessarily resembles disjointness.

Another ontology learning method which also generates negation is implemented by the DL-Learner tool [11]. It uses ***inductive logic programming (ILP)*** to yield complex axioms describing concepts from a given ontology. Unfortunately, this method suffers from two issues. First, it is limited to using ontologies or data sets convertible to ontologies as data sources, thus it is not adequate to handle unstructured data and probably most semi-structured information. Secondly, the approach is not directly applicable to large data sets. This is mainly because of the dependency on reasoning for generating the relevant axioms which introduces scalability problems. Hellmann et al. [9] propose a method to extract fragments from the data to reduce it to a processable size but this could nevertheless lead to the loss of relevant data.

Texts are reasonable sources to extract knowledge about negation axioms, and detecting negation in texts could be a valid first step towards reaching this goal. Thus, work regarding the general ***detection of negation in biomedical texts*** is also of interest for learning negation. Most research in detecting negation in texts has been made in the biomedical domain where the approaches are used to extract data on the presence or absence of certain findings. This detection is mainly done by means of a list of negation markers and regular expressions [1], by additionally using linguistic approaches like grammatical parsing [10,5] or by applying machine learning techniques [13,12,14]. It is particularly important that the detection of negation also requires the identification of its scope, i.e., the parts of the sentence the negation is referring to. Even if some of the mentioned works might be usable on open-domain texts, there is no evaluation in this direction but only for the biomedical domain and thus there is no information on their performance for other domains. Furthermore, it is not clear if detected negations are similar to the ones required in ontologies.

Recently, there has been more work on ***negation detection for open-domain texts*** mainly driven by its usefulness for sentiment analysis [3] or contradiction detection [8]. Councill et al., who particularly concentrate on the task of detecting the scopes of negation, also evaluated their approach on product review texts using an appropriate, annotated gold standard which unfortunately seems not to be publicly available. Despite these recent works, detecting negation in open-domain texts remains an open problem.

3 Expected Contributions

The main contribution of this work is expected to be the development of approaches to ***enrich given ontologies with negation axioms extracted from texts*** as a part of an overall ontology learning approach and accompanied by a corresponding implementation. For this purpose, we will take already existing, manually engineered ontologies and add negation axioms extracted from free texts.

Negations in ontologies could provide great benefit for many application. In the field of biomedicine, one example would be an ontology containing information on different drugs. For some of these drugs, it is known that there are bacteria which are resistant against them. For instance, *methicillin-resistant Staphylococcus aureus* (MRSA) are strains of *Staphylococcus aureus* resistant against beta-lactam antibiotics. The axiom $\texttt{BetaLactamAntibiotic} \sqsubseteq \neg\exists\texttt{effectiveAgainst.MRSA}$ could be used to represent this in an ontology. Given such negation axioms, it would be possible to deduce from the ontology which drugs are not suitable for treating diseases caused by specific pathogens.

A second contribution will be developing and employing approaches to ***combine multiple ways of extracting negation***. This will help compensating possible shortcoming of certain approaches or data sources and to achieve better overall results.

When enriching ontologies by negation, we have to pay special attention to the ***maintenance of the ontology's consistency and coherence***. Without this, there is the risk of rendering the ontology inconsistent and less useful for reasoning tasks. Such inconsistencies do not have to come from the addition of the learned negation axioms themselves but may also arise from erroneous non-negation axioms added by the overall learning approach.

To be able to actually evaluate the results gained by extracting negations from different data sources, an appropriate evaluation strategy is necessary. Based on related work, we will ***develop methodologies suited for the evaluation***.

4 Methodology and Approach

In the following, we give an overview on the methodology which we want to follow to come up with the aforementioned contributions.

Negation Extraction from Text. We expect the detection of negation in textual data to be domain-dependent to a high degree. However, we will focus on the biomedical domain because of the large amount of work already done there regarding negation detection and the availability of expressive ontologies. There are several kinds of negations in texts which we will have to handle. Mostly, these kinds of textual negations are distinguishable into direct negation like caused by the word *not* and indirect negation recognizable by words like *doubt*, which introduce the negation solely by their semantics, and *misunderstanding*, where the semantics of negation is characterized by morphological markers like *mis-*. For the first manner of indirect negation, the lexical-semantic relation of antonymy may provide some additional hints for detection. This is why we already did experiments on detecting antonymy relations by means of relatedness and similarity measures. We will evaluate the approaches from the biomedical domain regarding their coverage for these different kinds of negation and develop approaches to treat the yet uncovered ones. To do this, we will most likely start with pattern-based detection approaches and then additionally apply machine learning methods.

For the enrichment of ontologies, we have to develop approaches to actually transfer the extracted textual negations into suitable logical negation which is not a trivial problem because of the ambiguity of natural language. Furthermore, we will evaluate the way negation is used in existing ontologies particularly regarding possible modeling errors made by humans and regarding the expressivity required for these negation axioms. Based on the findings, we will choose description logic fragments best suited for representing the learned negation while maintaining desirable computational properties.

An especially interesting approach is the combination of ways to learn from multiple data sources. As mentioned, this can help to compensate shortcomings in different approaches or data sources. LeDA [16] already combined different approaches but this is only done in course of their disjointness extraction algorithm and not directly applicable for combining arbitrary approaches. Having a more general way of combining different approaches, we could use it to integrate the negation axioms extracted by our proposed text-based system and other systems like DL-Learner [11].

Consistency Maintenance. The task of consistency maintenance has to be employed for the overall ontology learning and enrichment process and not only for the enrichment by negation axioms. Most ontology learning approaches produce confidence values for generated axioms. Thus, we have to deal with uncertainty like Haase and Völker [7] who also considered uncertainty to create consistent ontologies by ontology learning. We will apply similar debugging methods but also more general approaches like the one by Schlobach [15]. Regarding the overall learning approach, we will also explore methods of instantiating a feedback loop from debugging to the actual learning process. For the ontologies containing negation axioms, we are also able to compute different measures, e.g., the number of incoherent concepts widely seen as an indicator for an ontology's quality.

Evaluation. The evaluation of the correctness of the created negation axioms is also important for the overall goal of learning negation. As there is no standard way of evaluating these axioms, we will propose a new methodology. There are different ways of evaluating general ontology learning approaches [4]. For negations, it seems to be less desirable to use a gold standard especially since its manual creation is extremely labor-intensive for large data sources. Alternatively, we could use the learning approach in an application which benefits from a more expressive ontology. For our evaluation, we will look for such applications. Finally, we could let human domain experts evaluate the extracted axioms regarding their correctness. Even if this means that there is no possibility of computing the completeness for the extracted axioms with respect to a given data source, important values such as the inter annotator agreement may still be computed.

5 Conclusion

In this paper, we presented our plans to develop and implement approaches to enrich ontologies by complex negation axioms. As described above, we consider this beneficial for a couple of reasons. Having the results in the area of negation detection for biomedical texts and some for open-domain texts, we already have some foundations regarding negations in texts which should enable us to achieve first results soon. All in all, learning approaches for negation can assist humans in creating more thoroughly formalized ontologies and thus lead to a more expressive semantic web.

References

1. Chapman, W.W., Bridewell, W., Hanbury, P., Cooper, G.F., Buchanan, B.G.: A simple algorithm for identifying negated findings and diseases in discharge summaries. Journal of Biomedical Informatics 34(5), 301–310 (2001)

2. Cimiano, P., Mädche, A., Staab, S., Völker, J.: Ontology learning. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, 2nd edn., pp. 245–267. Springer, Heidelberg (2009)
3. Councill, I.G., McDonald, R., Velikovich, L.: What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In: Proc. of the Workshop on Negation and Speculation in Natural Language Processing, pp. 51–59 (2010)
4. Dellschaft, K., Staab, S.: On how to perform a gold standard based evaluation of ontology learning. 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. 228–241. Springer, Heidelberg (2006)
5. Gindl, S., Kaiser, K., Miksch, S.: Syntactical negation detection in clinical practice guidelines. Studies in Health Technology and Informatics 136, 187–192 (2008)
6. Haase, P., Völker, J.: Ontology learning and reasoning - dealing with uncertainty and inconsistency. In: Proc. of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW), pp. 45–55 (2005)
7. Haase, P., Völker, J.: Ontology learning and reasoning — dealing with uncertainty and inconsistency. 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)
8. Harabagiu, S., Hickl, A., Lacatusu, F.: Negation, contrast and contradiction in text processing. In: Proc. of the 21st national conference on Artificial intelligence, vol. 1, pp. 755–762 (2006)
9. 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, 25–48 (2009)
10. Huang, Y., Lowe, H.J.: A novel hybrid approach to automated negation detection in clinical radiology reports. Journal of the American Medical Informatics Association 14(3), 304–311 (2007)
11. Lehmann, J.: DL-Learner: Learning concepts in description logics. Journal of Machine Learning Research 10, 2639–2642 (2009)
12. Li, J., Zhou, G., Wang, H., Zhu, Q.: Learning the scope of negation via shallow semantic parsing. In: Proc. of the 23rd International Conference on Computational Linguistics, pp. 671–679 (2010)
13. Morante, R., Daelemans, W.: A metalearning approach to processing the scope of negation. In: Proc. of the 13th Conference on Computational Natural Language Learning, pp. 21–29 (2009)
14. Sarafraz, F., Nenadic, G.: Using SVMs with the command relation features to identify negated events in biomedical literature. In: Proc. of the Workshop on Negation and Speculation in Natural Language Processing, pp. 78–85 (2010)
15. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 226–240. Springer, Heidelberg (2005)
16. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 175–189. Springer, Heidelberg (2007)

Optimizing Query Answering over OWL Ontologies[⋆]

Ilianna Kollia

ECE School, National Technical University of Athens, Greece
ilianna2@mail.ntua.gr

Abstract. Query answering is a key reasoning task for many ontology based applications in the Semantic Web. Unfortunately for OWL, the worst case complexity of query answering is very high. That is why, when the schema of an ontology is written in a highly expressive language like OWL 2 DL, currently used query answering systems do not find all answers to queries posed over the ontology, i.e., they are incomplete. In this paper optimizations are discussed that may make query answering over expressive languages feasible in practice. These optimizations mostly focus on the use of traditional database techniques that will be adapted to be applicable to knowledge bases. Moreover, caching techniques and a form of progressive query answering are also explored.

1 Problem

Query answering is an important task in the Semantic Web since it allows for the extraction of information from data as specified by the user. The answers to queries are based not only on the explicitly stated facts but also on the inferred facts. In order to dervive such implicit facts, we distinguish between the terminological and the assertional part of an ontology [2]. The terminological part, called TBox, describes general information about the modeled domain of the ontology, e.g., the relationships between classes and properties. The assertional part, called ABox, contains concrete instance data, e.g., stating which indviduals belong to a class or are related with a property.

The derivation of the implicit facts of a knowledge base is done by reasoners and is a computational problem of high complexity. For example, OWL 2 DL entailment is known to be N2ExpTime-complete [7]. Hence the expressivity of the TBox, which defines how complex the background knowledge that will be used to derive implicit facts is, constitutes one source of complexity in query answering. The other source is the size of the ABox.

A query answer is a mapping from the variables appearing in the query to terms of the queried knowledge base such that replacing the variables with their mappings yields an entailed consequence of the ontology. The well known conjunctive queries contain variables which can be mapped to individuals and literals appearing in the ABox of the queried knowledge base. A naive algorithm for finding the answers of a conjunctive query w.r.t. a knowledge base would check which of the instantiated queries (formed by substituting the variables of the query with every individual appearing in the ABox) are entailed by the knowledge base. Hence such an algorithm would perform m^n entailment checks, where m is the number of individuals in the ontology and n is the number of

⋆ This work is partially funded by the EC Indicate project.

variables in the query. The cost of an entailment check depends on the used logic, i.e., by using a less expressive formalism than OWL 2 DL one can reduce the complexity.

Because of the high complexity of entailment checking in OWL 2 DL, it is evident that query answering becomes problematic. Our research tackles this problem attempting to achieve as high expressivity as possible as well as efficiency in practice.

2 State of the Art

To achieve practicality, current query answering systems work either with logics of low complexity or are incomplete in the sense that they compute only some of the answers w.r.t. TBoxes of higher complexity. Moreover, in order to achieve scalability they use databases for storing the instance data.

In order to deal with large amounts of data, the reduction of a $\mathcal{SHIQ}$ knowledge base to a disjunctive datalog program that entails the same set of ground facts as the original knowledge base has been explored [6]. In this way optimization methods from deductive databases, such as the join order optimization or the magic set transformation [3] can be used to answer queries. This technique is better suited for knowledge bases with large ABoxes but small TBoxes.

According to rewriting techniques that are currently applicable to OWL QL and OWL EL, the ontology is split into a schema and a data part. The data can be stored into relational databases or triple stores. The schema part can be used to rewrite the user's query into a union of one or more conjunctive queries (for OWL QL which is based on a family of description logics called DL-Lite [4]) or a datalog query (for OWL EL) [10,11] which can then be evaluated over the data part without taking the schema into account. This approach addresses scalability issues well but suffers from the fact that the size of the rewritten queries can be large making the evaluation difficult.

Materialization techniques have been employed by systems like Jena, Sesame, OWL-LIM. These techniques extend the ABox with implicit facts that are entailed by the TBox. They produce incomplete answers when queries are evaluated over an ontology with an expressive TBox. This happens because in expressive fragments of OWL due to the presence of disjunctive information it is not the case that a unique canonical model can be used to answer queries. Moreover, due to the presence of existential information it cannot be guaranteed that the models of an ontology are finite. Apart from being incomplete for query answering, the use of materialization techniques is problematic in case the data change frequently because in such case the frequent recomputation of the entailed implicit facts is required which is costly.

Approximations of ontologies written in highly expressive languages have also been used for query answering. In [9] one such technique has been presented which approximates an OWL DL ontology with a DL-Lite ontology producing sound but incomplete answers to queries. It should be stated though that in the case of conjunctive queries that do not contain non-distinguished variables, the described method provides sound as well as complete answers.

A couple of optimizations for conjunctive queries over OWL ontologies have been presented in [12] that take advantage of instance retrieval optimization techniques for tableau reasoners. Most of them are more suitable to be used with tableau and not with resolution based reasoners which is the type of reasoner we will use.

3 Proposed Approach and Methodology

A naive query answering algorithm that checks all possible mappings for query variables needs optimizations in order to deal with expressive languages and behave well in practice. In my PhD, starting from the highly expressive OWL 2 DL, an optimized query answering algorithm will be devised that will use resolution based reasoners for answering conjunctive queries and it will be extended to cover also queries about the schema of an ontology. Through this work, it will be seen whether optimizations can make query answering over OWL 2 DL feasible.

A first set of optimizations will be targeted at transferring techniques from relational and deductive databases [1] to knowledge bases. For example, since it has been observed that the order in which query atoms are evaluated is of critical importance to the running time of a query, techniques from databases, such as cost based query reordering, will be used to find optimal query execution plans. These techniques will be appropriately adapted to take into account the schema of the ontology apart from the data. As an example, let us consider a conjunctive query C(x), R(x,y), D(y), where x,y are individual variables, C,D are classes and R is a property. At the moment it is not clear whether the query is more efficiently evaluated with the query atoms in the order presented above or in a different order such as C(x), D(y), R(x,y) or R(x,y), C(x), D(y).

In many cases there is no need to consider all the elements of the data part as possible mappings for query variables in conjunctive queries and hence avoid checking whether all of them lead to the entailment of the instantiated queries by the queried knowledge base. This can be so, either because the user is not interested in answers belonging to some set or because some sets of mappings are not relevant w.r.t. a query. Such cases will be identified and only an appropriate subset of the possible mappings for query variables will be checked leading hopefully to an important reduction in the running time of queries. Moreover, efficient caching techniques will be used to store parts of the models of the queried ontology since it holds that, queries instantiated by many different mappings and checked afterwards for entailment by the queried ontology, often use the same parts of models. Hence saving these models will avoid the reconstruction of them every time they are needed hopefully reducing the execution time of queries. This is especially useful in the highly expressive OWL 2 DL in which the construction of models requires a substantial amount of time.

The query answering algorithm will be made to work progressively, i.e., to output query answers as soon as they are computed, outputing first the answers that are easily computed and then answers that are more difficult to be found. For example, the answers coming from the explicitly stated facts in the ABox can be found and given to the user relatively quickly. Answers which require reasoning are more difficult to be computed and require more time. What is more, even between mappings that require reasoning to decide whether they constitute answers, the computation time needed differs substantially. This happens because in order to decide whether different mappings consistute answers, the reasoner might have to build models of different size and complexity. For example, the amount of backtracking that the reasoner performs while trying different possibilities that arise from disjunctions defines the complexity of a model and hence the time that is needed to be constructed. This, in turn, influences the time needed to decide whether a mapping constitutes an answer or not. A more complex setting will then be adopted in which query answers are given to the user in the order of decreased relevance. This includes the definition of appropriate relevance criteria. The profile of the user who types the query can be exploited to define such measures of relevance. Since in highly expressive languages the time to compute all the answers for a query is high, through progressive query answering the user is given some answers to work with as soon as they are derived and more answers as time passes. However, the user should expect incomplete answers since some "hard" answers cannot be computed in reasonable time. The user may, therefore, be interested in the degree of (in)completeness of the used system which can be computed and presented to him.

The above described algorithm will be developed in conjuction with the SPARQL query language which is an RDF based query language that has recently been extended by W3C to find query answers under the OWL entailment relation (the OWL entailment regime of SPARQL [5]). In SPARQL a new class of powerful queries can be written which go beyond conjunctive queries. These queries allow variables in place of classes, object and data properties of OWL axioms apart from individuals and literals and need different optimizations than the ones applicable to conjunctive queries. Such queries have only partly been considered [13].

The steps that will be followed during the research are briefly described below. First, the formalized optimizations and techniques will be implemented in a system that will use SPARQL as a query language. As explained above, we will start with ontologies expressed in OWL 2 DL and see whether the applicable optimizations reduce the running time of queries to such extent that query answering becomes more feasible. In case the time for query answering is not acceptable even with the use of the considered optimizations, we will use techniques like knowledge compilation to approximate OWL DL ontologies with simplified versions of them of lower complexity and see how the use of these simplified ontologies affects the query answering times.

4 Results

A first attempt towards an optimized algorithm has been made. In particular, SPARQL has already been extended to allow the use of OWL inference for computing query answers. A cost based query reordering approach that seems to work well with conjunctive queries has been developed. A couple of optimizations have been made for the new class of expressive queries that can be represented in SPARQL. Such optimizations include the use of query rewriting techniques that transform the initial query to an equivalent one that can be evaluated more efficiently, the use of the class and property hierarchy of the queried ontology to prune the search space of candidate bindings for query variables and the use of more specialized tasks of OWL reasoners than entailment checking to speed query execution. The proposed optimizations can reduce query execution time by up to three orders of magnitude [8]. [1]

[1] This work has been done in collaboration with Dr Birte Glimm and Professor Ian Horrocks in the Oxford University Computing Laboratory.

5 Conclusion

Taking into account the fact that naive query answering techniques are impractical over expressive languages like OWL 2 DL, in my PhD I will try to devise optimized algorithms that will hopefully behave well in practice. In order to achieve this, techniques from relational and deductive databases will be transferred to knowledge bases and an evaluation of their applicability and efficiency will be made. For example, we will analyse whether the magic set technique for finding relevant parts of data w.r.t. queries and rules can be extended in our setting, where we have disjunction and existential quantification in the rule head. The results taken so far are promising. However, more tests need to be performed using a greater range of ontologies and queries.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1994)
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. Beeri, C., Ramakrishnan, R.: On the power of magic. In: PODS. pp. 269–284 (1987)
4. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
5. 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)
6. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. J. Autom. Reason. 39, 351–384 (2007)
7. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Brewka, G., Lang, J. (eds.) Proc. 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 274–284. AAAI Press, Menlo Park (2008)
8. Kollia, I., Glimm, B., Horrocks, I.: SPARQL Query Answering over OWL Ontologies (2010), accepted for publication, http://www.comlab.ox.ac.uk/files/3681/paper.pdf
9. Pan, J.Z., Thomas, E.: Approximating OWL-DL ontologies. In: Proceedings of the 22nd National Conference on Artificial Intelligence, vol. 2, pp. 1434–1439. AAAI Press, Menlo Park (2007)
10. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. Journal of Applied Logic 8(2), 186–209 (2010)
11. Rosati, R.: On conjunctive query answering in EL. In: Proceedings of the 2007 International Workshop on Description Logic (DL 2007). CEUR Electronic Workshop Proceedings (2007)
12. Sirin, E., Parsia, B.: Optimizations for answering conjunctive abox queries: First results. In: Proc. of the Int. Description Logics Workshop, DL (2006)
13. 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 (2007), CEUR-WS.org

Hybrid Search
Ranking for Structured and Unstructured Data*

Daniel M. Herzig

Institute AIFB, Karlsruhe Institute of Technology, Germany
herzig@kit.edu

Abstract. A growing amount of structured data is published on the Web and complements the textual content. Searching the textual content is performed primarily by the means of keyword queries and Information Retrieval methods. Structured data allow database-like queries for retrieval. Since structured and unstructured data occur often as a combination of both, are embedded in each other, or are complementary, the question of how search can take advantage of this hybrid data setting arises. Of particular interest is the question of how ranking as the algorithmic decision of what information is relevant for a given query can take structured and unstructured data into account by also allowing hybrid queries consisting of structured elements combined with keywords. I propose to investigate this question in the course of my PhD thesis.

1 Introduction

Currently, an increasing amount of structured data is published on the Web according to the Linked Data principles. This structured data supplements the textual, unstructured data already available on the Web and thereby provides the basis for new ways of searching the Web. The structured data is available in several ways. On the one hand there are data sets available as purely structured RDF independent from a text base, and on the other hand there is structured data embedded directly in textual data via RDFa or data extracted from texts. Taking advantage of this heterogenous environment promises to improve search by making it possible to answer more kinds of information needs, because some information needs benefit greatly from structured data, e.g. "What is the population of Berlin?". Here, the answer is a fact assertion, whereas other information needs are better addressed with textual documents, e.g. "Why did Heinrich von Kleist commit suicide?", where a potential answers might be his suicide note, if at all, but certainly not a fact assertion. Moreover, texts can hold sentiments, preferences and opinions, which are often supported by facts and data. Therefore, a hybrid data scenario holds also the possibility to examine the retrieval of opinions or different views on a topic and the facts supporting them. Thus far, document and fact retrieval are often regarded as two separate disciplines and

* This work was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen project (grant 01IA08005K).

the combination of both for search is not yet investigated in a satisfying way[1]. This thesis is situated between these two disciplines and combines them on the data and on the query level. We call this scenario *Hybrid Search*. However, search comprises the entire technical spectrum from indexing to the user interface. This thesis concentrates on ranking, which is a core method of search and crucial for its effectiveness. The goal of this thesis is to investigate a unified ranking framework for hybrid search as the search on structured and unstructured data with queries consisting of keywords and structured elements. The question this thesis addresses is how structured and unstructured data can be used to improve search and how hybrid queries can be answered on hybrid data.

2 Problem Definition

This thesis addresses the problem of ranking on hybrid data for hybrid queries. The frame of this research problem is defined by the following data and query model: The proposed data model follows the RDF data model with Named Graphs[1] and is represented as a graph $G(R, L, E_R, E_L, E_G, \hat{G})$ consisting of resource nodes R, edges E_R connecting resource nodes, edges E_L connecting resource nodes to literal nodes L, and edges E_G connecting resource nodes to Named Graphs $\hat{G}$, which are graphs $\hat{G}(R', L', E'_R, E'_L)$ consisting of subsets of the elements of G, e.g. $R' \subset R$. Textual data is integrated following the same modeling paradigm and using the already mentioned modeling constructs. Each text entity is represented by a resource of the type *textual document*. This resource has one edge, labelled *content*, pointing to a literal node holding the textual information. In a later stage, there can be more edges providing more fine grated distinctions, such as *headline*, *paragraph*, etc. All triples comprised by the textual information of one textual entity form a Named Graph $\hat{g} \in \hat{G}$, as illustrated in Figure 1 by the dashed circle. The data model is a simplified RDF model with Named Graphs and allows to use RDF data easily.

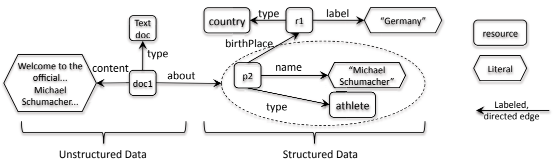


Fig. 1. Illustration of the data model. A textual document on the left side and structured data on the right side.

Queries to this data model should have a seamless flexibility ranging from purely textual keyword queries, over hybrid queries, to crisp structured queries. A hybrid query q can consist of a *structured* part q_s and a *textual* part q_t, i.e. $q = q_s \wedge q_t$. If one part is empty, the query is either purely textual or purely

[1] Named Graphs: http://www.w3.org/2004/03/trix/

structured. The structured part q_s follows the SPARQL query language and is a set of graph patterns, $q_s = \{q_{s1}, q_{s2}, ...\}$. The textual part q_t allows to associate a keyword query kw to each variable, i.e. $q_t = \{q_{t_i} | q_{t_i} = (x_i, kw), x_i \in Var(q)\}$. For example, assume the information need: *"Formula One drivers who moved to Switzerland"*[2], which is illustrated in Figure 2. The result to such a query are bindings to the distinguished variables. This model allows to represent purely structured, hybrid, and purely textual queries. A purely textual query, i.e. simple keyword query, would be the query in Fig. 2 without line (1). This query model is a close adaptation of the model by [3].

```
Select ?x where {
   ?x   rdf:type ns:FormulaOneRacer          # (1)
   ?x   {moved to Switzerland} }       # (2)
```

Fig. 2. Illustration of the query model, consisting of the a structured part, i.e. triple patterns (1) and unstructured part, i.e. keyword patterns (2).

3 State of the Art

Related fields of research are IR, in particular Web IR, ranking of structured data and databases, and the already existing work on hybrid. This section briefly outlines related work of these fields to the proposed approach in the Section 4.

Ranking originated in the IR community, where the retrieval units are textual documents. The main notions for ranking are descriptive statistics about the documents and the entire document corpus, such as term frequency. One of the most used algorithm in this line is BM25[4]. Language model approaches are increasingly applied, because of their formal basis in probability theory. Therefore, language models will be the basis for the proposed ranking approach. The work by [5] are of particular interest, since it builds on language models for structured retrieval, combines keywords and structured queries and addresses structural mismatches between query and data structure. However, structure means here the document and sentence structure and not a graph based data model. Ranking for Web search deals not just with fixed document corpora, but with the entire scale of the Web. However, it can take advantage of the link structure of the Web. Exemplars using this link analysis are foremost the well known works by [6] and [7]. Translating the idea of [6,7] for ranking data on the Web has been studied by [8,9]. Also concepts of XML retrieval [10] as the retrieval of semi-structured documents needs to be addressed. The parallels are here that elements in the XML scenario are similar to resources in ours. Ranking for databases draws on the advantage that the data schema is strict and rigid, which is not the case in our setting. Still, the idea of correlations[11] between values needs to be investigated. Combing text and structured data, i.e. hybrid approaches, such as [12], which uses a domain ontology for retrieving documents, or [13], which retrieves documents as well as factual assertions. However, the data setting is different to ours. Most notably is the approach by [3], which could be

[2] Topic GC-2009-49 taken from [2].

used as a reference in an evaluation, because it supports keyword augmented structured queries similar to ours. However, the approach does not take documents into account, and is centered around triples as the atomic units, where as our proposed approach regards entities, i.e. URIs of *subjects* respectively *objects*, as the atomic pieces.

4 Proposed Approach

Starting point are retrieval methods similar to [3,5] applied to a hybrid scenario, because they have proven to be applicable for similar settings and are the state of the art in IR. Following the idea of language models, we rank result graphs according to the probability of being an result graph g to the given query q, i.e. $P(g|q)$. The structured part of the query is regarded as a constraint for the beginning and can be relaxed later. It fulfills the purpose of selecting candidate results. Since q_s determines the shape of the result graphs, all possible graphs share the same structure. Therefore, the rank of a result depends only on the aspects, which differentiate the results, i.e. the bindings to the variables and their relations to q_t. Therefore, we can reduce the ranking to $P(g|q) \propto \prod_{i=1}^{n} P(q_i|x_i)$, with $q_i = q_{t_j} \wedge q_{s_k}, x_i \in q_{t_j}, g_{s_k}$, the keyword and triple patterns associated to variables x_i.

$$P(g|q) \propto P(g) \cdot P(q|g) \propto P(g) \cdot \prod_{i=1}^{n} P(q_i|x_i) \tag{1}$$

$P(q_i|x_i)$ is computed in two ways depending whether q is a purely structured query or not. If it is purely structured, the query is crisp and we assume that the information need is entirely captured. The ranking is then based on the popularity of the resulting URIs. If a textual part is present, the information need is rather imprecisely captured making it necessary to rank the results by measuring the relation of each keyword k of the textual part to the corresponding variable, see equation (2).

$$P(q_i|x_i) = \begin{cases} \prod_{k \in q_{t_i}} \alpha P_t(k|x_i) + (1-\alpha)P_t(k) & \text{if } q_t \neq \emptyset \\ \prod_{x_i} P_s(x_i) & \text{if } q_t = \emptyset \end{cases} \tag{2}$$

If a textual part is present, several models for computing $P_t(k|x_i)$ will be investigated: Starting with a simple one, which takes all textual information of x_i as one bag-of-words and a more fine grained one, which takes the edges from x_i to neighboring nodes into account.This ranking model is an initial model for the study of search in the hybrid scenario. Possible future directions of research are to extend this model and integrate more of the semantics provided by the underlying data.

5 Evaluation Methodology

The widest acceptance in IR for evaluating ranking approaches has the so-called *Cranfield methodology*[14]. It provides well studied grounds and will be the basis of the evaluation in line with [15]. However, the setting needs to be adapted to the hybrid scenario. This can be done by adding structured elements to the keyword queries of [15] and by using datasets, which are a combination of structured and unstructured data, e.g. the combination of *Wikipedia* and *dbpedia*.

6 Conclusion

The goal of this thesis is to investigate a unified ranking methodology for search on hybrid data using adaptive queries. The proposed approach builds on a graph based data model, which is compatible to RDF and incorporates textual documents. The query model allows seamless querying ranging from purely textual queries, to hybrid queries, and to purely structured queries. The ranking approach builds methodologically on language models. The evaluation methodology uses existing standards from the IR community, if applicable, but needs to be adapted to the hybrid context. The question this thesis addresses is how the combination of structured and unstructured data can be used to improve search.

References

1. Weikum, G.: DB & IR: both sides now. In: SIGMOD (2007)
2. Santos, D., Cabral, L.M.: GikiCLEF: crosscultural issues in an international setting: asking non-English-centered questions to wikipedia. In: CLEF (2009)
3. Elbassuoni, S., Ramanath, M., Schenkel, R., Sydow, M., Weikum, G.: Language-model-based ranking for queries on RDF-graphs. In: CIKM 2009 (2009)
4. Robertson, S., Zaragoza, H.: The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval 3(4), 333–389 (2010)
5. Zhao, L., Callan, J.: Effective and efficient structured retrieval. In: CIKM (2009)
6. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: ACM-SIAM, San Francisco, California, United States, pp. 668–677 (1998)
7. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: WWW, Brisbane, Australia, pp. 107–117 (1998)
8. Harth, A., Kinsella, S., Decker, S.: Using naming authority to rank data and ontologies for web search. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 277–292. Springer, Heidelberg (2009)
9. Delbru, R., Toupikov, N., Catasta, M., Tummarello, G., Decker, S.: Hierarchical link analysis for ranking web data. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 225–239. Springer, Heidelberg (2010)
10. Lalmas, M.: XML Retrieval. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, San Francisco (2009)
11. Chaudhuri, S., Das, G., Hristidis, V., Weikum, G.: Probabilistic ranking of database query results. In: VLDB, pp. 888–899 (2004)
12. Rocha, C., Schwabe, D., Aragao, M.P.: A hybrid approach for searching in the semantic web. In: World Wide Web, WWW 2004, New York, NY, USA (2004)
13. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid search: Effectively combining keywords and semantic searches. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 554–568. Springer, Heidelberg (2008)
14. Cleverdon, C.: The CRANFIELD Tests on Index Langauge Devices. Aslib (1967)
15. Halpin, H., Herzig, D.M., Mika, P., Blanco, R., Pound, J., Thompson, H.S., Tran, D.T.: Evaluating ad-hoc object retrieval. In: IWEST 2010, ISWC (2010)

Author Index