Lora Aroyo Grigoris Antoniou
Eero Hyvönen Annette ten Teije
Heiner Stuckenschmidt Liliana Cabral
Tania Tudorache (Eds.)

# The Semantic Web: Research and Applications

LNCS 6089

7th Extended Semantic Web Conference, ESWC 2010
Heraklion, Crete, Greece, May/June 2010
Proceedings, Part II

2 Part II

Springer

Lecture Notes in Computer Science 6089

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

Lora Aroyo Grigoris Antoniou
Eero Hyvönen Annette ten Teije
Heiner Stuckenschmidt Liliana Cabral
Tania Tudorache (Eds.)

# The Semantic Web: Research and Applications

7th Extended Semantic Web Conference, ESWC 2010
Heraklion, Crete, Greece, May 30 – June 3, 2010
Proceedings, Part II

Springer

Volume Editors

Lora Aroyo
Free University Amsterdam, The Netherlands
E-mail: l.m.aroyo@cs.vu.nl

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

Eero Hyvönen
Aalto University, Finland
E-mail: eero.hyvonen@tkk.fi

Annette ten Teije
Free University Amsterdam, The Netherlands
E-mail: annette@cs.vu.nl

Heiner Stuckenschmidt
Universität Mannheim, Germany
E-mail: heiner@informatik.uni-mannheim.de

Liliana Cabral
The Open University, Milton Keynes, UK
E-mail: l.s.cabral@open.ac.uk

Tania Tudorache
Stanford Biomedical Informatics Research Center,USA
E-mail: tudorache@stanford.edu

Library of Congress Control Number: 2010927493

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

ISSN 0302-9743
ISBN-10 3-642-13488-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-13488-3 Springer Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

# Preface

This volume contains papers from the technical program of the 7th Extended Semantic Web Conference (ESWC 2010), held from May 30 to June 3, 2010, in Heraklion, Greece. ESWC 2010 presented the latest results in research and applications of Semantic Web technologies. ESWC 2010 built on the success of the former European Semantic Web Conference series, but sought to extend its focus by engaging with other communities within and outside Information and Communication Technologies, in which semantics can play an important role. At the same time, ESWC has become a truly international conference.

Semantics of Web content, enriched with domain theories (ontologies), data about Web usage, natural language processing, etc., will enable a Web that provides a qualitatively new level of functionality. It will weave together a large network of human knowledge and make this knowledge machine-processable. Various automated services, based on reasoning with metadata and ontologies, will help the users to achieve their goals by accessing and processing information in machine-understandable form. This network of knowledge systems will ultimately lead to truly intelligent systems, which will be employed for various complex decision-making tasks. Research about Web semantics can benefit from ideas and cross-fertilization with many other areas: artificial intelligence, natural language processing, database and information systems, information retrieval, multimedia, distributed systems, social networks, Web engineering, and Web science.

To reflect its expanded focus, the conference call for research papers was organized in targeted tracks:

- Mobility
- Ontologies and Reasoning
- Semantic Web in Use
- Sensor Networks
- Services and Software
- Social Web
- Web of Data
- Web Science

The research papers program received 245 full paper submissions, which were first evaluated by the Program Committees of the respective tracks. The review process included evaluation by Program Committee members, discussions to resolve conflicts, and a metareview for each potentially acceptable borderline submission. After this a physical meeting among Track and Conference Chairs was organized to see that comparable evaluation criteria in different tracks had been used and to discuss remaining borderline papers. As a result, 52 research papers were selected to be presented at the conference and are included in the

proceedings. The ESWC 2010 proceedings also include ten PhD symposium papers presented at a separate track preceeding the main conference, and 17 demo papers giving a brief description of the system demos that were accepted for presentation in a dedicated session during the conference.

ESWC 2010 was happy to have had three keynote speakers and a dinner talk by high-profile researchers:

- Noshir Contractor, the Jane S. & William J. White Professor of Behavioral Sciences in the School of Engineering, School of Communication and the Kellogg School of Management at Northwestern University
- Sean Bechhofer, lecturer in the Information Management Group within the School of Computer Science at the University of Manchester
- Wolfgang Wahlster, Director and CEO of the German Research Center for Artificial Intelligence and a professor of Computer Science at Saarland University (Saarbruecken, Germany)
- Aldo Gangemi, senior researcher at the CNR Institute of Cognitive Sciences and Technology in Rome, and head of the Semantic Technology Lab.

Special thanks go to all the Chairs, Program Committee members, and additional reviewers of the different refereed tracks who all contributed to ensuring the scientific quality of ESWC 2010. Many thanks also go to the members of the Organizing Committee for their hard work in selecting outstanding tutorials, workshops, panels, lightning talks, and posters. We would like to also thank the Sponsorship Chair for reaching out to industry and various organizations supporting the 2010 edition of the conference, as well as the local organization, website and conference administration team put together by STI International for their excellent coordination during the conference preparation. Finally, we would like to thank the Proceedings Chair for the hard work in preparing this volume, Springer for the support with the preparation of the proceedings, and the developers of the EasyChair conference management system, which was used to manage the submission and review of papers, and the production of this volume.

May 2010

Lora Aroyo
Grigoris Antoniou
Eero Hyvönen
Annette ten Teije
Heiner Stuckenschmidt
Liliana Cabral
Tania Tudorache

# Organization

## Organizing Committee

| | |
|---|---|
| General Chair | Lora Aroyo (VU University Amsterdam, The Netherlands) |
| Program Chairs | Grigoris Antoniou (University of Crete, Greece)<br>Eero Hyvönen (Helsinki University of Technology, Finland) |

**Events Chairs**

| | |
|---|---|
| Workshop Co-chairs | Kalina Boncheva (University of Sheffield, UK)<br>Nicola Henze (University of Hannover, Germany) |
| Tutorials Co-chairs | Anna Fensel (FTW, Austria)<br>Aldo Gangemi (ISTC-CNR, Italy) |
| Demos/Posters Co-chairs | Liliana Cabral (OU, UK)<br>Tania Tudorache (Stanford University, USA) |
| PhD Symposium Co-chairs | Annette ten Teije (VU University Amsterdam, The Netherlands)<br>Heiner Stuckenschmidt (University of Mannheim, Germany) |
| Semantic Web Technologies Co-ordinators | Harith Alani (OU, UK)<br>Martin Szomszor (City eHealth Research Center, UK) |
| News from the Front Latest Results from EU Projects | Lyndon Nixon (STI International, Austria)<br>Alexander Wahler (STI International, Austria) |
| Panel Chair | Carlos Pedrinaci (OU, UK) |
| Proceedings Chair | Yiannis Kompatsiaris (ITI, Greece) |
| Sponsorship Chair | Manolis Koubarakis (University of Athens, Greece) |
| Publicity Chair | Valentina Prescutti (ISTC-CNR, Italy) |

**Local Organization Chairs**

| | |
|---|---|
| Treasurer | Alexander Wahler (STI International, Austria) |
| Local Organizer and Conference Administrator | Lejla Ibralic-Halilovic (STI International, Austria) |

## Program Committee - Mobility Track

### Track Chairs

Matthias Wagner and Hans Gellersen

### Members

Sebastian Böhm
Eugen Freiter
Johan Koolwaaij
Christian Kray
Thorsten Liebig
Marko Luther
Jerome Picault
Myriam Ribiere
Thomas Strang
Juan Ignacio Vazquez

## Program Committee - Ontologies and Reasoning Track

### Track Chairs

Jeff Pan and Adrian Paschke

### Members

Franz Baader
Nick Bassiliades
Paolo Bouquet
Diego Calvanese
Huajun Chen
Bernardo CuencaGrau
Mathieu D'Aquin
Martin Dzbor
Thomas Eiter
Giorgos Flouris
Tim Furche
Pascal Hitzler
Laura Hollink
Zhisheng Huang
Elisa Kendall
Francisco Martín-Recuerda
Diana Maynard
Leora Morgenstern
Boris Motik
Leo Obrst
Bijan Parsia
Dimitris Plexousakis
Axel Polleres
Guilin Qi
Yuzhong Qu
Riccardo Rosati
Sebastian Rudolph
Alan Ruttenberg
Harald Sack
Kai-Uwe Sattler
Stefan Schlobach
Michael Schroeder
Pavel Shvaiko
Michael Sintek
Kavitha Srinivas
Giorgos Stamou
Giorgos Stoilos
Umberto Straccia
Heiner Stuckenschmidt
Edward Thomas
Robert Tolksdorf
Holger Wache
Kewen Wang
Benjamin Zapilko
Yuting Zhao

## Program Committee - Semantic Web In-Use Track

### Track Chairs

Stuart Campbell and Sören Auer

### Members

David Aumueller
Domenico Beneventano
Andreas Blumauer
John Breslin
Vadim Chepegin
Richard Cyganiak
Andreas Doms
Muriel Foulonneau
Tim Furche
Martin Gaedke
John Goodwin
Mark Greaves
Gunnar Aastrand Grimnes
Tudor Groza
Olaf Hartig
Bernhard Haslhofer
Sebastian Hellmann
Martin Hepp
Robert Hoehndorf
Martin Kaltenböck
Birgitta König-Ries
Jens Lehmann
Steffen Lohmann
Markus Luczak-Rösch
Michele Missikoff
Claudia Müller
Axel-Cyrille Ngonga Ngomo
Massimo Paolucci
Tassilo Pellegrini
Barbara Pirillo
Yves Raimond
Harald Sack
Leo Sauermann
Sebastian Schaffert
Bernhard Schandl
Stefan Schulte
Jo Walsh
Michael Witbrock
Jun Zhao
Sven Abels
Georgi Pavlov

## Program Committee – Sensor Networks Track

### Track Chairs

Alan Smeaton and Oscar Corcho

### Members

Luis Bermudez
Boyan Brodaric
Michael Compton
Ixent Galpin
Raúl García-Castro
Alasdair Gray
Cory Henson
Jonas Jacobi
Krzysztof Janowicz
Laurent Lefort
Kirk Martinez
Holger Neuhaus
Noel O'Connor
Gregory O'Hare
Kevin Page
Vinny Reynolds
Ingo Simonis
Heiner Stuckenschmidt
Kerry Taylor
Andreas Wombacher

## Program Committee - Services and Software Track

### Track Chairs

Andreas Metzger and Elena Simperl

### Members

Stephan Bloehdorn
Matthias Book
Antonio Bucchiarone
Cyril Carrez
Dragan Gasevic
Dragan Ivanovic
Dimka Karastoyanova
Mick Kerrigan
Reto Krummenacher
Stephen Lane
Kim Lauenroth
Tiziana Margaria
Barry Norton
Michael Parkin
Carlos Pedrinaci
Srinath Perera
Pierluigi Plebani
Noël Plouzeau
Dumitru Roman
Fabrizio Silvestri
Michael Stollberg
Martin Treiber
Sanjiva Weerawarana

## Program Committee - Social Web Track

### Track Chairs

Julita Vassileva and Jie Bao

### Members

Sören Auer
Scott Bateman
Dave Beckett
Edward Benson
Shlomo Berkovsky
Dave Braines
John Breslin
Chris Brooks
Francesca Carmagnola
Federica Cena
Richard Cyganiak
Antonina Dattolo
Mike Dean
Darina Dicheva
Yihong Ding
Ying Ding
Jon Dron
Anna Fensel
Steve Harris
Tom Heath
Aidan Hogan
Akshay Java
Robert Jäschke
Lalana Kagal
Dae-Ki Kang
Pranam Kolari
Georgia Koutrika
Milos Kravcik
Markus Krötzsch
Tsvi Kuflik
Juanzi Li
Enrico Motta
Meenakshi Nagarajan
Daniel Olmedilla
Alexandre Passant
Jyotishman Pathak

Guilin Qi
Yuzhong Qu
Juan F. Sequeda
Paul Smart
Sergey Sosnovsky
Steffen Staab
Christopher Thomas
Haofen Wang
Mary-Anne Williams
Jie Zhang
Haizheng Zhang
Lina Zhou
Cristina Gena

## Program Committee – Web of Data Track

### Track Chairs

Paul Groth and Denny Vrandecic

### Members

Harith Alani
Anupriya Ankolekar
Christian Bizer
Uldis Bojars
Paolo Bouquet
Simone Braun
Vinay Chaudhri
Anne Cregan
Philippe Cudré-Mauroux
Richard Cyganiak
Peter Haase
Harry Halpin
Andreas Harth
Tom Heath
Martin Hepp
Rinke Hoekstra
Dietmar Jannach
Craig Knoblock
Georgi Kobilarov
Peter Mika
Malgorzata Mochol
Wolfgang Nejdl
Alexandre Passant
Yuzhong Qu
Yves Raimond
Leo Sauermann
Ansgar Schrep
Juan F. Sequeda
Patrick Sinclair
Katharina Siorpaes
Harold Solbrig
York Sure
Martin Szomszor
Hideaki Takeda
Jamie Taylor
Mischa Tuffield
Valentin Zacharias
Jun Zhao
Thanh Tran

## Program Committee - Web Science Track

### Track Chairs

Noshir Contractor and Leslie Carr

### Members

Ian Brown
Harry Halpin
Nick Gibbins
Kieron O'Hara

## Program Committee - Demo and Poster Track

### Track Chairs

Liliana Cabral and Tania Tudorache

### Members

Harith Alani
Sören Auer
Neil Benn
John Breslin
Christopher Brewster
Emilia Cimpian
Gianluca Correndo
Philippe Cudré-Mauroux
Mathieu d'Aquin
Danica Damljanovic
Klaas Dellschaft
Ying Ding
Leigh Dodds
Sean Falconer
Anna Fensel
Miriam Fernandez
Raúl García-Castro
Hugh Glaser
Tudor Groza
Olaf Görlitz
Peter Haase
Connor Hayes
Tom Heath
Masahiro Hori
Matthew Horridge
Luigi Iannone
Holger Lewen
Vanessa Lopez
Alexander Löser
Francisco Martin-Recuerda
Diana Maynard
Adrian Mocan
Knud Möller
Lyndon Nixon
Natasha Noy
Martin O'Connor
Alexandre Passant
Carlos Pedrinaci
Chantal Reynaud
Marco Rospocher
Marta Sabou
Elena Simperl
Patrick Sinclair
Katharina Siorpaes
Nenad Stojanovic
Heiner Stuckenschmidt
Mari Carmen Suárez-Figueroa
Martin Szomszor
Vlad Tanasescu
Thanh Tran
Mischa Tuffield
Giovanni Tummarello
Johanna Voelker
Holger Wache
Stuart Wrigley
Milena Yankova

## Program Committee - PhD Symposium

### Track Chairs

Heiner Stuckensmidt and Annette ten Teije

**Members**

Karl Aberer
Barry Bishop
Paolo Bouquet
Paul Buitelaar
Philipp Cimiano
Isabel Cruz
Jérôme Euzenat
Fabien Gandon
Chiara Ghidini
Paul Groth
Siggi Handschuh
Patrick Lambrix
Thomas Lukasiewicz
Gergely Lukácsy
Diana Maynard
Peter Mika
Dunja Mladenic
Mathias Niepert
Natasha Noy
Jeff Pan
Terry Payne
Valentina Presutti
Sebastian Rudolph
Marta Sabou
Stefan Schlobach
Pavel Shvaiko
Elena Simperl
Sergej Sizov
Umberto Straccia
Valentina Tamma
Johanna Voelker
Holger Wache
Mathieu d'Aquin
Jacco van Ossenbruggen

## Referees

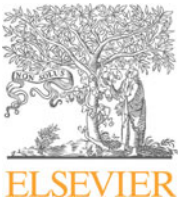
Alessandro Adamou
Dimitra Alexopoulou
Sofia Angeletou
Zelalem Bachore
Samantha Bail
Peter Barna
Sebastian Boehm
Stefano Bortoli
Daniele Broccolo
Stuart Campbell
Leyla Jael García Castro
Matteo Casu
Gong Cheng
Vadim Chepegin
Smitashree Choudhury
Jianfeng Du
Kai Eckert
Giorgos Flouris
Angela Fogarolli
Flavius Frasincar
Eugen Freiter
Fatih Gedikli
George Giannakopoulos
Gunnar Aastrand Grimnes
Ralf Heese
Daniel Herzig
Geerd-Dietger Hoffmann
Matthew Horridge
Wei Hu
Giovambattista Ianni
Ekaterini Ioannou
Antoine Isaac
Prateek Jain
Malte Kiesel
Sheila Kinsella
Pavel Klinov
Haris Kondylakis
Efstratios Kontopoulos
Johan Koolwaaij
Jacek Kopecky
Jakub Kotowski
Kalliopi Kravari
Christian Kray
Thomas Krennwallner
Vikash Kumar
Günter Ladwig
Georg Lausen
Thorsten Liebig
Dong Liu
Xin Liu
Nuno Lopes
Vanessa Lopez
Marko Luther
Theofilos Mailis
Maria Maleshkova
Michael Martin
Carlos Nana Mbinkeu
Georgios Meditskos
Christian Meilicke
Benedikt Meuthrath
Franco Maria Nardini
Nadejda Nikitina
Olaf Noppens
Vit Novacek
Andrea Giovanni Nuzzolese
Jasmin Optiz
Magdalena Ortiz

Ignazio Palmisano
Rahul Parundekar
Rafael Penaloza
Srinath Perera
Jerôme Picault
Valentina Presutti
Maryam Ramezani
Nataliya Rassadko
Michael Reiter
Yuan Ren
Myriam Ribiere
Myriam Ribire
Frank Sawitzki
Anne Schlicht
Stefan Schulz
Kostyantyn Shchekotykhin
Wei Shen
Nikolaos Simou
Sergey Sosnovsky
Natalia Stash
Heiko Stoermer
Thomas Strang
Olga Streibel
Thanassis Tiropanis
George Tsatsaronis
Joerg Unbehauen
Johanna Völker
Inaki Vazquez
Thomas Waechter
Andreas Wagner
William Waites
Shenghui Wang
Zhe Wang
Jesse Wang
Klara Weiand
Branimir Wetzstein
Gang Wu
Honghan Wu
Markus Zanker
Benjamin Zapilko
Viesturs Zarins
Yuqing Zhai
Xiaowang Zhang
Xiao Zhang
Antoine Zimmermann

## Sponsoring Institutions











IOS
Press


KNOWLEDGE MEDIA
KMi
INSTITUTE


LarKC
Large
Knowledge
Collider


NoTube


ONTORULE
Ontologies meet Business Rules


ontotext
Semantic Technology Lab


PASCAL2
Pattern Analysis, Statistical Modelling and
Computational Learning


Pathfinder
www.pathfinder.gr


PHAISTOS
networks


seekda!


SOA
Semantics
WWW
Web Services
Internet
All


semsphere


STI · INNSBRUCK


STI · INTERNATIONAL


talis


YAHOO!
RESEARCH

# Table of Contents – Part II

## Services and Software Track



## Social Web Track

## Web of Data Track

## Demo and Poster Track

## PhD Symposium

# Table of Contents – Part I

## Mobility Track



## Ontologies and Reasoning Track

## Semantic Web in Use Track

## Sensor Networks Track

# A Model of User Preferences for Semantic Services Discovery and Ranking

José María García, David Ruiz, and Antonio Ruiz-Cortés

University of Seville
Escuela Técnica Superior de Ingeniería Informática
Av. Reina Mercedes s/n, 41012 Sevilla, Spain
josemgarcia@us.es

**Abstract.** Current proposals on Semantic Web Services discovery and ranking are based on user preferences descriptions that often come with insufficient expressiveness, consequently making more difficult or even preventing the description of complex user desires. There is a lack of a general and comprehensive preference model, so discovery and ranking proposals have to provide *ad hoc* preference descriptions whose expressiveness depends on the facilities provided by the corresponding technique, resulting in user preferences that are tightly coupled with the underlying formalism being used by each concrete solution. In order to overcome these problems, in this paper an abstract and sufficiently expressive model for defining preferences is presented, so that they may be described in an intuitively and user-friendly manner. The proposed model is based on a well-known query preference model from database systems, which provides highly expressive constructors to describe and compose user preferences semantically. Furthermore, the presented proposal is independent from the concrete discovery and ranking engines selected, and may be used to extend current Semantic Web Service frameworks, such as WSMO, SAWSDL, or OWL-S. In this paper, the presented model is also validated against a complex discovery and ranking scenario, and a concrete implementation of the model in WSMO is outlined.

**Keywords:** User Preferences, Ontology Modeling, Semantic Web services, Service Discovery, Service Ranking.

## 1 Introduction

Semantic Web Services (SWS) definition frameworks provide comprehensive tools to describe services and their interactions. However, preferences cannot be described at the same detail level, *i.e.* users cannot define complex desires for a concrete service request. For instance, WSMO goals [19] only support the description of requirements about a request in the form of capabilities, but preferences to rank services fulfilling these requirements cannot be directly expressed by using a standard WSMO goal definition, which only provides means to define non-functional properties / values pairs. In other words, preferences are not considered first-class citizens in WSMO, in comparison to service capabilities, whose

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 1–14, 2010.

definitions are more expressive. Other frameworks, such as OWL-S [16] or SAWSDL [5], do not even define a specific model to describe user requests at all.

Discovery and ranking proposals try to fill this gap, extending SWS frameworks to support preferences definition, or just providing separate user preferences descriptions. There is a variety of formalisms proposed to define preferences, such as tendencies [4,22,24], relative weights [15,17,18,23], or utility functions [8,13,25]. These formalisms actually determine the level of expressiveness of each proposal, while resulting in a high coupling between user preferences definition and its corresponding discovery and ranking implementations.

In order to overcome these limitations, a highly expressive, intuitive model of user preferences is presented in the following. This proposal adapts a well-known model from database systems [12] that allows to define preferences constructively and user-friendly. In this paper, a user preference ontology is described and validated using a discovery scenario from the SWS Challenge[1]. Additionally, the proposed ontological model is applied to WSMO definitions, extending its *goal* element in order to allow the specification of preferences using our model.

The rest of the paper is structured as follows: Sec. 2 discussed the related work on preference definition. Then, in Sec. 3 our proposal is thoroughly presented and validated against a scenario from the SWS Challenge, along with an implementation in WSMO. Finally, Sec. 4 sums up our work and presents the benefits of our contribution.

## 2 Related Work

Concerning the representation of preferences, there are several approaches in the literature based on different formalisms. Thus, preferences modeled as utility functions have been widely used in economics [6,10] and web systems [1,8,25]. Another formalism based on partial orders were proposed in database systems field [3,12]. The main difference between these two formalisms is that the former constitutes a quantitative approach while the latter is qualitative.

Although quantitative approaches are more general because most preference relations can be defined in terms of utility functions, there are some intuitive preferences that cannot be captured by these functions [3]. On the other hand, qualitative approaches have higher expressiveness and are more intuitive and user-friendly, though they are not directly applicable to a SWS scenario because they do not take into account that properties may be expressed using different abstraction levels depending on the stakeholder. Our proposal constitutes a hybrid model, where both qualitative and quantitative preferences can be expressed.

Preference queries from database systems can be also applied to the Semantic Web scenario with ease. Thus, Siberski *et al.* [20] extend SPARQL query language with a preference clause similar to the proposed by Kießling [12], which in turn serves as the foundations for the preference model presented in Sec. 3.1. Applied to SWS discovery and ranking scenario, there are some approaches in the literature [9,14], though their preference model is essentially based on utility functions.

[1] http://sws-challenge.org

Additionally, their model is coupled with their actual implementation of discovery and ranking, consequently limiting their expressiveness. A more comprehensive comparison and overview of these approaches can be found in [7].

Other proposals to model preferences on SWS are more focused on ranking mechanisms, so their preference model are specifically tailored towards their implementations. Toma *et al.* [22] presents a multi-criteria approach based on logic rules, modeling preferences by including simple annotations to WSMO goals, in a similar fashion as policies defined in Palmonari *et al.* [17], though they provide more facilities to express relative weights and different offered policies. García *et al.* [8] provide means to semantically define preferences as utility functions, integrating both logic rules and constraint programming to rank services with respect to those preferences. Finally, a hybrid approach to SWS ranking is proposed by Kerrigan [11], where preferences are described using instances from an ontology, as in our proposal, while distinguishing filtering and ordering preferences that are used at different stages of his solution.

## 3 Defining an Ontology of User Preferences

In the following, an ontology representing our proposed preference model is presented in detail. In order to validate this model, an application to the Logistics Management scenario from the SWS Challenge[2] is also discussed. Finally, we depict how to extend WSMO framework so that users can define their preferences inside WSMO goals.

### 3.1 User Preferences Model

As discussed before, service descriptions and user preferences should be semantically described at the same detail level. Therefore, there is a need for the definition of an ontological model that leverages preference descriptions as first-class citizens in the discovery and ranking scenario. This model has to provide intuitive and user-friendly facilities to easily define both requirements and preferences, so that service descriptions can be matched with user requests. Furthermore, these facilities have to conform a sufficiently expressive model so that a user can describe any kind of preference, without being limited by a concrete formalism or representation.

In order to specify a preference model, firstly we need to establish a clear separation between requirements that have to be met, and preferences that have to be taken into account once requirements have been fulfilled. Typically, requirements are hard constraints that are used to filter service repositories in the discovery process, while preferences are used to rank previously discovered services so that the most appropriate service can be selected after the ranking process. Therefore, preferences define a strict partial order in our model, providing a framework to compare and rank a set of services.

[2] The complete scenario description can be found at `http://sws-challenge.org/wiki/index.php/Scenario:_Logistics_Management`

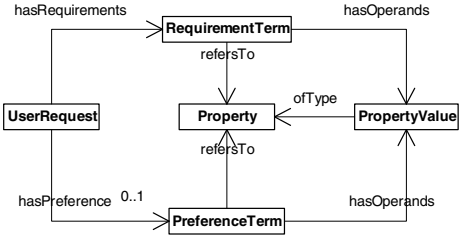



**Fig. 1.** Upper ontology of user preferences

Figure 1 shows the upper ontology of our model, which is represented using a UML-based notation for OWL ontologies [2] that is also used throughout the rest of the paper. The root concept in our proposed model is called `UserRequest`, which consists on the materialization of user desires with respect to a particular service request. These desires are described using `RequirementTerm`s and possibly a `PreferenceTerm`. Thus, hard requirements are defined by instances of subclasses of `RequirementTerm`, and the same applies to `PreferenceTerm`. Both requirements and preferences are related with one or more properties, which are referred inside each term, and with some property values that act as operands for each term specialization. Properties and property values depend on the specific domain being used within each scenario, so instances (or specializations) of Property and PropertyValue classes may be probably described using an external domain ontology.

Requirements definition has been widely discussed in the literature, and SWS frameworks provide sufficiently expressive facilities to define them, so in the following we will focus on preference modeling. In Sec. 3.2, requirement terms are simply considered as property / property value pairs, which is sufficient to describe user requests in the validation scenario.

Concerning preferences, Fig. 2 presents the middle ontology of our model, where a hierarchy of available preference constructors is introduced. Thus, a preference term can be an `AtomicPreference`, or a composition of two preference terms by applying one of the sub-classes of `CompositePreference`. On the one hand, atomic preferences are those which refers to a single property, and can describe either a qualitative or a quantitative preference that users may have with respect to the referred property. On the other hand, composite preferences relate different preferences between them, so that a complex preference can be described. These complex constructors are defined for two preferences in our model, though they can be generalized to a greater number of preferences obviously.

Each preference construct from Fig. 2 is defined intuitively in the following, including a motivating example described in natural language from the SWS Challenge scenario used to validate our proposed model in Sec. 3.2, where some of these constructs are applied to describe that scenario goals. A formal definition of the described preference terms can be found at [12], where the foundations of our model are thoroughly discussed.

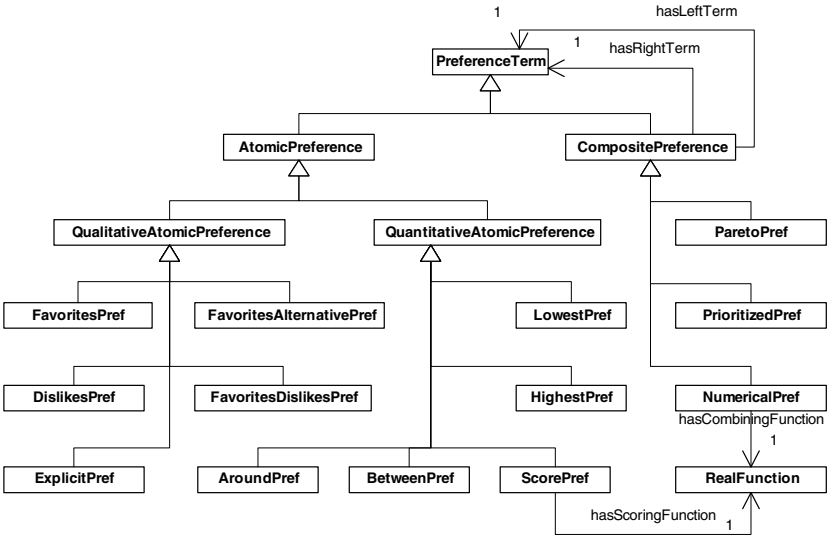



**Fig. 2.** Preference terms hierarchy

**Qualitative atomic preferences.** The first group of preferences presented in the following corresponds to the qualitative and atomic constructs, which means that every preference that belongs to this kind refers to a single, non-numerical property.

**a) Favorites preference -** `FavoritesPref`
A *favorites preference* defines a finite set of property values that constitute the desired values of the referred service property. Thus, services whose value for that property is a member of the *favorite set* are preferred to services that provide any other values from the property domain. An instance of this preference constructor has many operands as the cardinality of the favorite values set.
*Example:* I prefer WSs that provide *carriageForward* as a possible `PaymentMethod`[3].

**b) Dislikes preference -** `DislikesPref`
As opposite to `FavoritesPref`, a *dislikes preference* define a set of property values that the service should not provide for the referred property in order to be preferred to another service whose property values coincide with any of the values in the associated *dislikes set*.
*Example:* I prefer WSs that do not offer *refundForDamage* as an available `Insurance` option.

[3] In each example, *italics* text correspond to property values used as operands, while `typewriter` text are used to denote properties.

**c) Favorites or alternative preference -** `FavoritesAlternativePref`
A *favorites or alternative preference* is an extension of `FavoritesPref`, where there are two favorite sets. The second set is called *alternative set*. In this case, services whose property values are in the favorite set are the most preferred. Otherwise their values should be on the alternative set. If this is not the case either, then the corresponding services will be undesirable, because their property values are not member of any of the two sets. In order to differentiate between favorite and alternative set, `hasOperands` object property has to be specialized to two different object properties, namely `hasFavorite` and `hasAlternative`.
*Example:* I prefer WSs whose `PaymentMethod` is *carriagePaid*, but if that is infeasible, then it should be *carriageForward*.

**d) Favorites or dislikes preference -** `FavoritesDislikesPref`
It is also possible to combine `FavoritesPref` with `DislikesPref` in the following form: service properties should have a value on the defined favorite set. Otherwise, values should not be from the dislikes set. If none of these two conditions hold, then the service will be less preferred than others fulfilling the first or the second condition. Again, `hasOperands` is specialized for this preference constructor to `hasFavorite` and `hasDislikes` object properties.
*Example:* I prefer WSs that provide *refundForLoss* as a possible `Insurance`, but if that is infeasible, then it should not be *refundForDamage*.

**e) Explicit preference -** `ExplicitPref`
An *explicit preference* can be used to explicitly represent the strict partial order between a pair of property values. Thus, a better-than graph can be defined using several *explicit preferences*. In this case, `hasOperands` is specialized to `hasLeftOperand` and `hasRightOperand`, meaning that the left operand value is better than the right operand value, with respect to the referred property.
*Example:* WSs that provide *carriageForward* as a possible `PaymentMethod` are more preferred than those that provide *carriagePaid*.

**Quantitative atomic preferences.** When the referred property of an atomic preference is numerical, the following quantitative constructs may be used to express user preferences on that single property.

**a) Around preference -** `AroundPref`
An *around preference* determines which property value is better by determining the distance of each values to a concrete value provided as an operand of this preference term. Thus, services which provide exactly that value are preferred to the rest of them. If this is infeasible, services with closer values to the operand are preferred.
*Example:* I prefer WSs that provide a `BasePrice` closer to `180 Euros`.

**b) Between preference -** `BetweenPref`
In this case, a service should have values for the referred property between a range of values that are defined as operands in the preference (`hasOperands` is specialized to `lowBound` and `upBound`). If this is not the case, *between*

*preferences* prefer services closer to the interval boundaries, computing the distance as in around preferences.
*Example:* I prefer WSs that provide a `PaymentDeadline` within the interval of $[45, 60]$ days.

**c) Lowest preference - `LowestPref`**
A *lowest preference* does not have any operand, but prefer services whose property values are as low as possible for the referred service property.
*Example:* I prefer WSs that provide a `BasePrice` as low as possible.

**d) Highest preference - `HighestPref`**
In opposition to the last constructor, a *highest preference* is used when property values should be as high as possible.
*Example:* I prefer WSs that provide a `PaymentDeadline` as long as possible.

**e) Score preference - `ScorePref`**
A *score preference* basically defines a scoring function (*i.e.* a utility function like in [8]) that takes a property value as its argument and returns a real value that can be interpreted in the following form: the higher the value returned by the function is, the more preferred the property value entered as the argument. Note that this kind of preference is not as intuitive as the rest, but it is still useful when a user wants to express complex grades of preference, using for instance a piecewise function depending on the property values.
*Example:* I prefer WSs with the highest score with respect to `Price/Kg`, where the scoring function is defined as: $f(pricePerKg) = \frac{-1}{50} pricePerKg + 1$.

**Composite preferences.** The last group of preference constructs are used to compose two different preference terms by stating the preference relationship between each component term, which can be also a composite preference. Composite preferences `refersTo` property associate the preference with the union of the properties referred by component preferences.

**a) Pareto preference - `ParetoPref`**
A *pareto preference* P combines two preference terms P1 and P2 using the *Pareto-optimality principle*, which considers that P1 and P2 are equally important preferences. Thus, a service $SWS_1$ is better than another service $SWS_2$ with respect to P, if $SWS_1$ is better than $SWS_2$ with respect to P1 and $SWS_1$ is not worse than $SWS_2$ with respect to P2, and *vice versa*. Note that P1 is linked with P using the `hasLeftTerm` object property, and P2 using the `hasRightTerm` object property. Intutively, this preference balance the fulfillment of each preference component, so that the composite preference is the average degree of preference taking both components into account.
*Example: Cf.* Fig. 6 for a use case of a pareto preference.

**b) Prioritized preference - `PrioritizedPref`**
In the case of a *prioritized preference* P that compose two preference terms P1 and P2, P1 is considered more important than P2. Thus, P2 is evaluated only if P1 does not mind (*i.e.* service property values compared using P1 do not return enough information to rank those services).
*Example: Cf.* Fig. 4 for a use case of a prioritized preference.

**c) Numerical preference -** `NumericalPref`
Finally, a *numerical preference* is the combination of two score preferences using a function that takes the values returned by the score preferences as its arguments and returns another real number that gives information about the global preference, considering all the properties referred by concrete score preferences. Notice that component preferences must be score preferences in order to properly compose them using a combining function.
*Example:* Provided that $f(basePrice)$ and $g(pricePerKg)$ are already defined and they range within the interval $[0, 1]$, I prefer WSs that have a higher combined score, where the combining function is defined as:
$combF(basePrice, pricePerKg) = 0.8 * f(basePrice) + 0.4 * g(pricePerKg)$.

### 3.2 Validating the Model

The proposed preference model has been validated using one of the discovery scenarios from SWS Challenge. Concretely, the chosen one has been the Logistics Management scenario, because of its complexity and the inclusion of preference descriptions. It consists on seven logistics service offers, described in natural language in terms of different properties, such as price, covered geographical areas, operating hours and truck fleets, among others. Additionally, several service requests (*i.e.* user goals) applicable to this scenario are defined, which contain both hard requirements and user preferences (they are referred as *soft constraints* in the scenario) that can be used to choose the most appropriate service (*i.e.* the best one in terms of preferences) among those which fulfill hard requirements.

In the chosen scenario, goals B1, C1, D1 and E1 define a variety of preferences against different service properties, in addition to describe how preferences should be combined within each goal. In order to validate our preference model using this scenario, we provide in the following equivalent descriptions for each of these goals using the proposed model. Thus, textual descriptions of goals directly extracted from the scenario description are shown alongside their equivalent representation using the preference ontology presented in Sec. 3.1. For the sake of simplicity, service properties are included as instances inside the same default namespace as the goal, though a domain ontology should be externally defined, covering all the existing properties in the Logistics Management domain.

Figure 3 presents the instantiation of the goal B1 from the scenario. The goal as a whole is modeled with an instance of `UserRequest`, while each term is instantiated depending on its nature. Thus, requirements about pickup, delivery and transported goods are represented at the top of the figure. This representation is shown simplified, because requirements in every goal from the Logistics Management scenario are pairs between domain properties and their values. Consequently, in the following instantiated goals, requirements are omitted from the representation, though they can be easily described using property-value pairs.

Concerning the preference modeling, goal B1 states that the user prefers two properties, namely `PaymentMethod` and `Insurance`, to contain certain values,

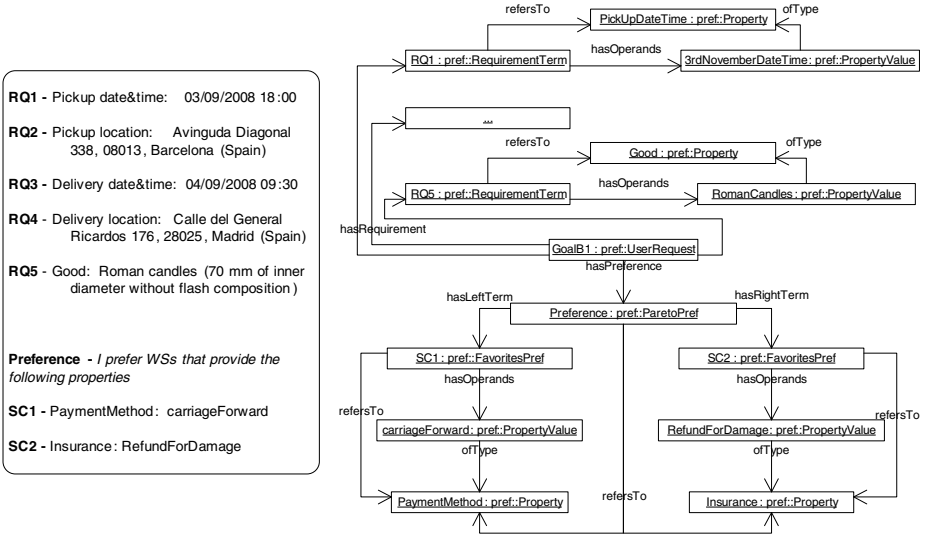



**Fig. 3.** Goal B1 description excerpt and its representation using our model

`carriageForward` and `refundForDamage`, respectively. Both of these soft constraints are considered qualitative preferences that define the favorite values for each property. However, the preference description do not explicitly express how to compose those two atomic preferences, so it can be inferred that a *pareto preference* can be applied to relate each one, because both atomic preferences can be considered equally important for the user.

The next goal used to validate our model is shown in Fig. 4. In this case, the atomic preferences defined in C1 are instantiated as a *favorites preference* and an *around preference*. Moreover, the preference description gives more importance to the favorites preference on the `PaymentMethod` property, taking

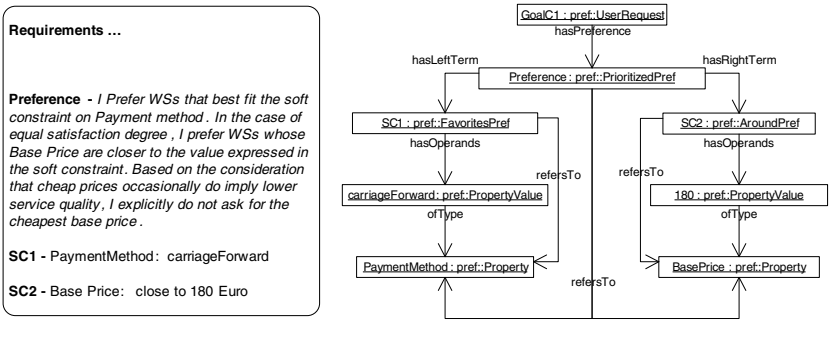



**Fig. 4.** Goal C1 description excerpt and its representation using our model

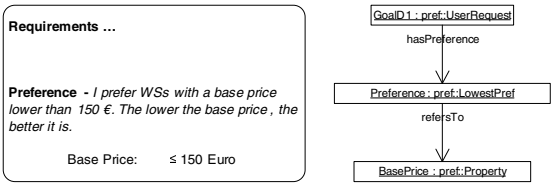



**Fig. 5.** Goal D1 description excerpt and its representation using our model

into consideration the preference about the `BasePrice` only if services have an equal satisfaction degree when evaluating the first preference. Thus, the most appropriate composite preference is a *prioritized preference*, because its semantics are exactly what the user is looking for in this goal.

Goal D1, which is represented in Fig. 5, is the most simple goal of the scenario. There is no composition of atomic preferences, because it only states that the `BasePrice` should be as low as possible. The limit for the price that is included in the scenario is not necessary in our solution, because the semantics of the *lowest preference* is sufficient in this case to properly rank services with respect to the stated user preferences. Nevertheless, it is possible to take that price limit into account by modeling the user preference as a *prioritized preference*, where P1 is a *between preference* on `BasePrice` with the interval $[0, 150]$, and P2 is the *lowest preference* shown in Fig. 5.

Finally, the most complex goal of the scenario is shown in Fig. 6, where some of the `refersTo` relations are omitted for the sake of clarity[4]. The different atomic preferences are composed using *pareto preferences*, because the goal E1 description explicitly states that the user wants an average satisfaction degree among the atomic preferences. Notice that SC3 is decomposed into two *favorites preferences*, because it was interpreted that `Insurance` property should have both values. If SC3 were modeled using only one *favorites preference* with the two values in the favorite set, then services that supports only one type of insurance would be considered equally preferred than those supporting both insurance values.

In conclusion, the presented validation using a relatively complex discovery and ranking scenario from the SWS Challenge proves that our proposed model is sufficiently expressive and intuitive, allowing to describe any kind of user preferences directly, user-friendly, and independently of the discovery and ranking technique to apply at a later stage. Additionally, the actual evaluation of the described preferences lead to the expected ranking results that are described in the scenario. This evaluation can be performed applying formal definitions of the equivalent preference constructs from [12]. Further validation may be performed using other scenarios and test cases, such as the shipment discovery scenario used in [8].

[4] Actually, this relation can be inferred from the type of the operands involved in each preference.

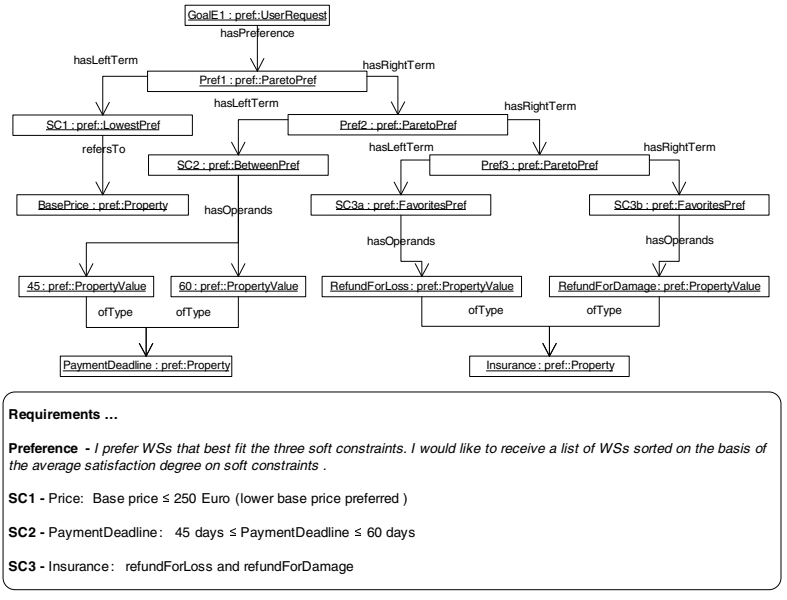



**Fig. 6.** Goal E1 description excerpt and its representation using our model

### 3.3 Implementing the Model in WSMO

Due to the fact that the proposed preference model is general and independent from the formalism, it can be applied as an extension to current SWS frameworks, such as WSMO, OWL-S, or even SAWSDL, so that these frameworks can support user preference modeling. Concerning WSMO, our proposed model can be implemented as an extension of its meta-model. Thus, user requests from our model corresponds to WSMO *goals*. Moreover, requirement terms are already supported by WSMO capabilities and interfaces, so that requirement terms described in our model can be easily translated into capabilities. However, preference terms have to be added to the specification of goals. Therefore, in order to apply our preference model to WSMO, we define a new meta-model class in Listing 1, `preferenceGoal`, which is a subclass of `goal` that adds a `hasPreference` property where preference terms can be linked with a user goal.

**Listing 1.** WSMO goal extended with preferences

```
Class preferenceGoal sub−Class goal
    hasNonFunctionalProperties type nonFunctionalProperties
    importsOntology type ontology
    usesMediator type {ooMediator, ggMediator}
    hasPreference type PreferenceTerm
        multiplicity = single−valued
    requestsCapability type capability
        multiplicity = single−valued
    requestsInterface type interface
```

This implementation allows a seamless integration of preference information in WSMO, without actually modifying the goal meta-model, because it is only refined. Thus, current WSMO discovery and ranking proposals could be still applied to extended goals transparently. A different approach can be found in [8], where preferences are included within `nonFunctionalProperties` section by using logic programming rules, although it is only applied to preferences defined as utility functions.

Listing 2 shows an example of how to describe a WSMO goal using our proposed implementation to include our preference model. Thus, goal D1 from Sec. 3.2 can be described in WSMO easily. The domain ontology for the Logistics Management scenario is supposed to be properly defined in `Logistics.wsml`.

**Listing 2.** Extended goal description with preferences from D1

```
namespace {_"GoalD1.wsml#", lm _"Logistics.wsml#",
  wsml _"http://www.wsmo.org/wsml/wsml-syntax/",
  pref _"http://www.isa.us.es/ontologies/PreferenceModel.wsml#"}

preferenceGoal GoalD1

 capability D1requestedCapability
 preference D1preference

ontology preferenceOntology

instance D1preference memberOf pref#LowestPreference
  pref#refersTo hasValue lm#BasePrice
```

From the above example, one concludes that transforming user requests modeled using our proposed ontology for preferences to a WSMO goal is a straightforward process, provided that the ontological model is expressed in WSML [21]. Also notice that the WSML variant used in Listing 2 includes new keywords to link specialized goals to preference terms which are described in a separate ontology.

## 4 Conclusions

In this work, a highly expressive preference model for SWS discovery and ranking is presented. This model, specified as an ontology, represents a novel approach that leverages preference descriptions so that they become a first-class citizen in SWS frameworks. Furthermore, the model has been validated using a complex discovery scenario from the SWS Challenge in order to prove the applicability of our solution to an actual discovery and ranking scenario. The main benefits of our proposed model can be summarized as follows:

- **Expressiveness.** The model is sufficiently expressive to describe complex user desires about requested services, providing a comprehensive hierarchy of preference terms.
- **Intuitive semantics.** Based on a strict partial order interpretation of preferences, the model is both user-friendly and machine-readable, so preferences may be automatically processed and inferred.

- **Qualitative and Quantitative.** Available constructs allow to express both qualitative and quantitative preferences, and even combine them in a general preference term.
- **Independence.** Our proposal is not coupled with a concrete SWS solution, neither with a discovery nor ranking mechanism, so it is not limited by the formalisms used to implement these mechanisms.
- **Extensibility.** Because the model is presented as an ontology, it can be further extended with new preference constructs with ease.
- **Applicability.** Our model can be implemented within any SWS framework, extending current proposals to leverage preference descriptions.

An implementation of our model that extends WSMO goals is also discussed. This actual application consists in a seamless extension of WSMO constructs to allow the definition of complex preferences, that can be used within any discovery and ranking solution, provided that it supports or adapts the proposed preference ontological model.

**Acknowledgments.** This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT projects Web-Factories (TIN2006-00472) and SETI (TIN2009-07366), and by the Andalusian Government under project ISABEL (TIC-2533).

## References

1. Agrawal, R., Wimmers, E.L.: A framework for expressing and combining preferences. ACM SIGMOD 29(2), 297–306 (2000)
2. Brockmans, S., Volz, R., Eberhart, A., Loffler, P.: Visual modeling of OWL DL ontologies using UML. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 198–213. Springer, Heidelberg (2004)
3. Chomicki, J.: Preference formulas in relational queries. ACM Transactions on Database Systems (TODS) 28(4), 466 (2003)
4. Dobson, G., Lock, R., Sommerville, I.: QoSOnt: a QoS ontology for service-centric systems. In: EUROMICRO-SEAA, pp. 80–87. IEEE Comp. Soc., Los Alamitos (2005)
5. Farrell, J., Lausen, H.: Semantic annotations for WSDL and XML Schema. Technical report, World Wide Web Consortium (August 2007)
6. Fishburn, P.C.: Utility theory for decision making. Wiley, Chichester (1970)
7. García, J.M., Rivero, C., Ruiz, D., Ruiz-Cortés, A.: On using semantic web query languages for semantic web services provisioning. In: The 2009 International Conference on Semantic Web and Web Services (SWWS). CSREA Press (2009)
8. García, J.M., Toma, I., Ruiz, D., Ruiz-Cortés, A.: A service ranker based on logic rules evaluation and constraint programming. In: 2nd ECOWS Non-Functional Properties and Service Level Agreements in Service Oriented Computing Workshop, CEUR Workshop Proceedings, Dublin, Ireland, November 2008, vol. 411 (2008)
9. Iqbal, K., Sbodio, M.L., Peristeras, V., Giuliani, G.: Semantic service discovery using SAWSDL and SPARQL. In: Fourth International Conference on Semantics, Knowledge and Grid, SKG 2008, pp. 205–212 (2008)

10. Keeney, R.L., Raiffa, H.: Decisions with multiple objectives: Preferences and value tradeoffs. Cambridge Univ. Press, Cambridge (1993)
11. Kerrigan, M.: Web service selection mechanisms in the web service execution environment (WSMX). In: SAC, pp. 1664–1668 (2006)
12. Kießling, W.: Foundations of preferences in database systems. In: VLDB, pp. 311–322. Morgan Kaufmann, San Francisco (2002)
13. Kritikos, K., Plexousakis, D.: Semantic QoS metric matching. In: ECOWS 2006, pp. 265–274. IEEE Computer Society Press, Los Alamitos (2006)
14. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: WWW 2007, pp. 1013–1022. ACM, New York (2007)
15. Liu, Y., Ngu, A.H.H., Zeng, L.: Qos computation and policing in dynamic web service selection. In: WWW (Alt Track Papers & Posters), pp. 66–73 (2004)
16. Martin, D., Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., et al.: OWL-S: Semantic markup for web services. Technical Report 1.1, DAML (2004)
17. Palmonari, M., Comerio, M., De Paoli, F.: Effective and Flexible NFP-Based Ranking of Web Services. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 546–560. Springer, Heidelberg (2009)
18. Pathak, J., Koul, N., Caragea, D., Honavar, V.G.: A framework for semantic web services discovery. In: WIDM 2005: Proceedings of the 7th annual ACM international workshop on Web information and data management, pp. 45–50. ACM Press, New York (2005)
19. Roman, D., Lausen, H., Keller, U.: Web service modeling ontology (WSMO). Technical Report D2 v1.3 Final Draft, WSMO (2006)
20. Siberski, W., Pan, J.Z., Thaden, U.: Querying the Semantic Web with Preferences. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 612–624. Springer, Heidelberg (2006)
21. Steinmetz, N., Toma, I. (eds.): The Web Service Modeling Language WSML. Technical report, WSML, WSML Working Draft D16.1v0.3 (2008)
22. Toma, I., Roman, D., Fensel, D., Sapkota, B., Gomez, J.M.: A multi-criteria service ranking approach based on non-functional properties rules evaluation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 435–441. Springer, Heidelberg (2007)
23. Vu, L.H., Hauswirth, M., Porto, F., Aberer, K.: A search engine for QoS-enabled discovery of semantic web services. International Journal of Business Process Integration and Management 1(4), 244–255 (2006)
24. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A QoS-aware selection model for semantic web services. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 390–401. Springer, Heidelberg (2006)
25. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)

# Towards Practical Semantic Web Service Discovery

Martin Junghans, Sudhir Agarwal, and Rudi Studer

Karlsruhe Institute of Technology (KIT)
Institute of Applied Informatics and Formal Description Methods (AIFB)
Karlsruhe Service Research Institute (KSRI)
Karlsruhe, Germany
{martin.junghans,sudhir.agarwal,rudi.studer}@kit.edu

**Abstract.** Service orientation is a promising paradigm for offering and consuming functionalities within and across organizations. Ever increasing acceptance of service oriented architectures in combination with the acceptance of the Web as a platform for carrying out electronic business triggers a need for automated methods to find appropriate Web services.

Various formalisms for discovery of semantically described services with varying expressivity and complexity have been proposed in the past. However, they are difficult to use since they apply the same formalisms to service descriptions and requests. Furthermore, an intersection-based matchmaking is insufficient to ensure applicability of Web services for a given request. In this paper we show that, although most of prior approaches provide a formal semantics, their pragmatics to describe requests is improper since it differs from the user intention. We introduce distinct formalisms to describe functionalities and service requests. We also provide the formal underpinning and implementation of a matching algorithm.

## 1 Introduction

Service-oriented computing is an interdisciplinary paradigm that revolutionizes the very fabric of distributed software development Applications that adopt service-oriented architectures (SOA) can evolve during their lifespan and adapt to changing or unpredictable environments more easily. When properly implemented, services can be discovered and invoked dynamically, while each service can still be implemented in a black-box manner. Despite these promises, service integrators, developers, and providers need to create techniques and tools to support cost-effective development, as well as the use of dependable services and service-oriented applications.

**Brief Overview.** Service discovery deals with finding appropriate Web services for the task at hand and is one of the central components needed for developing a SOA application. Universal Description, Discovery and Integration (UDDI) based service discovery is rather syntactical and requires a lot of manual effort

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 15–29, 2010.

for finding the right services [1]. For example, UDDI is not able to deal with synonyms or relations between terms that describe services. Since the advent of the Semantic Web, many semantic Web service discovery approaches have been proposed to deal with heterogeneity in the terminology used in different services. Some of them consider the functionality description of services, which allows for automated tasks like service composition. The common model to describe the functionality of a service is represented by inputs, outputs, preconditions, and effects, or shortly denoted by $(I, O, \phi, \psi)$. Inputs denote the set of user-provided message parts at Web service invocation. Outputs describe the set of values returned to the user after service execution. Preconditions and effects describe the information states of the world before and after service execution, resp., by logical formulas. Semantic Web service discovery approaches compute the match between a service offer that describes the functionality of the service and a service request.

OWL-S Matchmaker uses OWL-S profile for describing Web service offers as well as requests [2,3]. Even though OWL-S Profile has elements for preconditions and effects, the OWL-S matchmaker uses types of input and output parameters only. The approach presented in [4] models Web services as well as requests as description logic (DL) classes and bases the matchmaking on the intersection of service offer and request, which is computed by a DL reasoner. Such approaches fail to reason about the dynamics of Web services, since DL reasoners can not reason about changing knowledge bases. The approach in [5] deals with variables, but is limited to Web services that do not change the world and, thus, can be described by a query. Efficient semantic discovery approaches that can deal with functionality of Web services are presented in [6,7]. Efficiency is achieved by pre-computing a classification of services in a hierarchy of goal templates. However, the requirement of such a classification hierarchy hinders the usability of creating service descriptions and requests since it is not feasible to maintain a global hierarchy in a decentralized and open environment of the Web. Furthermore, [6,7] do not support matching of inputs and outputs nor do they deal with the possible inconsistency between functional description of Web services and their classification.

**Problem of Using One Formalism.** Apart from the problems mentioned above, one common problem of all existing approaches is that they apply the same formalism for describing service offers and requests. The use of the same formalism for both descriptions does not correspond with the intuition of the requester. Such mismatch between the semantics of formalisms and the intuitive interpretation of the requester makes these approaches hard to use in practice.

If the same formalism is used for offers and requests, then a service request corresponds to a service offer description from which a set of desired services is derived. In our view, this is an impractical and unintuitive approach as we will further justify in the subsequent section. Consequently, we propose to use two distinct formalisms for service descriptions and request. It is more intuitively that a service description formalizes the actual functionality of a Web service and a service request describes the set of services that provide a requested functionality.

**Structure.** In this paper we provide a semantic service discovery approach to overcome the above outlined problems. In Section 2, we elaborate on the problems of state of the art approaches and the motivation for our approach. Two different formalisms to describe services offers and requests are introduced in Sections 3 and 4, resp. Based on the semantics of the formalisms that we provide, the definition of a match between offers and requests is also presented in Section 4.4. We further present an implementation of our approach accompanied with performance tests in Section 5. In Section 6, we discuss the relation of our discovery approach to other approaches. Finally, we conclude in Section 7 by summarizing our results and giving an outlook.

## 2 Motivation

At first sight, a practical semantic service discovery should feature expressive description languages that are non-ambiguous, easy to use as well as support heterogeneous descriptions. We do not investigate usability aspects by means of a simple syntax, query language for expressing formulas or supportive user interfaces. To this extent, our approach distinguishes from goal driven approaches as in [8] by not focusing on an abstract and non-technical request (goal) description that a user creates. Such user goals have to be translated into machine understandable requests. The latter representation of a request is the starting point of our work.

The formalisms to describe Web service functionalities and requests must provide sufficient expressivity to allow users to formulate rich functionality descriptions and to precisely formulate constraining requirements in requests, resp. Users should be rather limited by their willingness to invest effort than by any technical limitation.

Although using a formal way to express a request, usability can be fostered by the provision of an unambiguous and thus comprehensible interpretation of service descriptions and requests. E.g., it should be obvious to users whether the requested inputs are interpreted as inputs that must or can be accepted by desired services.

Regarding the openness of the Web, semantic service descriptions are considered to be highly heterogeneous as different actors may use different vocabularies and ontologies. Also, a discovery solution must scale against a large number of available semantic Web service descriptions.

**Requirements on Formalisms.** A service request describes a class of services, namely the desired ones. If the description of a service request uses the same formalism as the one used for service descriptions, then there exists a mismatch between the interpretation of a service description $D = (I, O, \phi, \psi)$ and a request description $R = (I_R, O_R, \phi_R, \psi_R)$. As depicted in the left part of Figure 1, $D$ and $R$ are both interpreted as a set of execution runs. A match between them is given if there is an intersection between the two sets of runs. Degrees of matches, for instance plugin and subsume match, present different types of the intersection of both sets of runs. We refer to [3] for details on the degrees of matches.

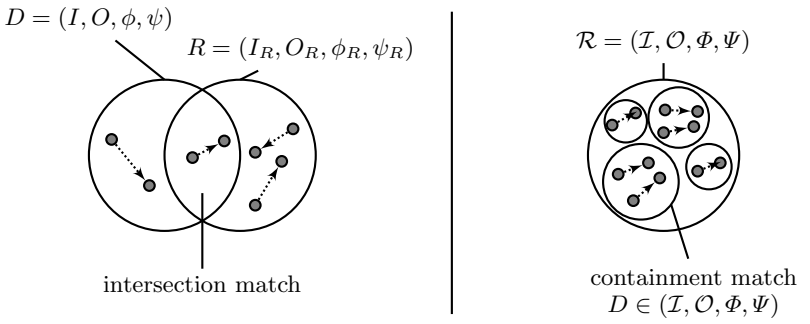



**Fig. 1.** On the left, the same formalism is applied to Web service description $D$ and request $R$. Using our different formalisms for offers and requests allows for the description of several run sets in a request $\mathcal{R}$ as shown in the right part.

Intersection-based approaches lack the ability to let requests exclude certain properties because not all requested properties need to be fulfilled by matching services. Furthermore, an intersection-based match cannot guarantee that a matching service can be successfully invoked as the execution run that the user wants to invoke may not comply to the requirements specified in the request. This can be since the desired run might not be member of the intersection between service offer and request. Thus, in order to guarantee applicability of a matching service, intersection-based approaches using the same formalism for offers and requests need to further check for applicability in a further step. Consequently, the freedom provided by the different matching degrees is not practicable for the purpose of service invocation. As an example that was already discussed in [9], a service offers to ship goods from a city in the UK to another city in Germany. A user requests for a shipping provider that operates between European cities. Using intersection based matchmaking will identify the mentioned service as a match. However, if the requester wants to ship an item from Berlin to Hamburg, then the matched service offer fails.

Another example explains why the service description formalism cannot be used for requests and vice versa. Consider the functional description of a book selling Web service that requires the invoker to provide an ISBN book number of the book to order as input. While inputs of the service descriptions are consentaneously interpreted as required for invocation of an atomic Web service, the inputs specified in a request can be interpreted differently. From the user perspective, a request for book selling Web services may contain different interpretations of inputs simultaneously. Either the user provides an ISBN number or alternatively author name and book title as inputs. This simple example leads to the observation that a request cannot simply specify a set of inputs. The same conceptual mismatch between service descriptions and requests occurs in preconditions and effects. Employing the same formalism for the description of a service and a set of desired services is not appropriate, because their interpretation and their intended use are different. Requests conceptually differ from service

functionality descriptions. We believe that this mismatch makes $(I, O, \phi, \psi)$-based formalisms difficult to use.

In order to develop a formalism that allows to practically describe requests that can be matched against $(I, O, \phi, \psi)$-based service descriptions, we identified the requirements of requests descriptions that also effect the formalism of service descriptions. First, users should be able to describe required properties of a set of desired services in a request. As depicted in the right part of Figure 1 and in analogy to database queries, a request rather describes properties of the result set than a precise desired service functionality. Second, it needs to be clear to requesters how the request description effects the set of matching services. In contrast, it is not obvious which requested properties of a matching service justify the match if discovery approaches consider different degrees of matches. The reason is that intersection-based matchmaker cannot guarantee that matching services comply to all requested properties. And third, to provide a matchmaking algorithm, offers and requests have to be mapped to a common formal model where matches can be identified.

## 3 Service Descriptions

In this section we first introduce our formal model of Web services and then present the formalism to describe service functionality descriptions. The functionality of a Web service is described by a set of inputs $I$, set of outputs $O$, precondition $\phi$, and effect $\psi$. We consider atomic Web services that may require user inputs at service invocation time and provide outputs at the end solely. There are no user interactions in between, which allows us to describe service functionalities by the states before and after execution without stating anything about the intermediate states.

### 3.1 Formal Model of Web Services

In our formal model, we consider a set of actors $\mathcal{A}$ identified by a unique identifier, e.g., their public key. For each actor $A \in \mathcal{A}$, we consider a knowledge base $KB_A$. Furthermore, each actor $A \in \mathcal{A}$ can provide a set of Web services. A Web service can use other Web services (of the same or different actors). That is, an execution of a Web service $W$ provided by an actor $A \in \mathcal{A}$ can cause changes not only in the knowledge base $KB_A$ of actor $A$, but also in the knowledge bases of (external) actors whose Web services are used by the Web service $W$. However, the execution of a Web service $W$ can not cause any changes in the knowledge bases of the actors that are not involved in $W$. We model a state as the set of knowledge bases of all the actors. Formally, a state is $\{KB_A : A \in \mathcal{A}\}$. The execution of a Web service $W$ is equivalent to a transition between states. The transition models changes in the knowledge bases within the resp. states.

### 3.2 Inputs and Outputs

The sets of inputs $I$ and outputs $O$ denote the set of inputs that are compulsory for service invocation and the set of outputs returned after successful execution,

respectively. They assign the service's inputs and outputs to variable names that can be referenced in preconditions and effects and also allow us to distinguish inputs and outputs from instances that already exist in the provider's knowledge bases. We model a book selling Web service as a running example for illustration. The types of inputs and outputs are specified in preconditions and effects. The service with $I = \{\mathsf{u}, \mathsf{p}, \mathsf{a}, \mathsf{t}\}$ requires a user identification $\mathsf{u}$, password $\mathsf{p}$, author name $\mathsf{a}$, and a title $\mathsf{t}$ for successful invocation. Further, $O = \{\mathsf{b}, \mathsf{i}\}$ describes that the service provides a book $\mathsf{b}$ and invoice $\mathsf{i}$ as outputs after execution to the invoker.

### 3.3 Preconditions and Effects

In order to provide a practical discovery approach, we now clearly specify the intention and the interpretation of preconditions and effects. By this we clarify the pragmatics and avoid ambiguities about what is modeled by a provider and in turn which conditions a requester may query in a request. The pragmatics of the logical formula that represents the precondition is restricted to the description of (i) requirements on inputs, like their types, relationships among them, conditions on the values of the inputs, and (ii) conditions that must hold in the resp. state from the perspective of service providers. A precondition describes the state from the perspective of the service provider(s) before a service can be successfully invoked. In contrast to [10], by preconditions and effect we do not intend to describe global states that model the knowledge bases of the entire world as perceived by an external observer which certainly causes several problems.

An effect formula is restricted to describe (i) constraints on returned outputs, (ii) the relation between inputs and outputs, and (iii) changes made by the service in the knowledge bases of service providers.

Below an example precondition $\phi$ and effect $\psi$ description of the book selling service is partly shown.

$$\begin{aligned}\phi \equiv\ & \mathsf{UserId}(\mathsf{u}) \wedge \mathsf{isRegistered}(\mathsf{u}) \wedge \mathsf{isAuthorized}(\mathsf{u}, \mathsf{p}) \wedge \mathsf{Password}(\mathsf{p}) \wedge \mathsf{Book}(\mathsf{b}) \wedge \\ & \mathsf{hasAuthor}(\mathsf{b}, \mathsf{a}) \wedge \mathsf{Author}(\mathsf{a}) \wedge \mathsf{hasTitle}(\mathsf{b}, \mathsf{t}) \wedge \mathsf{Title}(\mathsf{t}) \wedge \mathsf{isAvailable}(\mathsf{b}) \wedge \ldots \\ \psi \equiv\ & \mathsf{Order}(\mathsf{o}) \wedge \mathsf{containsProduct}(\mathsf{o}, \mathsf{b}) \wedge \mathsf{containsPrice}(\mathsf{o}, \mathsf{p}) \wedge \mathsf{hasPrice}(\mathsf{b}, \mathsf{p}) \wedge \\ & \mathsf{Invoice}(\mathsf{i}) \wedge \mathsf{containsProduct}(\mathsf{o}, \mathsf{i}) \wedge \mathsf{hasAddress}(\mathsf{u}, \mathsf{ad}) \wedge \mathsf{isShipped}(\mathsf{o}, \mathsf{ad}) \wedge \ldots\end{aligned}$$

The precondition $\phi$ states that the described service requires that the user with ID $\mathsf{u}$ is registered and authorized by its password $\mathsf{p}$. Both, $\mathsf{u}$ and $\mathsf{p}$ are inputs. Furthermore, the service requires for a successful execution that the book $\mathsf{b}$ with author $\mathsf{a}$ and title $\mathsf{t}$ is available.

### 3.4 Semantics of Service Description

The semantics of service descriptions translates them into a formal model. We use a labeled transition system (LTS) as formal state-based model for both service descriptions and requests. This powerful model allows us to enhance description formalisms in future work. An LTS $L = (S, \mathcal{W}, \rightarrow)$ comprises a set $S$ of states,

a labeled transition relation $\rightarrow \subseteq S \times \mathcal{W} \times S$, and a set $\mathcal{W}$ of transition labels. A state is described by the knowledge bases of involved service providers.

The description $(I, O, \phi, \psi)$ of a service $w$ is translated to an LTS $L = (S, \mathcal{W}, \rightarrow)$ such that the execution of $w$ is modeled by two states $s \in S$ and $t \in S$ and a transition $(s, w, t) \in \rightarrow$ that is labeled with $w \in \mathcal{W}$. The state $s$ is the one described by the precondition and inputs and the state $t$ is the one described by the effect, inputs, and outputs. Consequently, $s$ models the state before and $t$ models the state after service execution. In summary, the LTS $L$ that models a service description is defined as follows.

$$L = (S, \mathcal{W}, \rightarrow)$$
$$S = \{s, t\}, \mathcal{W} = \{w\}, \rightarrow = \{(s, w, t)\}$$

## 4 Service Requests and Matchmaking

A service request aims at specifying constraints in order to restrict the set of available Web services to the set of desired Web services. Within our model, this can be done by specifying (i) constraints on inputs and outputs and (ii) constraints on preconditions and effects. In Section 4.1, we show how constraints on inputs and output can be specified. Then we show in Section 4.2 how desired preconditions and effects can be expressed. Analogously to the prior section, we then present the semantics of a service request description. At the end of this section, we define matches between requests and service descriptions.

Service requests use a formalism that is different from the one of service descriptions. It allows for the description of a request such that a set of matching services is characterized. Requests are denoted by $(\mathcal{I}, \mathcal{O}, \Phi, \Psi)$. The right part of Figure 1 depicts the motivation of a clear distinction between service descriptions $D = (I, O, \phi, \psi)$ and a service request $\mathcal{R} = (\mathcal{I}, \mathcal{O}, \Phi, \Psi)$ describing a set of desired Web services.

### 4.1 Constraints on Inputs and Outputs

The specification of desired inputs (outputs) of a request describes the set of desired sets of inputs (outputs). The description of the set of sets is expressed in a logic that allows us to express conjunctions, alternatives, and exclusions of inputs and outputs.

As an example, a library system that frequently places book orders requests for services with the following input specification. Disregarding the remaining request description, the input specification $\mathcal{I}$ corresponds to the set of services that either only require an ISBN `i` or at least author `a` and title `t` of the book to order but do not require a date of birth `bday`. Any coherencies between the inputs or outputs are expressed in the formulas that model requested preconditions and effects, resp.

$$\mathcal{I} \equiv \big(\exists \mathsf{isbn} \in I \wedge \forall \mathsf{j} \in I : \mathsf{isbn} = \mathsf{j}\big) \oplus \big(\exists \mathsf{author} \in I : \exists \mathsf{title} \in I : \nexists \mathsf{bday} \in I\big)$$
$$\mathcal{O} \equiv \exists \mathsf{book} \in O$$

The input set $I$ is the set of inputs provided by the service description, which is matched against this query. The requested outputs $\mathcal{O}$ specify that desired services return a book book.

### 4.2 Constraints on Preconditions and Effects

Now we focus on formalizing the specification of requested preconditions $\Phi$ and effects $\Psi$. Similar to requesting for inputs and outputs, we aim at describing a set of desired services. Preconditions $\phi$ and effects $\psi$ of a service describe the states before and after service execution, resp., and model the available knowledge. A knowledge base is described by facts that must hold in it. Preconditions $\Phi$ and effects $\Psi$ of a request are interpreted as queries against a repository of service descriptions. The queries retrieve the set of services that accept or provide the requested conditions in the states before and after service execution modeled by $\phi$ and $\psi$, resp.

A requested precondition description $\Phi$ might be modeled as follows. The precondition request $\Phi$ is a query against the preconditions $\phi$ of service descriptions and matches those that provide the requested the required facts.

$$\Phi \equiv \exists \mathsf{b} : \mathsf{Book}(\mathsf{book}) \wedge \mathsf{hasAuthor}(\mathsf{book}, \mathsf{author}) \wedge \mathsf{hasTitle}(\mathsf{book}, \mathsf{title}) \wedge \mathsf{Author}(\mathsf{author}) \wedge \mathsf{isAvailable}(\mathsf{book}) \wedge \neg \mathsf{isRegistered}(\mathsf{user}) \wedge \mathsf{Birthday}(\mathsf{d})...$$

Using a logic like first-order logic to specify the set of requested preconditions and effects not only allows for precisely expressing which conditions are provided by service offers, but also for excluding services with undesired conditions. For example, the negation of the condition isRegistered(.) prevents matching Web services that require a user registration for the order of the specified book. Consequently, the example of $\Phi$ prevents that it will match the Web service indicated with the example precondition $\phi$ in the previous section since the service required a registered user. The description of requested effects $\Psi$ is conceptually similar and omitted due to space limitations.

### 4.3 Semantics of Service Request

A service request is translated to a set $\mathcal{L}$ of labeled transition systems. As a request describes a set of possible input sets, output sets, preconditions, and effects, the set $\mathcal{L}$ is used to formally model the set of possible service executions described by a request $(\mathcal{I}, \mathcal{O}, \Phi, \Psi)$. For each state $s \in S$ that fulfills the requested precondition $\Phi$ and for each state $t \in S$ that fulfills the requested effect $\Psi$, there exists one LTS $L \in \mathcal{L}$ with an unlabeled transition $(s, \epsilon, t) \in \longrightarrow$ in $L$. That is, there exists an LTS in $\mathcal{L}$ for each combination of a start and a end state. A start state $s$ represents a possible answer of the knowledge base query $\Phi$. Analogously, each knowledge base that is an answer to the requested effect $\Psi$ introduces one end state $t$.

In summary, an LTS $L = (S, \{\}, \rightarrow)$ that is member of the set $\mathcal{L}$ comprises one start state $s \in S$, one end state $t \in S$, and one transition $(s, \epsilon, t) \in \rightarrow$. The transition is labeled with an empty label $\epsilon$. Since a request potentially describes many services, several potential service executions modeled by the LTS' are modeled in the set $\mathcal{L}$.

For example, considering the example request above, the desired sets of inputs specified by $\mathcal{I}$ embrace $\{\mathsf{isbn}\}$, $\{\mathsf{author}, \mathsf{title}\}$, and further supersets of them. Combined with each desired precondition, a set of start states is constructed and combined to each end state that represents a combination of desired inputs, outputs, and effect.

### 4.4 Matchmaking

The discovery of semantically described services identifies service descriptions from a repository that fulfill the requirements specified in a request. After both formalisms and their translation to a common formal model were introduced, the task of discovery adds up to the identification of a containment relation between an LTS $L^d$, which models the description $(I, O, \phi, \psi)$ of a service $w$, and a set of LTS' $\mathcal{L}$, which models a request $(\mathcal{I}, \mathcal{O}, \Phi, \Psi)$. Let the indexes $d$ and $r$ of variables indicate their belonging to the LTS $L^d$ and $L^r \in \mathcal{L}$, respectively.

The description of service $w$ matches a request if and only if $L^d \in \mathcal{L}$. That is, $\exists L^r \in \mathcal{L} : L^r \equiv L^d$. Latter equivalence holds if and only if there are transitions $(s^d, w, t^d) \in \rightarrow^d$ and $(s^r, w, t^r) \in \rightarrow^r$ with (i) equivalent states $s^d \in S^d$ and $s^r \in S^r$, and (ii) equivalent states $t^d \in S^d$ and $t^r \in S^r$.

We have described a match within the formal model and we further show how this match corresponds to a match in terms of the descriptions of a service and a request. Basically, the containment relation $L^d \in \mathcal{L}$ corresponds to question answering task of a reasoner to compute the match. Therefore, the reasoner identifies a model for the query by binding the variables of the request to individuals modeled in the knowledge base that describes the service description. To show the equivalence of the match in the formal model and the match identified by a reasoner, both directions of the implication between them are discussed. For simplicity, we only consider knowledge bases that model the states before execution.

If $L^d \in \mathcal{L}$, then $s^d = s^r$ and $t^d = t^r$ as defined above. Then the knowledge bases that describe $s^d$ and $s^r$ are equal and the knowledge bases that describe $t^d$ and $t^r$ are equal. Then, there obviously exists a variable binding to answer the query against the knowledge base $KB^d$ that models the service description. In the other direction, if there exists a variable binding to answer the query, then the knowledge base $KB^d$ is a model of the request and contains at least the information that has to be satisfied to fulfill the request. Due to the definition of the request semantics, there must be also an LTS with states that are described by knowledge bases that are equal to $KB^d$, because the model $\mathcal{L}$ of a request contains all the LTS' with all possible states and respective knowledge bases that are model of the request.

The same applies to knowledge bases representing the states after service execution. Consequently, a match with respect to preconditions and effect is

given if query answering is used as the reasoning task. Matching inputs and outputs is guaranteed since they are part of the knowledge bases.

It was shown that in our discovery approach it is not necessary to distinguish different degrees of matches, which correspond to certain degrees of intersection of the set of service execution runs as depicted in Figure 1. Instead, we provided formalisms that allow users to clearly specify a query with unambiguous interpretation. The query is interpreted as the set of conditions that a matching service must at least provide. This implies, that a matching service may require further inputs, offer further outputs, require further preconditions, or generate further effects within the constraints the user could specify in a query using the expressivity we provide. If a query does not deliver any or the desired results, then query relaxation or manual query refinement can be applied. The former method can thus simulate a subsume or intersect match since a less restrictive query will be able to return services that would match a query by subsume or intersect match.

## 5 Implementation and Evaluation

We implemented the presented discovery approach that uses both formalisms for service descriptions and requests as well as the matching algorithm from Sections 3 and 4, resp. As we used a WSML reasoner[1], the syntax of descriptions and queries are bound to the syntax of the Web Service Modeling Language WSML [11]. Of course, our approach is not bound to this choice. It is possible to use a different syntax to create the query if another reasoner is used. The implementation is publicly available[2]. A simple user interface allows to enter a request. After submission, two reasoners compute the list of matching services out of a repository of randomly generated Web service descriptions. Services that were identified as match by both reasoners are results of the entire request and are thus displayed to the user.

In this section, we explain the creation of knowledge bases from service descriptions, the implementation of the matchmaking algorithm, and at the end we present some performance results.

**Knowledge Base Construction.** The description $(I, O, \phi, \psi)$ of a service $w$ describes two states. Two knowledge bases $KB_0$, $KB_e$ that model the state before and after execution, respectively, are constructed. The inputs from $I$ are parsed and modeled as instances in both knowledge bases. Output variables from $O$ are also modeled as instances but only in $KB_e$. The precondition $\phi$ is added as an axiom to the $KB_0$ that models the state before service execution and the effect $\psi$ is added as an axiom to $KB_e$. The same procedure applies for adding further service descriptions to the knowledge bases.

[1] See `http://tools.sti-innsbruck.at/wsml2reasoner` for details on the reasoning process.

[2] `http://www.aifb.uni-karlsruhe.de/WBS/mju/soa4all-discovery` subject to change. Please contact authors if necessary.

**Matchmaking.** The discovery engine receives a request $(\mathcal{I}, \mathcal{O}, \Phi, \Psi)$ from the user interface and translates it into two queries $q_0$ and $q_e$ expressed in the WSML query language syntax. The query $q_0$ is created from the requested inputs $\mathcal{I}$ and precondition $\Phi$ and is sent to the first reasoner instance that models $KB_0$. The query $q_e$ is created from the requested inputs $\mathcal{I}$, outputs $\mathcal{O}$, and effect $\Psi$ and is sent to the second reasoner instance that models $KB_e$. Both reasoner instances execute the respective queries on their knowledge bases, which may model several service descriptions. In order to answer the query $q_0$, the first reasoner determines for each service $w$ modeled in $KB_0$, whether required inputs and the precondition $\phi$ is a model of the requested precondition $\Phi$. That is, the reasoner checks whether the precondition $\Phi$ of a request is fulfilled by the facts in the A-Box that were introduced by the precondition $\phi$ of the service $w$. Query $q_e$ is processed analogously by the second reasoner instance on $KB_e$.

Below, a fragment of an example query that is sent to the first reasoner instance is presented in WSML syntax. The query contains the specification of requested inputs, their types, and the precondition. `sm#` denotes the namespace of the ontology that formalizes the service model. The shown query asks for services that have two inputs `?a` and `?t` of type Author and Title of an example ontology denoted by the namespace `ex#`, respectively. The inputs describe a book `?b` of type Book. The continuation of the example may specify further conditions on the book `?b` et cetera.

```
?w memberOf sm#Service and ?w[sm#hasPrecondition hasValue ?p] and
?w[sm#hasInput hasValue ?t] and ?t memberOf ex#Title and
?w[sm#hasInput hasValue ?a] and ?a memberOf ex#Author and
?p[sm#hasVariable hasValue ?b] and ?b memberOf ex#Book and
?b[ex#hasTitle hasValue ?t] and ?b[ex#hasAuthor hasValue ?a] and ...
```

After sending both queries $q_0$ and $q_e$ to the reasoners, the reasoners bind the variable `?w` to references of service annotations that fulfill the queries $q_0$ and $q_e$, respectively. A service is a match for a given request, if the service is an answer to both queries $q_0$ and $q_e$, i.e., the service is identified as a match by both reasoners.

**Performance Results.** We generated a repository of randomly generated service descriptions. We used the Semantic Web for Research Community ontology [12] as background knowledge base. It provides classes and properties used to model types of individuals and to express conditions used in preconditions and effects. The generated service descriptions are also available at the supplementary Web page. We measured the reasoner's mean query answering time of 100 repetitive runs. Both queries $q_0$ and $q_e$ were sent in parallel to the reasoners, which computed the answers on an ordinary laptop with dual core 2.4GHz CPU and 4GB of main memory. Both knowledge bases modeled 1000, 2000, 3000, 4000, and 5000 Web service descriptions. Queries of three different sizes were sent to each knowledge base. Small (S), medium (M), and large (L) queries with 1, 2, 3 instances and 2, 4, 6 properties on those instances were tested, respectively. Figure 2 shows the mean time in milliseconds for different knowledge base

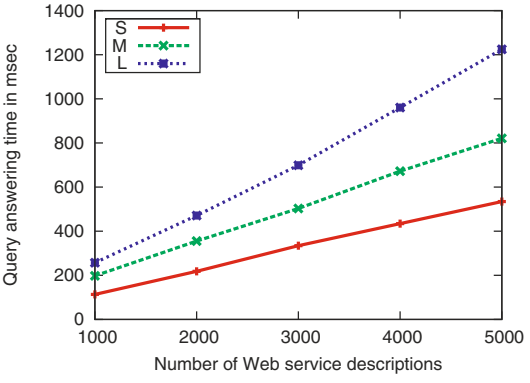



**Fig. 2.** Mean time to answer the two queries $q_0$ and $q_e$

and query size. It is not our intention to claim scalability based on these measurements. We rather want to show feasibility of the presented approach as this paper was mainly focusing on the underlying description formalisms.

## 6 Related Work

The description of the functionality of a software by preconditions and effects was introduced by [13]. In contrast to description and discovery approaches in the field of software specification, the assumption of a closed world does not hold for Web services. The ability to model side effects to the world and the consideration of background knowledge thus were not considered. Zaremski and Wing consider different match types based on the implication relations between preconditions and postconditions of software library components and a query [14].

OWL-S Profile introduced in [3] proposes to model Web services semantically with inputs, outputs, preconditions and effects. However, OWL-S Profile since being an OWL ontology can not capture the semantics of variables and thus the dynamics of Web services.

Recently, WSMO-Lite has been proposed for describing Web services semantically as the next evolutionary step after SAWSDL[3], filling SAWSDL annotations with concrete semantic service descriptions [7]. WSMO-Lite ontology is on one side lightweight and on the other side provides elements for modeling functionality of Web services. WSMO-Lite does not provide modeling of input and output parameters explicitly and relies on their derivation from the free variables in the formulas for precondition and effect. Note that such a derivation is not possible if a formula does not have any free variables but the Web service has inputs or outputs.

Description logic (DL) based approaches [15,2,16] for describing Web services propose to model inputs and outputs as concepts in description logics, while

[3] Semantic Annotations for WSDL `http://www.w3.org/2002/ws/sawsdl`

discovery, i.e., matchmaking is reduced to checking subsumption of input and output types.

Li et al. combine in [4] the use of description logics with DAML+OIL and DAML-S. Service description and request are similarly structured comprising inputs, outputs, preconditions, and effects. However, they also base their matchmaking on different matching degrees. Another problem is that DL based approaches lack the ability to describe changes in the world that are often caused by Web service executions. Consequently, more recent research activities concentrate on more detailed formalisms, for instance the state-based perspective on Web services that is discussed below. These models allow to model the dynamics of Web services.

Martin et al. presents a discovery approach in [17] that is based on OWL-S and describes services functionalities semantically by inputs, outputs, preconditions, and effects. This approach interprets preconditions as constraints that need to be satisfied for the service requester only and effects as side effects of the service execution on the world. In our approach we model conditions that hold at the service provider side since those conditions can be evaluated during service invocation and execution time.

The state-based service discovery approach [8,18] developed by Stollberg et al. uses an abstract state space as the underlying formal model of service descriptions [10]. The functionality of a Web service is formally described by the set of possible Web service executions while each normal execution of a Web service is determined by its start and end state. The discovery algorithm relies on the assumption that the precondition $\phi$ logically implies the effect $\psi$ of a Web service execution. Modeling a transition as a logical implication can be problematic, e.g., in case of a Web service that deletes a certain fact, the existence of the fact would imply non existence of the fact, e.g., a user subscription would imply that the user is not subscribed anymore.

Goal-driven approaches like [8,19,20] do not explicitly specify inputs as parts of the goal. However, a goal needs to be mapped to a request for finding appropriate Web services. In such a request, constraints on inputs can be useful, in particular if a user wishes to exclude a particular input parameter. In goal based approaches, goals are mapped to predefined goal templates that are used to find appropriate Web services. However, the usability of one global hierarchy of goal templates is hardly feasible in an open environment like the Web. One major difference between our approach and the goal based approaches is that we interpret inputs, outputs, preconditions, and effects of descriptions and requests differently, namely the former as a pair of states the latter as a pair of queries.

In contrast to the state-based approaches, Hull et al. propose a matching technique for stateless Web services in [5]. They argue that reasoning for stateful service descriptions and expressive background ontologies becomes practically impossible. Thus, this attempt solely considers inputs, outputs, and their relationships. With the restriction to conjunctive queries, the query containment problem is decidable. In our approach, we deal with stateful services since our discovery approach is not based on query subsumption but on query answering.

## 7 Conclusion and Outlook

In this paper we presented a discovery approach that is more practical as it overcomes several problems of other approaches. We thoroughly investigated the problem of using the same formalism for service descriptions and requests. A formalism to describe service requests as a set of potentially matching services was introduced. Also the formalism to describe service functionalities semantically was renovated by clarifying its interpretation. We defined the matchmaking between descriptions and requests. Therefore, the semantics of both formalisms that translates the descriptions to the same formal model was provided. We presented a prototypical implementation of our semantic discovery technique, which is a part of the larger system developed under the EU funded project SOA4All.

The focus of the present paper was to provide appropriate formalisms in a first step. Since scalability and efficiency is crucial to enable semantic Web service discovery in a large scale on the Web, we will focus on improving the performance and scalability in the future. Among other options, by the introduction of indexing structures, materialization, and more computational resources we expect to handle larger sets of semantic Web service descriptions.

We furthermore plan to integrate non-functional properties to discovery since it is also valid to request for services including the specification of desired values of non-functional properties. In the settings of an open Web with distributed service providers and consequently decentralized knowledge bases, conditions specified in preconditions and effects cannot hold generally. The functionality description of a software artifact is usually described with respect to a local and closed world of the runtime environment [14]. However, Web service functionalities cannot be described in a closed and local context as service executions may involve further external services. Preconditions and effects must be capable to distinguish different actors of the Web. Conditions and changes must therefore explicitly state where they hold. We will therefore examine techniques to refer to the respective actor in order to identify the knowledge base in which a condition holds.

**Acknowledgments.** The authors gratefully acknowledge funding by the European project SOA4All (FP7-215219, http://www.soa4all.eu).

## References

1. UDDI: UDDI Executive White Paper. Technical report, UDDI.org (2001)
2. Paolucci, M., Kawmura, T., Payne, T., Sycara, K.: Semantic Matching of Web Services Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, p. 333. Springer, Heidelberg (2002)
3. Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated Discovery, Interaction and Composition of Semantic Web Services. Web Semantics: Science, Services and Agents on the World Wide Web 1(1), 27–46 (2003)
4. Li, L., Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. Int. J. Electron. Commerce 8(4), 39–60 (2004)

5. Hull, D., Zolin, E., Bovykin, A., Horrocks, I., Sattler, U., Stevens, R.: Deciding Semantic Matching of Stateless Services. In: Proc. of 21st Nat. Conf. on Artificial intelligence (AAAI 2006), pp. 1319–1324. AAAI Press, Menlo Park (2006)
6. Stollberg, M., Martin Hepp, J.H.: A Caching Mechanism for Semantic Web Service Discovery. 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. 480–493. Springer, Heidelberg (2007)
7. Vitvar, T., Kopeck, J., Viskova, J., Fensel, D.: WSMO-Lite Annotations for Web Services. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 674–689. Springer, Heidelberg (2008)
8. Stollberg, M., Hepp, M., Hoffmann, J.: A Caching Mechanism for Semantic Web Service Discovery. 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. 480–493. Springer, Heidelberg (2007)
9. Grimm, S., Motik, B., Preist, C.: Variance in e-business service discovery. In: Proceedings of the ISWC Workshop on Semantic Web Services (2004)
10. Keller, U., Lausen, H., Stollberg, M.: On the Semantics of Functional Descriptions of Web Services. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 605–619. Springer, Heidelberg (2006)
11. de Bruijn, J., Fensel, D., Kerrigan, M., Keller, U., Lausen, H., Scicluna, J.: Modeling Semantic Web Services: The Web Service Modeling Language. Springer, Heidelberg (2008)
12. Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The SWRC Ontology - Semantic Web for Research Communities. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005. LNCS (LNAI), vol. 3808, pp. 218–231. Springer, Heidelberg (2005)
13. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM 12(10), 576–580 (1969)
14. Zaremski, A.M., Wing, J.M.: Specification matching of software components. ACM Trans. Softw. Eng. Methodol. 6(4), 333–369 (1997)
15. Benatallah, B., Hacid, M.S., Leger, A., Rey, C., Toumani, F.: On automating Web services discovery. The VLDB Journal 14(1), 84–96 (2005)
16. Gonzalez-castillo, J., Trastour, D., Bartolini, C.: Description Logics for Matchmaking of Services. In: KI 2001 Workshop on Applications of Description Logics (2001)
17. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing Semantics to Web Services: The OWL-S Approach. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005)
18. Stollberg, M., Keller, U., Lausen, H., Heymans, S.: Two-phase Web Service Discovery based on Rich Functional Descriptions. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 99–113. Springer, Heidelberg (2007)
19. Lara, R., Corella, M., Castells, P.: A Flexible Model for Locating Services on the Web. Int. J. Electron. Commerce 12(2), 11–40 (2008)
20. Keller, U., Lara, R., Lausen, H., Polleres, A., Fensel, D.: Automatic Location of Services. In: Proceedings of the 2nd European Semantic Web Symposium (ESWS 2005), Heraklion, Crete (2005)

# iSeM: Approximated Reasoning for Adaptive Hybrid Selection of Semantic Services

Matthias Klusch and Patrick Kapahnke

German Research Center for Artificial Intelligence, Saarbrücken, Germany
{klusch,patrick.kapahnke}@dfki.de

**Abstract.** We present an intelligent service matchmaker, called iSeM, for adaptive and hybrid semantic service selection that exploits the full semantic profile in terms of signature annotations in description logic $\mathcal{SH}$ and functional specifications in SWRL. In particular, iSeM complements its strict logical signature matching with approximated reasoning based on logical concept abduction and contraction together with information-theoretic similarity and evidential coherence-based valuation of the result, and non-logic-based approximated matching. Besides, it may avoid failures of signature matching only through logical specification plug-in matching of service preconditions and effects. Eventually, it learns the optimal aggregation of its logical and non-logic-based matching filters off-line by means of binary SVM-based service relevance classifier with ranking. We demonstrate the usefulness of iSeM by example and preliminary results of experimental performance evaluation.

## 1 Introduction

Semantic service selection is commonly considered key to the discovery of relevant services in the semantic Web, and there are already quite a few matchmakers available for this purpose [11]. In this paper, we present the first adaptive semantic service IOPE matchmaker. In essence, its innovative features are (a) approximated logical signature (IO) matching based on non-monotonic concept abduction and contraction together with information-theoretic similarity and evidential coherence-based valuation of the result to avoid strict logical false negatives, (b) stateless strict logical specification (PE) plug-in matching to avoid failures of signature matching only, and (c) SVM (support vector machine)-based semantic relevance learning adopted from [9] but extended to full functional service profile (IOPE) matching and use of approximated IO matching results to prune the feature space for precision. Preliminary evaluation results particularly indicate that this kind of approximated logical matching can perform significantly better than its strict logical counterpart, performs close to its non-logic-based approximated counterparts, that are text and structural matching, and does even more so in adaptive hybrid combination with the latter.

The remainder of the paper is structured as follows. We motivate and provide an overview of the matchmaker iSeM in Sections 2 and 3. This is followed by a detailed description of its signature matching filters with focus on approximated

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 30–44, 2010.

logical matching in Section 4, while Section 5 discusses its stateless, logical specification matching filter. Section 6 describes the SVM-based service relevance learning for selection, while preliminary evaluation results are provided in Section 7. Eventually, we comment on related work in Section 8 and conclude in Section 9.

## 2 Motivation

The specific problems of semantic service selection the matchmaker iSeM has been particularly designed to cope with are motivated by the following service example, which is used throughout the paper.

**Example 1.** Consider the semantic profiles of service request $R$ and offer $S$ in Figure 1, taken from the standard test collection OWLS-TC3 according to which $S$ is relevant to $R$.

The desired service $R$ is supposed to purchase a book for a given person by debiting his own debit account, shipping the book to him and eventually acknowledging the completed deal. The e-shopping service $S$ like amazon.com offers arbitrary articles including books that are requested by some customer whose own credit card account gets respectively charged while sending an invoice for and pricing information about the deal. Both services are written in OWL-S with semantic signature (IO) concept definitions in description logic $\mathcal{SH}$ and their logical preconditions and effects (PE) in SWRL. In the following, we assume the matchmaker to have an appropriate shared ontology and a service registry available over which semantic service selection is performed. ○

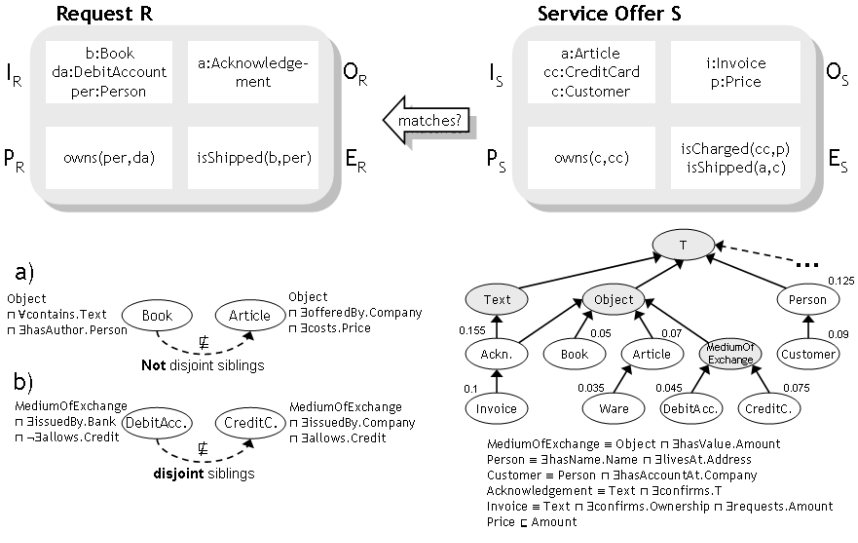



**Fig. 1.** Service request (book purchasing) and relevant service offer (article purchasing)

**False negatives of strict logical signature matching.** The majority of semantic service matchmakers perform logical signature matching [11]. One prominent set of strict logical matching filters for this purpose is provided below [17,8]. Each of these filters requires (a) each service input concept to be more generic than or equal to those provided in the request and (b) the complete requested output to be covered by that of the service in terms of the respectively considered type of logical concept subsumption relation.

**Definition 1.** *Strict logical signature matching.*
Let $S, R$ semantic services, $in(S), out(S), in(R), out(R)$ the sets of input, resp., output concepts of semantic signatures of $S$ and $R$ defined in a shared OWL ontology $O$; $BPG_{\sqsubseteq}(\bar{C}, \bar{D})$ the set of injective concept assignments $(C, D)$, $C \in \bar{C}$, $D \in \bar{D}$ as valid solution of bipartite graph matching with maximized sum of binary weights $v$ of edges between concepts as nodes in the graph indicating whether the considered type of strict logical subsumption relation ($\equiv, \sqsupseteq, \sqsubseteq_1, \sqsupseteq_1$) holds ($v = 1$ iff $C \sqsubseteq D$ else $v = 0$); $BPG_{\sqsubseteq}(\bar{C}, \bar{D}) = \emptyset$ iff no such assignment is possible ($|\bar{D}| < |\bar{C}|$). $BPG_X(\bar{C}, \bar{D})$ with $X \in \{(\equiv, \sqsubseteq_1, \sqsupseteq_1\}$ are defined analogously.

The degree $\text{MatchIO}_{Logic}(R, S)$ of strict logical signature matching is as follows:
$\text{MatchIO}_{Logic}(R, S) \in \{$ **Exact, Plug-in, Subsumes, Subsumed-by, LFail** $\}$ with

**Exact:** $S$ equivalent to $R$
$\Leftrightarrow \forall\ I_S \in in(S)\ \exists\ I_R \in in(R)$: $(I_S, I_R) \in BPG_{\equiv}(in(S), in(R))$
$\wedge\ \forall\ O_R \in out(R)\ \exists\ O_S \in out(S)$: $(O_R, O_S) \in BPG_{\equiv}(out(R), out(S))$
**Plug-in:** $S$ plugs into $R$
$\Leftrightarrow \forall\ I_S \in in(S)\ \exists\ I_R \in in(R)$: $(I_S, I_R) \in BPG_{\sqsupseteq}(in(S), in(R))$
$\wedge\ \forall\ O_R \in out(R)\ \exists\ O_S \in out(S)$: $(O_R, O_S) \in BPG_{\sqsupseteq_1}(out(R), out(S))$
**Subsumes:** $R$ subsumes $S$
$\Leftrightarrow \forall\ I_S \in in(S)\ \exists\ I_R \in in(R)$: $(I_S, I_R) \in BPG_{\sqsupseteq}(in(S), in(R))$
$\wedge\ \forall\ O_R \in out(R)\ \exists\ O_S \in out(S)$: $(O_R, O_S) \in BPG_{\sqsupseteq}(out(R), out(S))$
**Subsumed-by:** $R$ subsumed by $S$
$\Leftrightarrow \forall\ I_S \in in(S)\ \exists\ I_R \in in(R)$: $(I_S, I_R) \in BPG_{\sqsupseteq}(in(S), in(R))$
$\wedge\ \forall\ O_R \in out(R)\ \exists\ O_S \in out(S)$: $(O_R, O_S) \in BPG_{\sqsubseteq_1}(out(R), out(S))$
**LFail:** None of the above logical filter constraints are satisfied. ◇

Applying these strict logical matching filters to the example above produces a logical fail (LFail), hence a false negative. The reasons are that (a) the inputs book and article are not strictly logically disjoint siblings in the ontology, that is ($Book \sqcap Article \not\sqsubseteq \bot$), and (b) the inputs debit account and credit card are strictly logically disjoint, that is ($DebitAccount \sqcap CreditCard \sqsubseteq \bot$).
Such cases of logical signature mismatches may appear quite often, in fact, applying the above filters to the standard collection OWLS-TC3 yields a relatively high number of strict logical false negatives for each request in the order of 45% of the size of its relevance set in average. As shown, for example, in [9,8,10] and the contest S3 (http://www.dfki.de/-klusch/s3), some hybrid combination of strict logical with non-logic-based approximated signature matching methods may avoid failures of strict logical signature matching filters defined above in practice[1]. But how can logical matching itself be improved by what kind of

[1] Avoidance or higher (lower) ranking of false negatives (positives) increases average precision of ranked result lists.

complementary approximation (cf. Section 4), and how well does this perform compared to and in combination with its non-logic-based counterparts in practice (cf. Section 7)?

**Failures of signature matching only.** It is well known that matching of semantic signatures only may fail in many cases, since they do not capture the functional behavior commonly encoded in logical service preconditions and effects (PE). There are different approaches to logical PE-matching [11] - but which one to use in case of a third-party matchmaker that usually has no access to concept instances for registered semantic services (cf. Section 5)?

**Best combination of semantic matching filters.** How to best combine different kinds of semantic service matching filters in terms of precision? One option proposed, for example, in [9,10] is to let the matchmaker learn the optimal aggregation of different matching results for its semantic relevance decision - rather than to put the burden of finding and hard-coding the solution by hand on the developer. Though this turned out to be quite successfull in the S3 contest restricted to semantic signatures, how can approximated logical matching be used to improve the learning for better precision of service selection (cf. Section 6)?

## 3 iSeM Matchmaker: Overview

Before delving into the technical details of the matchmaker iSeM, we shall first provide an overview of its functionality.

**Matchmaking algorithm in brief.** For any given service request, the iSeM matchmaker returns a ranked set of relevant, registered services as its answer set to the user. For this purpose, it first learns the weighted aggregation of different kinds of service IOPE matching results off line over a given training set of positive and negative samples by means of SVM-based binary relevance classification with ranking. These different kinds of matching a given service request $R$ with service offers $S$ in OWL-S or SAWSDL concern strict and approximated logical, text similarity-based and structural semantic matching of service signatures (IO) in $\mathcal{SH}$, as well as stateless, logical plug-in matching of service preconditions and effects (PE) in SWRL, if they exist.[2] Once learning has been done, the same filters are used by the learned relevance classifier for selecting relevant services for previously unknown requests. iSeM classifies as adaptive, hybrid semantic service IOPE matchmaker [11].

**Logical signature (IO) matching.** Logical signature matching of iSeM comes in two complementary flavors: Strict logical matching and approximated logical matching. For every service pair $(R, S)$ for which strict logical signature matching

[2] Restriction to annotation in $\mathcal{SH}$ is due to respective limitation of the adopted concept abduction reasoner [3]; its extension to $\mathcal{SHOIN}$ is ongoing.

as defined above (Section 2, Def. 1) fails, iSeM computes the approximated logical matching degree $\text{MatchIO}_{ALogic}(R, S)$ based on approximated subsumption relations ($C \sqsubseteq_{AC} D$) between I/O concepts $C, D$ via contraction and structured abduction together with their information-theoretic valuation. This leads to two hypotheses of approximated logical signature matching, that are approximated logical plug-in ($H_1$) and subsumed-by ($H_2$), both of which weighted by their averaged informative quality $v \in [-1, 1]$. Eventually, the degree $\text{MatchIO}_{ALogic}(R, S) = (H,\ v)$ of approximated logical service signature matching is determined as the hypothesis $H$ with maximal valuation $v$. The approximated logical matching results are used in the learning process over a given training set of service pairs to prune the respective feature space restricted to logic-based matching to compensate for strict logical false negatives. In addition, iSeM performs non-logic-based approximated, that are text and structural semantic similarity-based signature matching for which purpose it applies the respective filters of OWLS-MX3 [9] (cf. Section 4).

**Logical specification (PE) matching.** To cope with failures of signature matching only and allow for third-party matchmaking without having access to service concept instances, iSeM performs stateless, logical plug-in matching $\text{MatchPE}(S, R)$ of service preconditions and effects by means of approximated theorem proving, that is theta-subsumption, of required logical PE-implications like in LARKS[17] (cf. Section 5).

**Learning of full service profile (IOPE) selection.** To combine the results of its different IOPE matching filters for optimal precise service selection, iSeM performs binary SVM-based semantic relevance learning off line over a given training set of positive and negative samples $(S, R)$ each of which is represented as a vector $x$ in the 10-dimensional feature space of different matching filters. This space gets particularly pruned by exploiting the approximated logical signature matching results to compensate for strict logical false negatives. Once that has been done, the learned binary classifier $d$ with ranking $r$ is applied by iSeM to any service pair $(S, R)$ with unknown request $R$ to return the final result: $\text{MatchIOPE}(S, R) = (d,\ r)$ (cf. Section 6, 7).

## 4 Hybrid Semantic Signature Matching

Semantic signature matching by iSeM is performed by means of both logic-based and non-logic-based matching. While the first type basically relies on strict logical (cf. Definition 1) and approximated logical concept subsumptions (cf. Section 4.1), the second exploits text and structural similarities of signature concepts (cf. Section 4.2). Both kinds of approximated logical and non-logic-based matching are performed by iSeM in particular to compensate for strict logical signature matching failures in due course of its relevance classification learning (cf. Section 6).

### 4.1 Approximated Logical Matching

Inspired by [2,3,14], approximated logical signature matching of a given service pair $(S, R)$ relies on the combined use of logical contraction and abduction of signature concepts for approximated concept subsumption (cf. Definition 2) which is valuated in terms of the information gain and loss induced by its construction (cf. Definition 3). Eventually, we extend both means of approximation and valuation on the concept level to its application on the signature level (cf. Definition 4).

**Definition 2.** *Logical concept contraction and abduction* [2,3].
Let $C, D$ concepts of ontology $O$ in $\mathcal{SH}$.
The *contraction* of $C$ with respect to $D$ is $CCP(C, D) = (G, K)$ with $C \equiv G \sqcap K$ and $K \sqcap D \not\sqsubseteq \bot$.[3]
The abducible concept $K^h$ is derived from concept $K$ through rewriting operations [3]: $K^h = h_0 \sqcap rew(K)$, $rew(A) = A$, $rew(\neg A) = \neg A$, $rew(C_1 \sqcap C_2) = rew(C_1) \sqcap rew(C_2)$, $rew(\exists R.C) = \exists R.(h_i \sqcap rew(C))$ and $rew(\forall R.C) = \forall R.(h_i \sqcap rew(C))$;
where $i$ is incremented per application of $rew$, $A$ primitive component (in the logical unfolding of $K$ in $O$), $C_i$ concepts in $\mathcal{SH}$, and $\bar{H} = (h_0, \ldots, h_n)$.
*Structural abduction* of concept $K$ with respect to $D$ is $SAP(K, D) = H = (H_0, \ldots, H_n)$ with $\sigma[\bar{H}, H](K^h) \sqsubseteq D$ and $\sigma[\bar{H}, H](K^h) \not\sqsubseteq \bot$.
The *approximated concept* $C' := \sigma[\bar{H}, H](K^h)$ of $C$ with respect to $D$ is constructed by applying $\sigma[\bar{H}, H] = \{h_0 \mapsto H_0, \ldots, h_n \mapsto H_n\}$ to the abducible concept $K^h$.
The *approximated logical concept subsumption* $C \sqsubseteq_{AC} D$ is defined as follows: $C \sqsubseteq_{AC} D \Leftrightarrow C' \sqsubseteq D$ with $(G, K) = CCP(C, D)$, $H = SAP(K, D)$ and $C' = \sigma[\bar{H}, H](K^h)$. $\diamond$

To avoid strict logical false negatives for increasing average precision, iSeM assumes the user to be consent to give up those parts of logical signature concept definitions that cause strict logical subsumption failures and keeping the remaining parts instead. The latter are used to compute approximated concept subsumption relations and the respectively approximated signature matching.

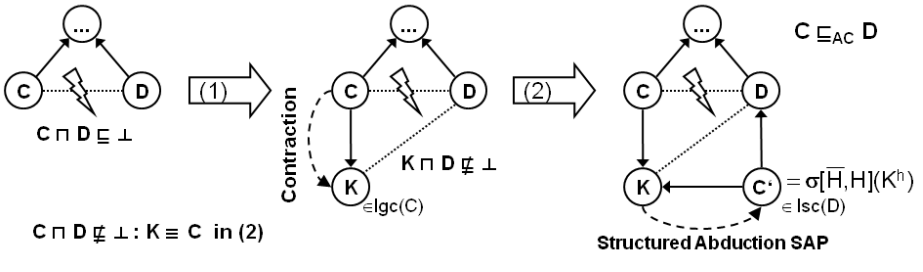



**Fig. 2.** Approximated logical concept subsumption

[3] $K$ ("'keep"') denotes the compatible part of $C$ with respect to $D$, while $G$ ("'give up"') denotes the respectively incompatible part.

**Example 2.** Consider Example 1. The approximated logical subsumption between strict logically disjoint siblings $DebitAccount, CreditCard$ is computed as follows:
$(G, K) = CCP(DebitAccount, CreditCard)$
$= (\neg\exists allows.Credit^P, MediumOfExchange \sqcap \exists issuedBy.Bank^P)$,
$K^h = h_0 \sqcap Object^P \sqcap \exists hasValue.(h_1 \sqcap Value^P) \sqcap \exists issuedBy.(h_2 \sqcap Bank^P)$,
$\bar{H} = (h_0, h_1, h_2)$,
$H = SAP(DebitAccount, CreditCard) = (\exists allows.Credit^P, \top, Company^P)$,
$\sigma[\bar{H}, H] = \{h_0 \mapsto \exists allows.Credit^P, h_1 \mapsto \top, h_2 \mapsto Company^P\}$,
$DebitAccount' = \sigma[\bar{H}, H](K^h)$
$= \exists allows.Credit \sqcap MediumOfExchange \sqcap \exists issuedBy.(Bank \sqcap Company)$,
It holds that $DebitAccount' \sqsubseteq CreditCard$, hence $DebitAccount \sqsubseteq_{AC} CreditCard$. ∘

In order to rank the computed approximations, we valuate them by means of their informative quality. Roughly, the informative quality of approximated logical subsumption between signature concepts $C, D$ is the difference between the information gain and loss induced by its construction. That is, the utility of the respectively approximated concept $C'$ is the trade off between its information-theoretic similarity [14] with the original concept $C$ and the targeted one $D$. The similarity bases on the probabilistic information content of concepts with respect to the frequency of their occurrence in semantic service signatures.

**Definition 3.** *Informative quality of approximated concept subsumption.*
Let $SR$ set of service offers registered at the matchmaker, $in(S)$, $out(S)$ multi-set of concepts used for signature (IO) parameter annotation of service $S$, $SAC(SR)$ set of all concepts used for annotating services in $SR$.
We define the *informative quality* $v$ *of approximated concept subsumption* $C \sqsubseteq_{AC} D$ (cf. Definition 2) as: $v(C, D) = sim_{inf}(C', D) - (1 - sim_{inf}(C', C))$
with the information-theoretic similarity of concepts $C$ and $D$ taken from [14]:
$sim_{inf}(C, D) = 2 \cdot IC(maxdcs(C, D))/(IC(C) + IC(D))$
where $maxdcs(C, D) = argmax_{c \in dcs(C,D)}\{IC(c)\}$ is the direct common subsumer ($dcs$) of $C, D$ in ontology $O$ with maximum information content $IC(c)$.
The *information content* of concept $C \in \text{SAC}(S, R)$ is $IC(C) = -\log P(C)$, else $IC(C) := max_{D \in SAC(SR)}\{IC(D)\}$.
We define the *probability of concept* $C$ being used for semantic service annotation as the frequency of its occurrence in semantic signatures of services in service registry $SR$:
$P(C) = \sum_{S \in SR} \frac{|\{D \in in(S) \cup out(S) : D \sqsubseteq C\}|}{|in(S) \cup out(S)|}$. ◇

**Example 3.** Informative quality of $DebitAccount \sqsubseteq_{AC} CreditCard$ in Example 2 is computed as follows:

$IC(DebitAccount) = -\log P(DebitAccount) = -\log 0.045 \approx 1.348$ ,
$IC(CreditCard) = -\log P(CreditCard) = -\log 0.075 \approx 1.125$ ,
$IC(DebitAccount') = -\log 0.075 \approx 1.727$ ,
$sim_{inf}(DebitAccount', CreditCard) = \frac{2 \cdot 1.125}{1.727 + 1.125} \approx 0.789$ ,
$sim_{inf}(DebitAccount', DebitAccount) = \frac{2 \cdot 1.348}{1.727 + 1.348} \approx 0.877$ ,
$v(DebitAccount, CreditCard) = 0.789 - (1 - 0.877) = 0.666$. ∘

For each service pair, depending on the computed type of their approximated signature concept subsumption relations one can determine two hypotheses of approximated logical service signature matching, that are approximated logical plug-in and approximated subsumed-by, each of which with maximal informative quality through respective bipartite concept graph matchings.

**Definition 4.** *Approximated logical signature match with informative quality.* Let $S, R$ semantic services, $in(S), out(S), in(R), out(R)$ multisets of their signature concepts and $BPG_{\sqsubseteq_{AC}}(\bar{C}, \bar{D})$ the concept assignment via bipartite graph matching as in Definition 1 but with approximated subsumption $\sqsubseteq_{AC}$ and informative quality of edge weights $v(C, D)$ for $C \in \bar{C}$, $D \in \bar{D}$; $BPG_{\sqsupseteq_{AC}}(\bar{C}, \bar{D})$ analogously with edge weights $v(D, C)$.
*Approximated logical plug-in signature matching hypothesis* $H_1(R, S)$ holds iff:
$\forall I_S \in in(S) \exists I_R \in in(R)$ s.t. $(I_S, I_R) \in BPG_{\sqsupseteq_{AC}}(in(S), in(R))$
$\wedge \forall O_R \in out(R) \exists O_S \in out(S)$ s.t. $(O_S, O_R) \in BPG_{\sqsubseteq_{AC}}(in(S), in(R))$.
*Approximated logical subsumed-by signature matching hypothesis* $H_2(R, S)$ holds iff:
$\forall I_S \in in(S) \exists I_R \in in(R)$ s.t. $(I_S, I_R) \in BPG_{\sqsupseteq_{AC}}(in(S), in(R))$
$\wedge \forall O_R \in out(R) \exists O_S \in out(S)$ s.t. $(O_S, O_R) \in BPG_{\sqsupseteq_{AC}}(in(S), in(R))$.
*Informative quality* $val_{(S,R)} : \{H_1, H_2\} \mapsto [-1, 1]$ *of an approximated signature matching hypothesis* is the average of informative qualities of its respective approximated concept subsumptions:

$$\begin{aligned} val_{(S,R)}(H_1) &= \tfrac{1}{2 \cdot |in(S)|} \cdot \textstyle\sum_{(I_R, I_S) \in BPG_{\sqsupseteq_{AC}}(in(R), in(S))} v(I_R, I_S) \\ &\quad + \tfrac{1}{2 \cdot |out(R)|} \cdot \textstyle\sum_{(O_S, O_R) \in BPG_{\sqsubseteq_{AC}}(out(S), out(R))} v(O_S, O_R). \\ val_{(S,R)}(H_2) &= \tfrac{1}{2 \cdot |in(S)|} \cdot \textstyle\sum_{(I_R, I_S) \in BPG_{\sqsupseteq_{AC}}(in(R), in(S))} v(I_R, I_S) \\ &\quad + \tfrac{1}{2 \cdot |out(R)|} \cdot \textstyle\sum_{(O_S, O_R) \in BPG_{\sqsupseteq_{AC}}(out(S), out(R))} v(O_S, O_R). \end{aligned}$$

The *approximated logical signature matching degree* is the approximation hypothesis with maximum informative quality:
$\text{MatchIO}_{ALogic}(S, R) := (H, v)$ with $H = argmax_{x \in \{H_1, H_2\}} val(x)$ and $v = val_{(S,R)}(H)$.
Semantic relevance ranking of services $S$ bases on $\text{MatchIO}_{ALogic}(S, R)[2] \in [-1, 1]$.
*Binary relevance classification by approximated logical matching*:
$\text{MatchIO}_{ALogic}(S, R)^* = 1$ iff $\text{MatchIO}_{ALogic}(S, R)[2] > 0$, else $\text{MatchIO}_{ALogic}(R, S)^* = 0$. ◇

**Example 4.** Consider Examples 1 - 3. The approximated logical signature match of $S, R$ is comuted as follows:
$BPG_{\sqsupseteq_{AC}}(in(R), in(S)) = \{(Book, Article), (DebitAccount, CreditCard), (Person, Customer)\}$, $BPG_{\sqsubseteq_{AC}}(out(S), out(R)) = \{(Invoice, Acknowledgement)\}$ ,
$val_{(S,R)}(H_1) = \frac{1}{2 \cdot 3} \cdot (0.829 + 0.666 + 0.927) + \frac{1}{2 \cdot 1} \cdot 0.895 = 0.8905$. In this example, the same valuation holds for $H_2$, and $\text{MatchIO}_{ALogic}(S, R) := (H_1, 0.8905)$ ○

Obviously, the approximated logical matching relation $\text{MatchIO}_{ALogic}(R, S)$ always exists, and its binary decision variant $\text{MatchIO}_{ALogic}(R, S)^*$ is redundant to its logical counterpart $\text{MatchIO}_{Logic}(R, S)$ with respect to positive service selection, that is their true and false positives are the same, but not vice versa. The latter fact is used in iSeM to restrict its computation of approximated

logical signature matches in the learning phase to cases of strict logical false negatives only and use the evidential coherence of the matching results to heuristically prune the feature space for precision (cf. Section 6.2).

### 4.2 Text and Structural Signature Matching

Non-logic-based approximated signature matching can be performed by means of text and structural similarity measurement. For iSeM, we adopted those of the matchmaker OWLS-MX3, since they have been experimentally shown to be most effective for this purpose [9]. For text matching of signature concepts in the classical vector space model, their unfoldings in the shared ontology are represented as weighted keyword vectors for token-based similarity measurement, while the structural semantic similarity of concepts relies on their relative positioning in the subsumption graph, in particular on the shortest path via their direct common subsumer and its depth in the taxonomy [13].

**Definition 5.** *Approximated non-logic-based signature matching*
Let $SR$ service registry, $\mathcal{I}$ text index of service signature concepts, shared ontology $O$, $S_{in}$ TFIDF weighted keyword vector of conjunction of unfolded input concepts of $S$.
*Text similarity-based signature matching* is the average of the respective signature concept similarities: $\text{MatchIO}_{Text}(S,R) = (sim_{text}(S_{in},R_{in}) + sim_{text}(S_{out},R_{out}))/2 \in [0,1]$ with Tanimoto coefficient (Cosine similarity) $sim_{text}(S_j,R_j)$, $j \in \{in,out\}$.
*Structural semantic signature matching* is the averaged maximal structural similarity of their signature concepts:
$\text{MatchIO}_{Struct}(S,R) = (sim_{struct}(in(S),in(R)) + sim_{struct}(out(S),out(R)))/2 \in [0,1]$ with $sim_{struct}(A,B) = 1/|A| \sum_{a \in A} max\{sim_{csim}(a,b) : b \in B\}$, and structural concept similarity adopted from [13]: $sim_{csim}(C,D) = e^{-\alpha l}(e^{\beta h} - e^{-\beta h})/(e^{\beta h} + e^{-\beta h})$ if $C \neq D$ else 1, with $l$ shortest path via direct common subsumer between given concepts and $h$ its depth in $O$, $\alpha = 0.2$ and $\beta = 0.6$ weighting parameters adjusted to structural features of ontology $O$. ◇

**Example 5.** Applied to Example 1, we obtain a high score for text-based signature matching $\text{MatchIO}_{text}(S,R) = 0.71$ which correctly accounts for semantic relevance of $S$ to $R$, hence avoids the strict logical false negative. The same holds for the structural semantic matching $\text{MatchIO}_{struct}(S,R) = 0.69$. For example, text and structural similarities of the strict logically disjoint input concept siblings *DebitAccount* and *CreditCard* are high ($sim_{text}(DA,CC) = 0.94$, $sim_{csim}(DA,CC) = 0.63$) which indicates their semantic proximity. ○

While text matching of signatures may avoid strict logical matching failures, structural semantic matching may also compensate for text matching failures, in particular when mere is-a ontologies with inclusion axioms only are used for semantic annotation of service signatures. For reasons of space limitation, we refer to [9] for more details and examples.

## 5 Stateless Logical Specification Matching

As mentioned above, semantic signatures of services do not cover functional service semantics usually encoded in terms of logical service preconditions and effects such that signature matching only may fail. Though semantic service descriptions rarely contain such specifications in practice [12], we equipped the implemented iSeM matchmaker with the most prominent PE-matching filter adopted from software retrieval: Logical specification plug-in matching.

**Definition 6.** *Stateless, logical specification plug-in matching.*
Let $(S, R)$ services with preconditions $(Pre_R, Pre_S)$ and effects $(Eff_R, Eff_S)$ defined in SWRL. Service $S$ *logically specification-plugin matches* $R$:
MatchPE$(S, R)$iff $\models (Pre_R \Rightarrow Pre_S) \wedge (Eff_S \Rightarrow Eff_R)$.

*Stateless checking* of MatchPE$(S, R)$ in iSeM 1.0 is adopted from LARKS [17]: Preconditions and effects specified as SWRL rules are translated into PROLOG as in [18] and then used to compute the required logical implications by means of $\theta$-subsumption checking stateless, that is without any instances (ABox)[4], as given in [19]:

$$(\forall p_S \in Pre_S : \exists p_R \in Pre_R : p_R \leq_\theta p_S) \Rightarrow (Pre_R \Rightarrow Pre_S)$$
$$(\forall e_R \in Eff_R : \exists e_S \in Eff_S : e_S \leq_\theta e_R) \Rightarrow (Eff_S \Rightarrow Eff_R)$$

A clause $C$ $\theta$-subsumes $D$, written $C \leq_\theta D$, iff there exists a substitution $\theta$ such that $C\theta \subseteq D$ holds; $\theta$-subsumption is an incomplete, decidable consequence relation [6]. $\diamond$

**Example 6.** If applied to Example 1, this PE-matching filter succeeds, hence avoids the respective false negative of strict logical signature matching only. Further, consider a service pair $(S', R')$ having the same, identical, or strict logically equivalent semantic signatures as $(S, R)$ given in Example 1 - but with the requested effect to only register a book at a given local index such that service $S'$ is irrelevant to $R'$: The false positive $S'$ of (strict or approximated) logical signature matching only can be avoided by additional specification plug-in matching which, in this case, would correctly fail. $\circ$

## 6 Off-Line Service Relevance Learning

In order to find the best combination of its different matching filters for most precise service selection, iSeM learns their optimal weighted aggregation by using a support vector machine (SVM) approach. In particular, the underlying feature space is pruned by evidential coherence-based weighting of approximated against strict logical signature matching results over a training set to improve precision.

### 6.1 Overview: Learning and Selection

The training set $TS$ is a random subset of the given service test collection $TC_{\mathcal{SH}}$ created from a given standard service retrieval collection $TC$ by restricting service annotations to $\mathcal{SH}$. It contains user-rated service pairs $(S, R)$ each of which

[4] Third-party matchmakers may not have access to service instances of providers.

with 10-dimensional matching feature vector $x_i$ for positive and/or negative service relevance samples $(x_i, y_i) \in X \times \{1, -1\}$ in the possibly non-linearly separable[5] feature space $X$. The different matching results for $(S, R)$ are encoded as follows: x[1] ... x[5]$\in \{0,1\}^5$ for MatchIO$_{Logic}(R, S)$ in decreasing order; x[6] = $val_{(S,R)}(H_1)$ and x[7] = $val_{(S,R)}(H_2) \in [-1, 1]$ for MatchIO$_{ALogic}(R, S)$; x[8]$\in$ $[0, 1]$ for MatchIO$_{Text}(R, S)$; x[9]$\in$ $[0, 1]$ for MatchIO$_{Struct}(R, S)$; and x[10]$\in \{0, 1\}$ for MatchPE$(R, S)$. For example: $x = (0, 0, 0, 0, 1, 0.85, 0, 0.4, 0.6, 1)$ encodes a strict logical fail but approximated logical plugin with informative quality of 0.85, text (structural) match of 0.4 (0.6) and plugin specification match.

The SVM-based classification learning problem of iSeM then is to find a separating hyperplane $h$ in $X$ such that for all samples $(x, y) \in TS$ for $(S, R)$ with minimal distances to $h$ these distances are maximal. This yields a binary relevance classifier $d(x)$ with respect to the position of feature vector $x$ to the separating $h$ while ranking of $S$ is according to the distance $dist(x)$ of $x$ for $(S, R)$ to $h$. Once that has been done, the learned classifier can be applied to any service pair $(S, R)$ with potentially unknown request $R$ and returns MatchIOPE$(S, R)$ = $(d(x)$, $dist(x))$. As kernel of the SVM, iSeM uses the *Radial Basis Function* (RBF) and performs 6-folded cross-validation. For more details of this learning process in general, we refer to [9,10].

### 6.2 Evidential Coherence-Based Feature Space Pruning

To improve the performance of the binary SVM-based relevance classier to be learned by iSeM, iSeM exploits information available from the given trainings set $TS$ to prune the feature space $X$ based on the classification results of strict Vs. approximated logical signature matching. Due to redundance of both logical matching types for (true and false) positive classification, it restricts the pruning of feature vectors $x \in X$ to cases of strict logical matching failures $(MatchIO_{ALogic}(R, S) = LFail)$. The respective set $Ev = \{(x, y) : x[5] = 1\}$ of classification events is partitioned with respect to binary classification results of approximated logical matching (MatchIO$_{ALogic}(R, S)$*) for these cases as follows:

$$E_1 = \{(x, y) \in Ev : y = 1 \wedge (x[6] > 0 \vee x[7] > 0)\}, \quad (1)$$

$$E_2 = \{(x, y) \in Ev : y = 0 \wedge x[6] \leq 0 \wedge x[7] \leq 0\}, \quad (2)$$

$$E_3 = \{(x, y) \in Ev : y = 1 \wedge x[6] \leq 0 \wedge x[7] \leq 0\}, \quad (3)$$

$$E_4 = \{(x, y) \in Ev : y = 0 \wedge (x[6] > 0 \vee x[7] > 0)\}. \quad (4)$$

For example, $E_1$ denotes all relevant samples (x,y)$\in$ $Ev$ classified correctly as (true) positives by $MatchIO_{ALogic}$ while $E_2$ contains all irrelevant samples (x,y)$\in$ $Ev$ classified correctly as (true) negatives by $MatchIO_{ALogic}$. The sets $E_3$ and $E_4$ contain wrong classifications of approximated matching, hence are

[5] For example, the feature space for the test collection OWLS-TC3 is non-linearly separable.

redundant to their strict logical counterpart and deleted from the respectively pruned feature space for learning.

Inspired by the work of Glass [4], the feature space $X$ is pruned further by modification of logical matching results of feature vectors $x \in X$ of samples in $E_1$ or $E_2$ based on evidential coherence-based weighting of approximated matching results as follows:

$$(x,y) \in E_1 \wedge x[6] \geq x[7] \mapsto x[5] := 0, x[6] := Co(H_1^+, E^+) \cdot x[6], x[7] := 0,$$
$$(x,y) \in E_1 \wedge x[6] < x[7] \mapsto x[5] := 0, x[6] := 0, x[7] := Co(H_2^+, E^+) \cdot x[7],$$
$$(x,y) \in E_2 \wedge x[6] \geq x[7] \mapsto x[6] := Co(H_1^-, E^-) \cdot x[6], x[7] := 0,$$
$$(x,y) \in E_2 \wedge x[6] < x[7] \mapsto x[6] := 0, x[7] := Co(H_2^-, E^-) \cdot x[7].$$

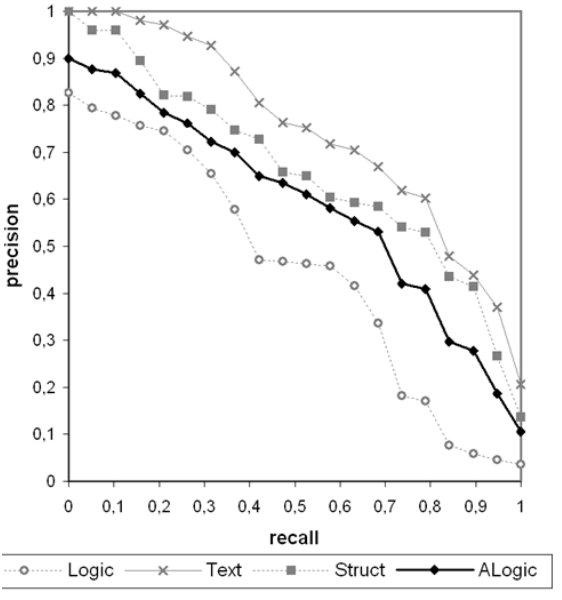
In case of true positive of approximated logical matching, the encoded strict logical misclassification in $x$ is discarded ($x[5] = 0$), and the respective approximation ($H_1$ or $H_2$) is weighted with the evidential coherence value of one of the following hypotheses (A1, A2) of relevance explanation: (A1) $MatchIO_{ALogic}$ is a correct explanation of semantic *relevance* (avoids logical false negatives), and (A2) $MatchIO_{ALogic}$ is a correct explanation for semantic *irrelevance* (avoids introduction of false positives). While hypothesis A1 (A2) is represented by special case set $H_i^+$ ($H_i^-$), the set $E^+$ ($E^-$) provides cases of observed evidence for relevance (irrelevance) in the test collection.

Which of both hypotheses of semantic relevance explanation is best with respect to a given test collection? Following [4], iSeM determines the quality of an explanation by measuring the impact of evidence $E$ on the probability of explanation $H$ (with coherence or confirmation measures) rather than measuring its posterior probability with Bayes. In other words, it determines the most plausible explanation $H$ instead of the most probable measured in terms of its coherence with evidence $E$ over given training set. The coherence overlap measure $Co(H,E) = \frac{P(H \cap E)}{P(H \cup E)}$ performed best in practice [4], and is used by iSeM to compute the weights of approximated logical signature matching results (x[6], x[7]) for respective feature space pruning as defined above.

**Example 7.** Consider training set $TS$ with $|Ev| = 20$, $|E_1| = 10$ and $|E_4| = 1$. $E_1$ contains 8 events (cases) of approximated plug-in matching ($x[6] \geq x[7]$), the only event in $E_4$ is also an approximated plug-in match. Required posterior probabilities for $Co(H_1^+, E^+)$ are computed as follows: $P(H_1^+) = \frac{|\{x \in E_1 \cup E_4 : x[6] \geq x[7]\}|}{|Ev|} = \frac{9}{20}$, $P(E^+) = \frac{|E_1 \cup E_3|}{|Ev|} = \frac{14}{20}$, $P(H_1^+|E^+) = \frac{|\{x \in E_1 : x[6] \geq x[7]\}|}{|E_1 \cup E_3|} = \frac{8}{14}$. The resulting evidential coherence-based weight of approximated logical matching is: $Co(H_1^+, E^+) = \frac{P(E^+) \cdot P(H_1^+|E^+)}{P(E^+) + P(H_1^+) - P(H_1^+ \cap E^+)} \approx 0.5333$. ∘

## 7 Evaluation

Our preliminary experimental performance evaluation of the implemented iSeM 1.0 is restricted to semantic signature matching, since the otherwise required




| | | AvgP | |
|---|---|---|---|
| | | $TC_{SH}$ | $TC_{LFN/FP}$ |
| *Basic Variants* | Logic | 0.41 | 0.06 |
| | Text Sim | 0.74 | 0.56 |
| | Struct | 0.65 | 0.58 |
| | Alogic | 0.57 | 0.46 |
| *SVM* | Logic + Text + Struct | 0.76 | |
| | All w/o pruning | 0.78 | |
| | All w. pruning | **0.82** | |

**Fig. 3.** Macro-averaged recall/precision (MARP) and average precision (AvgP) of basic and adaptive signature matching by iSeM 1.0

standard service retrieval test collection for IOPE-based matching does not exist yet[6]. For evaluation, we used the public tool SME2 v2.1[7] and the subset $TC_{\mathcal{SH}}$ of services in OWLS-TC3 annotated in $\mathcal{SH}$.

In summary, the evaluation results shown in Figure 3 reveal that (a) approximated logical matching via abduction and informative quality can perform significantly better than its strict logical counterpart, (b) performs closer to but still worse than its non-logic-based approximated counterparts (text and structural matching), and (c) adaptive hybrid combination outperforms all other variants in terms of precision. The first two findings can be directly derived from the MARP graph and the AvgP values shown in Figure 3.

As expected, due to the redundance of strict and approximated logical signature matching positives approximated logic-based matching alone was not able to outperform its non-logic-based counterparts but performed better than strict logical matching only in this respect. However, additional evaluation restricted to $TC_{LFN/FP} \subset TC_{\mathcal{SH}}$ that only contains cases of false positives and false negatives of strict logical signature matching indicated that, according to the statistical Friedman Test, none of the tested matching variants performed significantly better than the others at 5% level. This implies that each of the basic signature matching filters of iSeM contributes to an overall increase of

[6] The public standard test collections OWLS-TC3 for OWL-S and SAWSDL-TC2 for SAWSDL at semwebcentral.org contain services with semantic signatures only.

[7] http://projects.semwebcentral.org/projects/sme2/

performance for some cases of strict logical false classification, i.e. none of the tested variants outperformed the others for almost all service requests in the test collection.

The adaptive hybrid aggregation of the four different semantic signature matching filters as done by iSeM (cf. Section 6) significantly increases the retrieval performance compared to that of its individual matching filters. While the combination of strict logic-based, text similarity and structure matching already yields good results, the additional consideration of approximated logical matching (in the learning process) performs even if only slightly better. Finally, the service retrieval by iSeM based on the use of approximated logical matching for pruning the feature space of its semantic relevance learning (cf. Section 6.2) performed best.

## 8 Related Work

iSeM is the first adaptive, hybrid semantic service IOPE matchmaker, and there are quite a few other matchmakers available [11,8]. For example, the strict logical and the non-logic-based semantic signature matching filters as well as the SVM-based learning process of iSeM are adopted from the adaptive signature matchmaker OWLS-MX3 [9]. However, unlike iSeM, OWLS-MX3 neither performs approximated logical signature matching, nor PE-matching, nor is its adaptive process applicable to IOPE matching results and the feature space is not evidentially pruned. The same holds for the adaptive hybrid semantic signature matchmaker SAWSDL-MX2 [10]. Besides, SAWSDL-MX2 performs structural matching on the WSDL grounding level only which significantly differs from the semantic structural matching performed by iSeM. The use of abduction for approximated logical signature matching is inspired by DiNoia et al. [3,2]. However, their non-adaptive matchmaker MaMaS performs abduction for approximated matching of monolithic service concept descriptions in $\mathcal{SH}$, while iSeM exploit it for significantly different approximated structured signature matching and its use for learning. Besides, MaMaS has not been evaluated yet.

## 9 Conclusion

We presented the first adaptive, hybrid and full semantic service profile (IOPE) matchmaker that, in particular, performs approximated logical reasoning and respectively evidential coherence-based pruning of learning space to improve precision over strict logical matching. The preliminary evaluation of iSeM revealed, among others, that its adaptive hybrid combination with non-logic-based approximated signature matching improves each of them individually. The approximated logical matching results by iSeM can also be exploited for explanation-based interaction with the user during the selection process, if required, though in its initially implemented version iSeM remains non-obtrusive in this respect. Such interaction together with extending the abductive approximated reasoning to OWL2-DL annotations is future work.

[8] See also S3 contest in 2009: http://www.dfki.de/ klusch/s3/html/2009.html

## References


1. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
2. Colucci, S., et al.: Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an E-Marketplace. Electronic Commerce Research and Applications 4(4) (2005)
3. Di Noia, T., Di Sciascio, E., Donini, F.M.: A Tableaux-based Calculus for Abduction in Expressive Description Logics: Preliminary Results. In: Proceedings of 22nd International Workshop on Description Logics (DL) (2009)
4. Glass, D.H.: Inference to the Best Explanation: A comparison of approaches. In: Proceedings of the AISB 2009, Convention, Edinburgh, UK (2009), www.aisb.org.uk/convention/aisb09/Proceedings/
5. Hau, J., Lee, W., Darlington, J.: A semantic similarity measure for semantic web services. In: Proceedings of Web Service Semantics Workshop at WWW conference (2005)
6. Idestam-Almquist, P.: Generalization of Clauses under Implication. Artificial Intelligence Research 3, 467–489 (1995)
7. Keerthi, S.S., Lin, C.J.: Asymptotic behaviour of support vector machines with Gaussian kernel. Neural Computation 15(7), 1667–1689 (2003)
8. Klusch, M., Fries, B., Sycara, K.: OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S Services. In: Web Semantics, vol. 7(2). Elsevier, Amsterdam (2009)
9. Klusch, M., Kapahnke, P.: OWLS-MX3: An Adaptive Hybrid Semantic Service Matchmaker for OWL-S. In: Proceedings of 3rd International Workshop on Semantic Matchmaking and Resource Retrieval (SMR2), USA, CEUR 525 (2009)
10. Klusch, M., Kapahnke, P., Zinnikus, I.: Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL Analyzer. 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. 550–564. Springer, Heidelberg (2009)
11. Klusch, M.: Semantic Web Service Coordination. In: Schumacher, M., Helin, H., Schuldt, H. (eds.) CASCOM - Intelligent Service Coordination in the Semantic Web, ch. 4. Birkhuser Verlag, Springer (2008)
12. Klusch, M., Xing, Z.: Deployed Semantic Services for the Common User of the Web: A Reality Check. In: Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC), Santa Clara, USA. IEEE Press, Los Alamitos (2008)
13. Li, Y., Bandar, A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. Transactions on Knowledge and Data Engineering 15, 871–882 (2003)
14. Lin, D.: An Information-Theoretic Definition of Similarity. In: Proceedings of the 15th International Conference on Machine Learning, USA (1998)
15. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. Web Semantics 3(1), 41–60 (2005)
16. Robinson, J.A.: A Machine-Oriented Logic Based on the Resolution Principle. Association for Computing Machinery 12(1), 23–41 (1965)
17. Sycara, K., Widoff, S., Klusch, M., Lu, J.: LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. In: Autonomous Agents and Multi-Agent Systems, vol. 5, pp. 173–203. Kluwer, Dordrecht (2002)
18. Samuel, K., et al.: Translating OWL and semantic web rules into prolog: Moving toward description logic programs. Theory and Practice of Logic Programming 8(03), 301–322 (2008)
19. Scheffer, T., Herbrich, R., Wysotzki, F.: Efficient theta-Subsumption based on Graph Algorithms. In: ILP 1996. LNCS, vol. 1314. Springer, Heidelberg (1997)

# Measures for Benchmarking Semantic Web Service Matchmaking Correctness

Ulrich Küster and Birgitta König-Ries

Institute of Computer Science, Friedrich-Schiller-University Jena
D-07743 Jena, Germany
{Ulrich.Kuester,Birgitta.Koenig-Ries}@uni-jena.de

**Abstract.** Semantic Web Services (SWS) promise to take service oriented computing to a new level by allowing to semi-automate time-consuming programming tasks. At the core of SWS are solutions to the problem of SWS matchmaking, i.e., the problem of filtering and ranking a set of services with respect to a service query. Comparative evaluations of different approaches to this problem form the base for future progress in this area. Reliable evaluations require informed choices of evaluation measures and parameters. This paper establishes a solid foundation for such choices by providing a systematic discussion of the characteristics and behavior of various retrieval correctness measures in theory and through experimentation.

## 1 Introduction

In recent years, Semantic Web Services (SWS) research has emerged as an application of the ideas of the Semantic Web to the service oriented computing paradigm. The grand vision of SWS is to have a huge online library of component services available, which can be discovered and composed dynamically based upon their formal semantic annotations. One of the core problems in the area concerns SWS matchmaking, i.e., the problem of filtering and ranking a set of services with respect to a service query. A variety of competing approaches to this problem has been proposed [1]. However, the relative strengths and shortcomings of the different approaches are still largely unknown. For the future development of the area it is thus of crucial importance to establish sound and reliable evaluation methodologies.

Evaluations in the area typically follow the approach taken in the evaluation of Information Retrieval (IR) systems: As a basis for the evaluation a test collection is provided. This collection contains a number of service offers, a (smaller) number of service requests and relevance judgments. These relevance judgments are provided by human experts and specify for each offer-request pair how relevant the offer is for the request, i.e., whether or to which degree the offer is able to satisfy the request. Matchmakers are then evaluated by comparing their output rankings with the one induced by the relevance judgments. This is done via retrieval correctness measures which assign an output ranking a performance score based upon the available reference relevance judgments.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 45–59, 2010.

While the general procedure is agreed upon, there has been little work up to now that investigates the influence of different settings on the stability and meaningfulness of the evaluation results. For instance, relevance judgments can be binary (relevant versus irrelevant) or graded (multiple levels of relevance), they may be subjective and different ways to deal with conflicting judgments are possible. Furthermore, a variety of evaluation measures with very different characteristics are available from IR.

Therefore, informed decisions about the evaluation measures employed, the underlying model of relevance and the procedure of how to obtain reliable relevance judgments are necessary for meaningful evaluations. In previous work we dealt extensively with the latter two issues [2]. We also presented a preliminary work discussing the applicability of different evaluation measures from IR to the SWS matchmaking domain [3]. In this paper, we extend this work by providing a comprehensive discussion of retrieval correctness measures in the domain of SWS matchmaking in theory and through experimentation.

The rest of the paper is structured as follows. In the following section, we provide an overview of related work in the area. Section 3 defines requirements to evaluation measures and discusses measures common in IR with respect to those requirements. A number of issues are identified and solutions to these issues are proposed. Section 4 complements this theoretic treatment by an analysis of the behavior of the measures in practice, based upon data from a community benchmarking event we organized. In particular we investigate the effects of three factors to the evaluation results: changes in the underlying definition of relevance, inconsistent relevance judgments and the choice of evaluation measure. The paper concludes with recommendations for appropriate decisions on evaluation measures that will make future evaluations more meaningful and reliable.

## 2 Related Work

Experimental evaluation of SWS retrieval correctness has received relatively little attention in the past [4]. Almost all approaches have so far relied on binary relevance and standard measures based on precision and recall without further motivating this evaluation approach.

Tsetsos et al. [5] were the first to raise the issue that binary relevance may be too coarse grained for reliable SWS retrieval evaluations. They proposed to use a relevance scale based on fuzzy linguistic variables and the application of a fuzzy generalization of recall and precision that evaluates the degree of correspondance between the rating of a service by an expert and a system under evaluation. However, a systematic investigation of the properties of different measures was not within the scope of their work. Apart from the work by Tsetsos et al. we are not aware of any work directly dealing with evaluation measures for SWS retrieval correctness evaluation.

In contrast, there is a large body of related work from the area of Information Retrieval that concerns the development and discussion of evaluation measures, e.g., [6,7,8,9,10]. However, it is not clear to which extent findings about stability

and sensitivity of measures from IR transfer to the domain of SWS retrieval, since there are important fundamental as well as practical differences between SWS retrieval and general IR evaluation [2]. Furthermore, we are not aware of a previous systematic discussion of the properties of all the measures covered in this paper, in particular not with respect to what we will define below as measure correctness.

Our work is directly related to the S3 Contest on Semantic Service Selection[1], an annual campaign dedicated to the comparative evaluation of SWS matchmakers. The most recent 2009 edition introduced the usage of some graded retrieval performance measures in addition to standard binary recall and precision and we organized the experiment we will use to analyze measure behavior in practice as part of this contest. By providing a systematic discussion of the characteristics and behaviors of all common retrieval measures, this paper aims at providing the foundation for well-founded choices of parameters and measures for future SWS retrieval correctness evaluations.

## 3 Retrieval Effectiveness Measures

Service matchmakers in the context of this paper compare a service request with a set of available service offers and return a list of matching services, ordered by decreasing estimated relevance to the request. Retrieval effectiveness measures need to quantify the quality of the output lists produced by various matchmakers. The following definitions will be used throughout this paper.

**Definition 1 (Ranking).** *A* ranking r *of a set of services* S *is an ordered sequence of the elements from* S*, i.e.:* $r = (r_1, r_2, \ldots, r_n)$, $n <= |S|$, $r_i \in S$, $r_i = r_j \Rightarrow i = j$. *The number* i *is called the* rank *of the service* $r_i$ *with respect to the ranking* r*. A ranking with* $n = |S|$ *is called a* full ranking*.*

**Definition 2 (Gain).** *The* gain g *(*$g >= 0$*) of a service* s *with respect to a query* q *denotes the relevance of* s *to* q*. The function* $g_q$ *which assigns each service* s *from a ranking* r *a gain* g *with respect to a query* q *is called a* gain function*. We furthermore define a binary flag that denotes whether a service at a given rank* $i$ *is relevant or not:* $isrel_{r,q}(i) = 1$*, if* $g_q(r_i) > 0$ *and* 0 *otherwise.*

For the sake of simplicity, we will generally omit the query index $q$ and the ranking index $r$ in the following if the query or ranking under consideration is clear from the context or no specific query or ranking is referenced.

**Definition 3 (Ideal ranking).** *A full ranking* r *is called* ideal *iff it lists the services in decreasing order of relevance, i.e.:* $\forall i \in \{2..|S|\} : g(r_i) <= g(r_{i-1})$.

**Definition 4 (Retrieval effectiveness measure).** *A* retrieval effectiveness measure m *is a function which assigns a ranking* r *a value from* $[0, 1]$ *with respect to a gain function* g*:* $m_g(r) \rightarrow [0, 1]$.

[1] http://dfki.de/~klusch/s3/

Having introduced a basic notion of retrieval effectiveness measure, we now turn to defining desirable properties of such measures. Again, a few definitions are helpful.

**Definition 5 (Ranking superiority).** *A ranking* r *is called* superior *to a different ranking* r′ *with respect to a given gain function* g *(*$r > r'$*) iff* r′ *can be changed into* r *by a sequence of pair wise item swaps within* r′ *and for each two swapped items* $r_i$ *and* $r_j$ *from* $r'$ *it holds:* $i < j \Rightarrow g(r_i) < g(r_j)$ *(items with higher relevance are moved upwards).*

**Definition 6 (Measure correctness).** *A retrieval effectiveness measure* $m$ *is called* correct *iff for any two rankings* r *and* r′ *and a gain function* g, $r > r' \Rightarrow m_g(r) > m_g(r')$ *holds.*

Ranking superiority and measure correctness formalize the intuitive notion that a ranking that lists items of higher relevance at comparatively higher ranks should always receive a superior effectiveness measure score. Besides this notion of correctness, three more properties of retrieval measures are desirable.

First, performance measures should allow to be compared meaningfully over queries. To avoid normalization problems, we require that an ideal ranking always receives a performance score of 1. Second, for graded relevance, measures should allow to configure the extent to which an item of comparatively higher relevance is preferred over an item of comparatively lower relevance. Third, for typical retrieval tasks, performance at the beginning of the output ranking is more important than performance at the end of the output ranking since a user typically will not completely read through a long ranking till its end. A good retrieval measure should thus emphasize top rank performance over bottom rank performance and allow to configure the extent of this emphasis.

### 3.1 Retrieval Measures from IR

After having briefly discussed desirable properties of retrieval effectiveness measures, we now turn to recalling some well established measures from IR. A complete coverage is beyond the scope of this paper, but available in [11,12].

IR retrieval effectiveness measures are almost exclusively based upon the well-known *Recall* and *Precision* measures. Let $R$ be the set of relevant items for a query and $L$ be the set of the first $l$ items returned in response to that query. Then $Recall_l$ is defined as the proportion of all relevant items that are contained in $L$ and $Precision_l$ as the proportion of items in $L$ that are relevant:

$$Recall_l = \frac{L \bigcap R}{R}, \quad Precision_l = \frac{L \bigcap R}{L}.$$

Precision can then be measured as a function of Recall by scanning the output ranking from the top to the bottom and observing the Precision at standard Recall levels. These measures average well for different queries and the corresponding R/P charts are the most widely used measure to compare the retrieval

performance of systems. If a system's performance needs to be captured in a single measure, the common one is *Average Precision* over relevant items:

$$AveP = \frac{1}{|R|} \sum_{i=1}^{|L|} isrel(i) \frac{\sum_{j=1}^{i} isrel(j)}{i}.$$

Historically, IR evaluation has primarily been based on binary relevance [11]. However, since about 2000, there is an increased interest in measures based on graded or continuous relevance [12,9]. Various proposals have been made to generalize the Recall and Precision based measures from binary to graded relevance. We briefly recall the most common ones.

All of them are based on or can be expressed in terms of *Cumulated Gain* proposed by Järvelin and Kekäläinen [7]. Intuitively, Cumulated Gain at rank $i$ measures the gain that a user receives by scanning the top $i$ items in a ranked output list. More formally, the *Cumulated Gain* at rank $i$ is defined as $CG(i) = \sum_{j=1}^{i} g(r_j)$. Moreover, the *Ideal Cumulated Gain* at rank $i$, $ICG(i)$, refers to the cumulated gain at rank $r$ of an ideal ranking. This allows to define the *Normalized Cumulated Gain* at rank $i$ as the retrieval performance relative to the optimal retrieval behavior: $NCG(i) = \frac{CG(i)}{ICG(i)}$.

Normalized Cumulated Gain allows a straightforward extension of AveP which has sometimes been referred to as *Average Weighted Precision* [6]:

$$AWP = \frac{1}{|R|} \sum_{i=1}^{|L|} isrel(i) \frac{CG(i)}{ICG(i)}.$$

Unfortunately, NCG(i) has a significant flaw that AWP inherits. ICG(i) has a fixed upper bound ($ICG(i) <= ICG(|R|)$). Thus, NCG(i) and AWP cannot penalize late retrieval of relevant items properly since NCG(i) cannot distinguish at which rank relevant documents are retrieved for ranks greater or equal than $|R|$ [6]. Several measures have been proposed that resolve this flaw of AWP.

Järvelin and Kekäläinen [7] suggested to use a discount factor to penalize late retrieval and thus reward systems that retrieve highly relevant items early. They defined *Discounted Cumulated Gain* at rank $i$ as $DCG(i) = \sum_{j=1}^{i} \frac{g(i)}{disc(i)}$ with $disc(i) >= 1$ being an appropriate discount function. Järvelin and Kekäläinen suggested to use the log function and use its base $b$ to customize the discount which leads to

$$DCG_{\log_b}(i) = \sum_{j=1}^{i} \frac{g(i)}{max(1, \log_b i)}.$$

An according definition of *Ideal Discounted Cumulated Gain* ($IDCG(r)$) can be used to define the *Normalized Discounted Cumulated Gain* at some document cutoff level $l$ ($NDCG_l$) and a straightforward Version of AWP that we call *Average Weighted Discounted Precision*: $AWDP = \frac{1}{|R|} \sum_{i=1}^{|L|} isrel(i) \frac{DCG(i)}{IDCG(i)}$.

Kishida [9] proposed a generalization of AveP that also avoids the flaw of AWP:

$$GenAveP = \frac{\sum_{i=1}^{|L|} isrel(i)\frac{CG(i)}{i}}{\sum_{i=1}^{|R|}\frac{ICG(i)}{i}}.$$

Sakai [6] proposed an integration of AWP and AveP called Q-measure which inherits properties of both measures and possesses a parameter $\beta$ to control whether Q-measure behaves more like AWP or more like AveP:

$$Q\text{-}measure = \frac{1}{|R|}\sum_{i=1}^{|L|} isrel(i)\frac{\beta CG(i) + \sum_{j=1}^{i} isrel(j)}{\beta ICG(i) + i}.$$

Finally, it has also been proposed to use Kendall's $\tau$ or other rank correlation measures to measure retrieval effectiveness by comparing a ranking $r$ with an ideal ranking $r'$ [13]. Kendall's $\tau$ measures the correlation between two rankings via the number of pair wise adjacent item swaps that are necessary to turn one ranking into another. Since Kendall's $\tau$ yields values between 1 (identical rankings) and -1 (inverse rankings), it needs to be normalized to yield values from $[0, 1]$: $\tau'(r) = \frac{\tau(r,r')+1}{2}$.

### 3.2 Discussion of Measures

With the exception of $\tau'$ and AveP, all measures introduced above allow fine-tuning the extent to which highly relevant items are preferred over less relevant items by choosing an appropriate gain function. Furthermore, except for $\mathrm{CG}_l$ and $\mathrm{DCG}_l$ all measures are properly normalized and assign an ideal ranking a score of 1. We now discuss the measures with respect to correctness and the degree of control over the extent to which late retrieval is penalized. For illustration, please consider the rankings displayed on the left side in Table 1. The numbers in the rankings represent gain values or corresponding items to be retrieved. The right side of the table provides the performance scores that various measures assign to the given rankings. Please observe that $R_1$ is the optimal ranking and that $R_1 > \{R_2, R_3\} > R_4 > R_5 > R_6 > R_7$. Furthermore, $R_2$ should be considered preferable to $R_3$ since the single item swap compared to the optimal ranking occurs at lower ranks than is the case with $R_2$. These relations should be reflected in the performance scores assigned by the measures. This is not always the case as will be discussed below.

*AveP:* Trivially, binary AveP can not distinguish among items of different relevance grades and is thus not correct for graded relevance: $AveP(R_1) = AveP(R_2)$.

*NDCG:* $\mathrm{NDCG}_l$ is correct, if the used discount function is valid, i.e., positive and strictly monotonic increasing for $i \in [1, l]$. Notably, this is not the case for the originally suggested and typically used $max(1, \log_b(i))$ discount function which is constant for ranks 1 through $b$. With valid discounting functions (e.g., $\sqrt{i}$), however, $\mathrm{NDCG}_l$ is correct as far as rankings are only considered up to rank $l$.

**Table 1.** Comparison of evaluation measures

$R_1 = (10, 6, 3, 0, 0, 0, 0, 0, 0, 0)$
$R_2 = (10, 3, 6, 0, 0, 0, 0, 0, 0, 0)$
$R_3 = (6, 10, 3, 0, 0, 0, 0, 0, 0, 0)$
$R_4 = (3, 6, 10, 0, 0, 0, 0, 0, 0, 0)$
$R_5 = (0, 0, 0, 3, 6, 10, 0, 0, 0, 0)$
$R_6 = (0, 0, 0, 0, 0, 10, 6, 3, 0, 0)$
$R_7 = (0, 0, 0, 0, 0, 0, 10, 6, 3)$

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $R_6$ | $R_7$ |
|---|---|---|---|---|---|---|---|
| $AveP$ | 1.00 | 1.00 | 1.00 | 1.00 | 0.38 | 0.28 | 0.24 |
| $NDCG_9(\sqrt{i})$ | 1.00 | 0.98 | 0.93 | 0.81 | 0.52 | 0.46 | 0.43 |
| $AWP$ | 1.00 | 0.94 | 0.87 | 0.62 | 0.54 | 0.79 | 0.79 |
| $Q\text{-}measure(\beta = 1)$ | 1.00 | 0.94 | 0.88 | 0.66 | 0.50 | 0.65 | 0.63 |
| $GenAveP$ | 1.00 | 0.94 | 0.84 | 0.57 | 0.23 | 0.26 | 0.23 |
| $AWDP(\sqrt{i})$ | 1.00 | 0.94 | 0.81 | 0.54 | 0.29 | 0.37 | 0.35 |
| $\tau'$ | 1.00 | 0.97 | 0.97 | 0.92 | 0.67 | 0.58 | 0.50 |

$\text{NDCG}_l$ also allows configuring the extent to which late retrieval is penalized by choosing a more or less quickly growing discount function.

*AWP:* As mentioned above, AWP can not differentiate among rankings that are equal till rank $|R|$, e.g., $AWP(R_6) = AWP(R_7)$. Even worse, the order of items may matter more than their absolute rank, e.g., $AWP(R_5) < AWP(R_6)$, despite of $R_5 > R_6$. AWP is thus not correct. To the best of our knowledge, this order versus rank defect has not been discussed so far. AWP also does not allow configuring the extent to which late retrieval is penalized.

*Q-Measure:* Q-Measure was designed to resolve the first defect of AWP, but unfortunately inherits the second one, e.g., $Q\text{-}measure(R_5) < Q\text{-}measure(R_6)$. The actual vulnerability of Q-Measure to this defect depends upon the actual choices for the gain values and its $\beta$ factor. But for any setting, it either inherits the vulnerability from AveP of not properly distinguishing among items of varying relevance or the order versus rank defect from AWP and is thus not correct. Q-Measure provides limited control over the extent to which late retrieval is penalized via its $\beta$ factor.

*GenAveP:* GenAveP shares the order versus rank defect with Q-Measure and AWP, e.g., $GenAveP(R_5) < GenAveP(R_6)$. Therefore, just like Q-Measure and AWP, it is not correct. However, in practice, GenAveP seems to be somewhat less vulnerable to the mentioned defects than the other two measures. GenAveP does not allow configuring the extent to which late retrieval is penalized.

*AWDP:* AWDP resolves the first defect of AWP if the used discounting function is valid. Nevertheless it inherits the order versus rank defect from AWP, e.g., $AWDP_{\sqrt{i}}(R_5) < AWDP_{\sqrt{i}}(R_6)$. It is therefore also not correct. Like the choice of $\beta$ for Q-Measure, the choice of a discount function for AWDP has an influence on its practical vulnerability to this particular defect. By choosing a proper discount function AWDP allows configuring the extent to which late retrieval is penalized.

*Rank Correlation Measures:* Kendall's $\tau$, respectively $\tau'$, is correct in the sense provided above. However, it does not differentiate between swaps that occur at

the top and those that occur at the bottom of a ranking, e.g., $\tau(R_2) = \tau(R_3)$. Furthermore, as mentioned above, it also does not allow to configure the extent to which highly relevant items are preferred over less relevant ones.

*Summary:* It is remarkable, that, as can be seen from this discussion, with the exception of NDCG and Kendall's $\tau$ all commonly used evaluation measures based on graded relevance are not correct in the sense defined above. Furthermore, NDCG is typically used with a discount function that renders it effectively incorrect, too, and Kendall's $\tau$ lacks the ability of emphasizing top versus bottom rank performance and configuring the extent to which highly relevant items are preferred over marginally relevant ones.

### 3.3 Proposed Improvements

After having discussed shortcomings of most commonly used measures for graded relevance, we now propose improvements to avoid these shortcomings. Table 2 shows a comparison of the original with the altered versions of the measures that illustrates how the altered versions avoid the problems of the original ones: in contrast to the scores of AWP, GenAveP and AWDP, those of ANCG, GenAveP′ and ANDCG are strictly decreasing from $R_1$ to $R_7$.

*NDCG:* The issues with NDCG can be trivially avoided by using an adapted version of the original discount function, namely $disc(i) = \log_b(i + b - 1)$, or any other valid function, like a root function, i.e., $disc(i) = i^a, 0 < a <= 1$. Such obvious adaptations have been proposed previously, e.g., [14]. Therefore, it is somewhat surprising to see that most literature still uses the original flawed discounting functions, e.g., [8].

*AWP/AWDP and GenAveP:* The defects of AWP/AWDP and GenAveP can be avoided by not averaging over relevant items only, but over all items, i.e.:

$$AW(D)P' = \frac{1}{|R|} \sum_{i=1}^{|L|} \frac{(D)CG(i)}{I(D)CG(i)}, \quad GenAveP' = \frac{\sum_{i=1}^{|L|} \frac{CG(i)}{i}}{\sum_{i=1}^{|R|} \frac{ICG(i)}{i}}.$$

AW(D)P' can be interpreted as the area under a N(D)CG-chart [7]. To properly distinguish the altered from the original versions, we will refer to the altered ones as *Averaged Normalized Cumulated Gain (ANCG)* and *Averaged Normalized Discounted Cumulated Gain (ANDCG)* in the following.

*Others:* In contrast to the previous measures, Q-Measure can not be fixed in the same fashion. Averaging over all, and not only relevant items, decreases the performance value of the AveP part of Q-Measure to values much smaller than 1.0 even for optimal rankings if the number of relevant items is much smaller than the total number of items. This makes averaging of results over queries with differing numbers of relevant items unstable.

**Table 2.** Comparison of altered evaluation measures

| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $R_6$ | $R_7$ |
|---|---|---|---|---|---|---|---|
| $AWP$ | 1.00 | 0.94 | 0.87 | 0.62 | 0.54 | 0.79 | 0.79 |
| $ANCG$ | 1.00 | 0.98 | 0.96 | 0.87 | 0.51 | 0.37 | 0.26 |
| $GenAveP$ | 1.00 | 0.94 | 0.84 | 0.57 | 0.23 | 0.26 | 0.23 |
| $GenAveP'$ | 1.00 | 0.97 | 0.91 | 0.76 | 0.30 | 0.20 | 0.13 |
| $AWDP$, $disc(i) = \sqrt{i}$ | 1.00 | 0.94 | 0.81 | 0.54 | 0.29 | 0.37 | 0.35 |
| $ANDCG$, $disc(i) = \sqrt{i}$ | 1.00 | 0.96 | 0.89 | 0.72 | 0.27 | 0.18 | 0.12 |

Similarly the issues with Kendall's $\tau$ can also not been fixed easily. Rank correlation measures are not designed to distinguish between whether rankings differ at the top or bottom. Furthermore, rank correlation does not offer an intuitive way of configuring the extent to which highly relevant items are preferred over less relevant items.

### 3.4 Conclusions

The discussion above has shown that various measures for graded relevance are available, but that even some of the common ones behave unintuitively in certain cases. A fix for the problems associated with AWP, AWDP and GenAveP has been proposed. With this fix, NDCG, ANCG (fixed AWP), ANDCG (fixed AWDP) and GenAveP' (fixed GenAveP) are correct as defined above.

While this correctness guarantees a ranking of matchmakers which corresponds to intuition if the matchmaker's output rankings are pair wise superior, it does not guarantee a good ranking of matchmakers that produce outputs that are not pair wise superior, the common case in realistic settings. For such rankings, there is no objective notion of superiority, since a decision has to be made how to balance highly against less relevant items and performance in top against that in lower ranks (or recall versus precision for that matter).

The following Section 4 will thus complement the already presented discussion by an investigation of the behavior of the covered measures based on real rankings in a realistic retrieval experiment. Based upon this investigation, recommendations for retrieval effectiveness measures will be provided in Section 5.

## 4 Analysis of Measure Behavior in Practice

We organized an evaluation of semantic service matchmakers across formalisms as part of the 2009 S3 Contest on Semantic Service Selection. Full information about this evaluation campaign, its setup and results is available online[2]. Due to space restrictions, we will only provide a brief introduction to the setup of the evaluation before discussing the characteristics of retrieval correctness measures based upon the data gathered from the evaluation.

[2] http://fusion.cs.uni-jena.de/professur/jgdeval

*Goals:* The evaluation targets the use case of a human developer that is searching for a Web service that provides a functionality needed in some application being developed. Semantic service matchmakers are expected to make this discovery process more efficient by providing efficient filtering and ranking of services in registries. The task being evaluated is thus to rank a list of given Web services with respect to their relevance to given user queries. Relevance is defined by reference judgments from human experts (see evaluation parameters below).

*Data Set:* For the evaluation, a data set of real services with rich information was needed. Furthermore, to make the retrieval task challenging, a large number of related by slightly different services was desired. Existing data sets did not meet these requirements in an ideal way [15]. Thus, the Jena Geography Dataset (JGD) was created[3] [2]. This data set consists of 200 real service operations from the geography domain which have been collected with all the information available online, i.e. all the information that a human developer finds when searching these services.

*Experimental Setup:* The experiment was executed in multiple phases. In the first phase, services were released to participating groups and the particants provided (semantic) annotations for the services in a way that they felt most suitable for their needs and matchmakers. Unfortunately, participants were overcharged by annotating the full 200 services and the dataset had to be reduced to 50 services. In a second phase, nine requests were released. Relevance judgments were not released together with the requests and the participating groups were asked to have members formalize the queries who had not been involved in the annotation of the services previously.

Finally, participants had to provide an implementation of their matchmaking system, pluggable to the SWS Matchmaker Evaluation Environment (SME2)[4] via a predefined interface. After services, queries and ontologies had been collected, the matchmakers were installed on a dedicated machine. SME2 was used to execute the evaluation, i.e., send queries to the registered matchmakers and retrieve the returned service rankings.

Five groups participated with six matchmakers (Themis-S, WSColab, IRS-III, SAWSDL-MX1/MX2, SAWSDL iMatcher) in the experiment, representing a variety of approaches from NL processing via folksonomy tagging to the usage of logic semantic annotations of services. Details on these matchmakers can be found online. Furthermore, we added the average performance of 50 random service rankings to the results as a performance bottom line.

*Evaluation Parameters:* Relevance judgments for the JGD have been created according to a multi-dimensional graded relevance scale which differentiates among the interface compatibility, the functional completeness and the functional equivalence of services. The judgments have been created by three judges independently. Afterwards, consensus judgments were built by debating judgments that

[3] http://fusion.cs.uni-jena.de/professur/jgd

[4] http://projects.semwebcentral.org/projects/sme2/

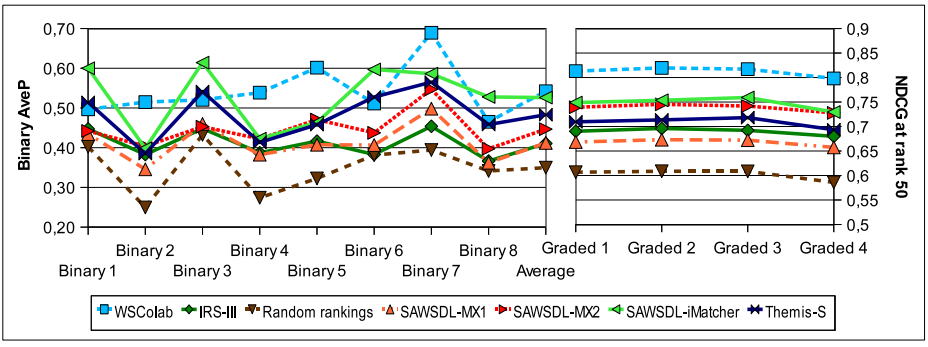



**Fig. 1.** Sensitivity of binary AveP and $NDCG_{50}$ to changes in the relevance definition

differed among judges. Again, full information is available online and in previous work [2]. The measures described in Section 3 allow evaluating SWS retrieval systems based on graded relevance but leave open the question about the proper parameter combinations to use in an evaluation. In order to investigate the effects of different definitions of relevance, we used four different gain value settings for the graded relevance and eight different definitions of binary relevance, i.e. different ways how to reduce the multi-dimensional graded relevance judgments to binary ones.

### 4.1 Influence of Relevance

We now turn to analyzing the characteristics of the discussed retrieval correctness measures and start with the effect of changes in the relevance definition to the evaluation results. We concentrate on the question whether a measure correctly orders the matchmakers by their retrieval effectiveness and is not influenced by other factors not of interest and under control during the evaluation.

Figure 1 illustrates the sensitivity (changes in the relative order of evaluated matchmakers) of binary AveP and $NDCG_{50}$ with discount $\log_2(i+1)$ to changes in the relevance definition. It highlights drastic swaps in the relative performance of the evaluated matchmakers if binary relevance is used (left side). The usage of Binary 2 compared to Binary 3, for instance, results in largely different evaluation results. These findings are in line with similar studies from IR, e.g. [16].

In contrast, measures based on graded relevance are almost entirely stable against moderate changes in the gain values. Using $NDCG_{50}$ with discount $\log_2(i+1)$, for instance, there was not a single swap in matchmaker order for the four different graded relevance settings (right side). This finding is, again, in line with similar findings from the IR community [8]. Nevertheless the amount of difference in stability is remarkable. At least our test data makes a very strong case for preferring graded over binary relevance for the given evaluation use case.

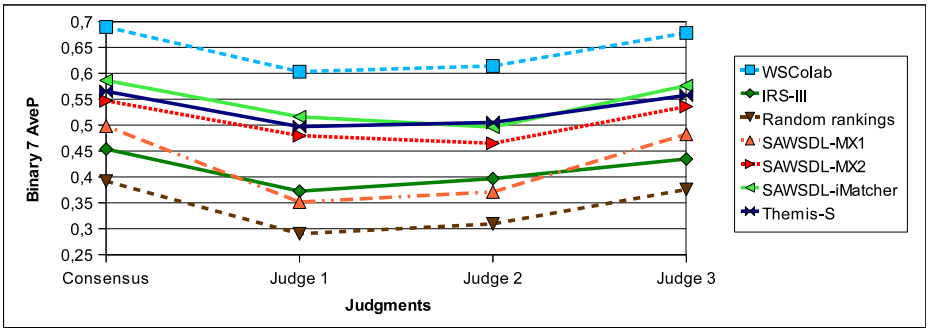



**Fig. 2.** Sensitivity of AveP to inconsistent relevance judgments (Binary 7 relevance)

### 4.2 Influence of Relevance Judge

It is well known from IR, that relevance judgments for retrieval evaluations differ among judges and for the same judge at different times [10]. In previous work, we investigated this issue in depth in the context of relevance judgments for service retrieval evaluation. We found significant inconsistency in judgments in this domain, too [2]. We are now able to complement the corresponding discussion by analyzing the effect that judgments by different judges really have on the comparative evaluation results.

Figure 2 show the computed AveP scores for the most liberal binary relevance setting using the consensus judgments as well as the original ones obtained from each of the three judges. The figure illustrates that changes in rankings, even notable ones, do occur but also that the influence is much smaller than that of switching the definition of relevance. Again, graded relevance (not shown in the figure) was more stable than binary relevance. However, with the exception of $NDCG_{50}$, swaps in rankings occurred occasionally using graded relevance, too.

### 4.3 Influence of Evaluation Measure

Finally, we now turn to discussing the influence of the choice of evaluation measure to the evaluation results. We consider NDCG, ANDCG, AWDP, ANCG, AWP, Q-Measure ($\beta \in \{0.5, 1, 2\}$), AveP, GenAveP and GenAveP'. The measures including a discount are analyzed using $\sqrt{i}$, $\log_2(i+1)$, $\log_3(i+2)$ and $\log_5(i+4)$ as discount function.

Figure 3 shows the performance scores from these measures using the Graded 1 relevance setting (AveP is computed assuming all services with a positive gain as relevant). The figure illustrates that the choice of evaluation measure influences evaluation results and also demonstrates the issues discussed in theory in Section 3.

As can be seen, there is a drastic difference in measure behavior between the incorrect AWP and its fixed counterpart ANCG. Please recall, that AWP had two defects. First, its inability to punish very late retrieval, second, its property

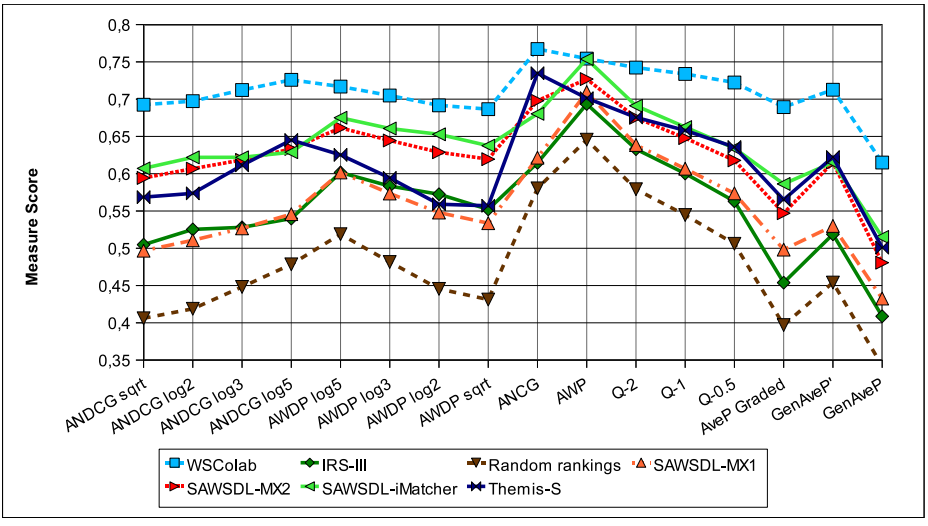



**Fig. 3.** Comparison of graded evaluation measures

of rewarding correct order of relevant items rather than their absolute ranks. Themis-S is the only matchmaker whose score declines when switching from ANCG to AWP. This can be primarily explained through the first AWP defect, since Themis-S is inferior at retrieving highly relevant items at the top ranks, but superior at retrieving all relevant items relatively soon (see detailed evaluation results online). The first characteristic is correctly punished by both measures, whereas the second is not rewarded by AWP.

However, Themis-S is also evaluated comparatively poorly by the AWDP measures, which suffer from the order versus rank defect, but not from the inability of properly punishing late retrieval. This bias of AWDP against Themis-S is particularly evident by comparing AWDP and its correct counterpart ANDCG (please note that $NDCG_{50}$ rated equal to ANDCG and was thus not included in the chart). A comparison of scores within the different versions of ANDCG and AWDP illustrates nicely the effect of discounting. Stronger discounting comparatively benefits IRS-III whereas Themis-S profits from smaller discounts. This is an expected behavior and results from IRS-III performing a precision oriented matchmaking versus the recall oriented matchmaking of Themis-S. The fact that stronger discounting penalizes Themis-S is another argument for the bias of AWDP against Themis-S being caused by the order versus rank defect and not an insufficient punishment of late retrieval.

It is notable, that Q-Measure and GenAveP, which also suffer from the order versus rank defect, are less biased against Themis-S than AWDP. Still, the fixed GenAveP' and Q-Measure with a small $\beta$ are more favorable for Themis-S than the incorrect GenAveP and Q-Measure with a larger $\beta$.

## 5 Summary and Conclusions

This paper dealt with measures for evaluating the retrieval correctness of SWS matchmakers. To the best of our knowledge it is the first such work in the area of SWS retrieval. Desireable properties of evaluation measures were defined and various measures from IR introduced. Properties of these measures were first discussed in theory. Defects in some measures were identified and fixes for these defects proposed. Finally, the theoretic discussion was complemented by an experimental investigation of measure behavior in practice. From the discussion and experimental analysis, some important conclusions for future retrieval effectiveness evaluations may be derived.

First, binary AveP is highly sensitive towards changes in the definition of relevance underlying the relevance judgments. Unless one knows about this definition very well, is certain that the definition matches the use case of the evaluation and that the reference judges applied the definition correctly, we recommend against using binary relevance in the future. In contrast, graded relevance is extremely stable against moderate changes in the gain values (and thus the underlying definition of relevance) and therefore should be preferred over binary relevance.

Second, inconsistency in relevance judgments influences evaluation results, but only moderately. Again, binary relevance is less stable than graded relevance. Obviously, more reliable judgments are preferable, but the effects of inconsistency seem to remain in a tolerable range, at least for graded relevance.

Third, the choice of evaluation measure influences the evaluation results. The choice of a graded measure has less influence than the choice of relevance with binary AveP, but more influence than inconsistent judgments. To obtain reliable evaluation results, one should not choose a particular measure without justifying the choice. For a fair and unbiased treatment, analysis with different measures and corresponding reporting is recommended. Contradicting measures indicate differing retrieval characteristics of the matchmakers exchanging ranks and thus allow tracing those characteristics. Corresponding insights are an important additional advantage of using different evaluation measures.

Fourth, as was suggested before, AWP is not a reliable evaluation measure because of its inability to properly punish late retrieval. However, Q-Measure, GenAveP and in particular AWDP may also show an unintuitive measure behavior. The alternative $NDCG_l$ and the newly proposed ANCG/ANDCG are correct with respect to the definition provided in Section 3 and offer the most intuitive and flexible way of customizing the emphasis on top over bottom ranks. These measures are recommended for future retrieval effectiveness evaluations. NDCG charts are probably the most informative way of presenting evaluation results, since they provide an indication of the performance of matchmakers over ranks and still provide a summary measure by the value at the bottom rank ($NDCG_{50}$ in our case). Attention should be paid to choosing a valid discount function for this measure.

We hope that these findings will help establishing sound evaluation methodologies and further advancing the state of the art in SWS matchmaking.

***Acknowledgments.*** We owe great thanks to Patrick Kapahnke and Matthias Klusch from DFKI Saarbrücken for providing the SME2 tool and performing the

actual execution of the evaluation on their machines. Additionally we would like to thank all participants in Track 3 of the 2009 S3 Contest for all their efforts without which this work would not have been possible.

## References

1. Klusch, M.: Semantic web service coordination. In: Schumacher, M., Helin, H. (eds.) CASCOM - Intelligent Service Coordination in the Semantic Web. Springer, Heidelberg (2008)
2. Küster, U., König-Ries, B.: Relevance judgments for web services retrieval - a methodology and test collection for sws discovery evaluation. In: Proc. of the 7th IEEE European Conference on Web Services (ECOWS 2009), Eindhoven, The Netherlands (2009)
3. Küster, U., König-Ries, B.: Evaluating semantic web service matchmaking effectiveness based on graded relevance. In: Proc. of the 2nd International Workshop $SMR^2$ on Service Matchmaking and Resource Retrieval in the Semantic Web at ISWC 2008 (2008)
4. Küster, U., König-Ries, B., Petrie, C., Klusch, M.: On the evaluation of semantic web service frameworks. International Journal on Semantic Web and Information Systems 4(4) (2008)
5. Tsetsos, V., Anagnostopoulos, C., Hadjiefthymiades, S.: On the evaluation of semantic web service matchmaking systems. In: 4th IEEE European Conference on Web Services (ECOWS 2006), Zürich, Switzerland (2006)
6. Sakai, T.: Ranking the NTCIR systems based on multigrade relevance. In: Revised Selected Papers of the Asia IR Symposium, Beijing, China, pp. 251–262 (2004)
7. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)
8. Sakai, T.: On the reliability of information retrieval metrics based on graded relevance. Information Processing and Management 43(2), 531–548 (2007)
9. Kishida, K.: Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. Technical Report NII-2005-014E, National Institute of Informatics, Tokyo, Japan (2005)
10. Saracevic, T.: Effects of inconsistent relevance judgments on information retrieval test results: A historical perspective. Library Trends 56(4), 763–783 (2008)
11. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)
12. Demartini, G., Mizzaro, S.: A classification of IR effectiveness metrics. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsikrika, T., Yavlinsky, A. (eds.) ECIR 2006. LNCS, vol. 3936, pp. 488–491. Springer, Heidelberg (2006)
13. Melucci, M.: On rank correlation in information retrieval evaluation. ACM SIGIR Forum 41(1), 18–33 (2007)
14. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.N.: Learning to rank using gradient descent. In: Proc. of the Twenty-Second International Conference on Machine Learning (ICML 2005), Bonn, Germany, pp. 89–96 (2005)
15. Küster, U., König-Ries, B.: Towards standard test collections for the empirical evaluation of semantic web service approaches. International Journal of Semantic Computing 2(3), 381–402 (2008)
16. Voorhees, E.M.: Evaluation by highly relevant documents. In: Proc. of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2001), New Orleans, LA, USA, pp. 74–82 (2001)

# Efficient Semantic Event Processing: Lessons Learned in User Interface Integration

Heiko Paulheim

SAP Research
heiko.paulheim@sap.com

**Abstract.** Most approaches to application integration require an unambiguous exchange of events. Ontologies can be used to annotate the events exchanged and thus ensure a common understanding of those events. The domain knowledge formalized in ontologies can also be employed to facilitate more intelligent, *semantic* event processing, but at the cost of higher processing efforts.

When application integration and event processing are implemented on the user interface layer, performance is an important issue to ensure acceptable reactivity of the integrated system. In this paper, we analyze different architecture variants of implementing such an event exchange, and present an evaluation with regard to performance. An example of an integrated emergency management system is used to demonstrate those variants.

## 1 Introduction

Integrating existing applications to form new systems is an important topic in software development, both for the purpose of saving engineering and maintenance efforts and for enabling the cooperation of existing systems (within as well as across organizations) [1]. Application integration can be carried out on three different levels: the data source level, the business logic level, and the user interface level [2]. Integration on the user interface level, or *user interface integration* for short, has two significant advantages [3]:

- Existing applications' user interfaces can be reused. Since the development of the user interface consumes about 50% of a software's total development efforts [4], the degree of reuse can be raised significantly.
- Users already familiar with existing user interfaces do not have to learn how to work with new ones. Therefore, the usability of an integrated user interfaces can be higher than of a user interface developed from scratch.

Integrated applications, and especially the implementation of cross-application interactions, require an event exchange mechanism [2]. To facilitate integration, the events issued by each application have to be commonly understood. Therefore, an ontology formalizing the information contained in the events is required [5]. It can be used to annotate the events, thus facilitating unambiguous event

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 60–74, 2010.

exchange. Using ontologies and the domain knowledge encoded therein also allows for a more sophisticated approach of dealing with events, called *semantic event processing* [6].

Such a sophisticated approach, e.g. incorporating an ontology reasoner, makes event processing a more complex and thus more time consuming task. But when dealing with UI integration and user interfaces in general, reactivity is an important factor with massive influence on the users' performance and satisfaction [7,8]. Therefore, semantic event processing mechanisms have to be implemented in an *efficient*, high-performance way. Various options exist for such implementations: events can be processed in a centralized or a decentralized manner [2], and instance data can be made available to the reasoner via push or pull mechanisms. In this paper, we analyze those different implementation variations and evaluate them with respect to performance.

The rest of this paper is structured as follows: in section 2, we outline the basic concepts of semantic event processing. In section 3, we introduce a framework for UI integration and present an example for semantic event processing, which does not only demonstrate how ontologies can be used to facilitate cross-application interactions such as drag and drop, but also how to make those interactions more intelligent and comfortable for the user. In section 4, we compare the different implementation variations based on the framework introduced, and we show the impact on the system's performance with each variant. We conclude with a survey of related work, a summary, and an outlook on future developments.

## 2 Background

Following the survey in [9], event-driven approaches can be roughly categorized in *event detection* (dealing with the detection and creation of events) and *event processing* (dealing with reacting to those events, e.g. by creating new events and/or changing a system's state). Furthermore, logic-based and non-logic-based approaches can be distinguished, where the former encorporates formal logic to detect or process events, while the latter does not. Following this categorization, the work presented in this paper is a formal approach to event processing.

The term *semantic event processing* denotes the processing of events based on information on the semantics of that event [6]. The decisions in event processing may range from filtering and routing of events to the production of new events from the detected ones. An events' semantics may be comprised of information about the actor who caused the event, the objects with which the event was performed, and many more. Westermann and Jain propose a six-dimensional common event model, including temporal and spatial aspects as well as information about the involved actors and information objects [5]. As semantics can be described by using ontologies based on formal logics, semantic event processing is a subset of logic-based event processing.

One simple form of event processing systems are *publish-subscribe-systems*. Here, clients subscribe to events which deal with a certain topic or, more general, fulfill a certain set of conditions. Ontologies may be used to provide a hierarchy

of topics, in the simplest case. Sophisticated approaches can use more complex annotations of events and allow subscription not only on topics, but also on subgraphs of the annotations, e.g. by using SPARQL queries [10,11,12].

More advanced approaches of event processing do not only forward or discard events, but may also create new events or allow the triggering of actions if events occur under certain conditions, an approach known as *event-condition-action* (ECA) rules. There are several approaches to implementing event-driven systems based on ECA rules, e.g. in Datalog [13] or RuleML [14]. The approach presented in this paper uses F-Logic [15] for implementing event-processing rules.

## 3 A Framework for Integration on the UI Level

Application integration on the user interface level, or UI integration for short, means assembling applications in a way that their existing user interfaces are preserved. Typically, those interfaces are presented as individual parts on the screen within one common frame, such as a portal [16], or a mashup [17]. In each case, the user can simultaneously and parallely interact with different applications. Most current approaches to implementing cross-application interactions, such as drag and drop from one application to another, are still very limited: they require writing a larger amount of glue code in each of the applications to be integrated, most often leading to code-tangling and non-modular integrated systems [2]. In this section, we introduce an ontology-based framework for UI integration which aims at remedying those limitations by introducing centralized semantic event processing.

### 3.1 Framework Architecture

Our framework for UI integration is based on Java and uses OntoBroker [18] as a reasoner and rule engine. It can be used to integrate applications written in Java as well as applications which can be wrapped in Java components, e.g. Flex applications by using libraries such as JFlashPlayer [19].

The integrated applications are connected via an event exchange, where events can be sent in a directed or broadcast way. To allow a common understanding and sophisticated semantic event processing, each event is annotated with different information, such as the action that has caused the event, the component that this action was performed with, and the types of objects that are involved in the action (see Fig. 1). Events can than be analyzed by a reasoner, and re-distributed and further processed by using a rule engine [3].

As user interface integration requires formal and modular models of the integrated applications as well as the part of the real world for which the applications are built [20], we use ontologies in our framework for modeling the relevant parts of the applications as well as the real world [3]. With the help of those ontologies, the events can be annotated to make them universally and unambiguously understandable by all parties, and to allow sophisticated semantic event processing. In our framework, we use three types of ontologies:

1. An ontology of the user interfaces and interactions domain defines basic concepts such as user interface components and actions that can be performed with those components. Furthermore, it defines a basic category for information objects, which are objects in the application carrying information (and which are typically visualized in user interfaces). Events are annotated with this ontology to categorize the type of action underlying the event, and the reasoner uses this ontology for formulating the queries needed in event processing.

   The UI and interactions ontology is an integral part of our framework.
2. A real world domain ontology defines the objects from the applications' real world domain, such as banking, travel, etc. The real world domain ontology is used to annotate data objects passed between the integrated applications. Each data object and its attributes is annotated with concepts from the domain ontology. Furthermore, it provides background knowledge which can be used to formulate more elaborate interaction rules.

   The real world domain ontology is not part of our framework, since the framework is domain-independent. For integrating applications, an appropriate real world domain ontology needs to be created or reused.
3. For each integrated application, an application ontology defines this application's components and the interactions that are possible with them. The components and actions defined in the application ontologies are *subclasses* of the respective concepts defined in the user interfaces and interactions domain ontology. All applications and their components are instances of the components defined in these ontologies, and the information objects they process *represent* objects from the real world domain ontology.

   During the integration, a developer has to implement one application ontology per integrated application.

The application ontologies also contain interaction rules that define how the user can interact with the different integrated applications, formalized in F-Logic [15]. Those interaction rules are used to define cross-application behaviour.

The integration framework is currently used in the research project *SokNOS*[1], an integrated application in the emergency management field [21]. The SoKNOS system itself is a larger system which consists of 20 integrated applications. On average, there are nine integration rules per application, forming a total of 180 integration rules. In SoKNOS, we use an ontology of the emergency management domain [22] as a real world domain ontology.

Further details of the framework are introduced in section 4, where the different architectural variants are discussed.

### 3.2 Example Interaction Rules: Intelligent Drag and Drop

Examples for cross-application interaction include displaying related information in one application when selecting an object in another one [23], cross-application

[1] ***S**ervice-**o**rientiertierte Archite**K**turen für **N**etzwerke im Bereich **O**effentlicher **S**icherheit* (German for *service oriented architectures for networks in the field of public security).*

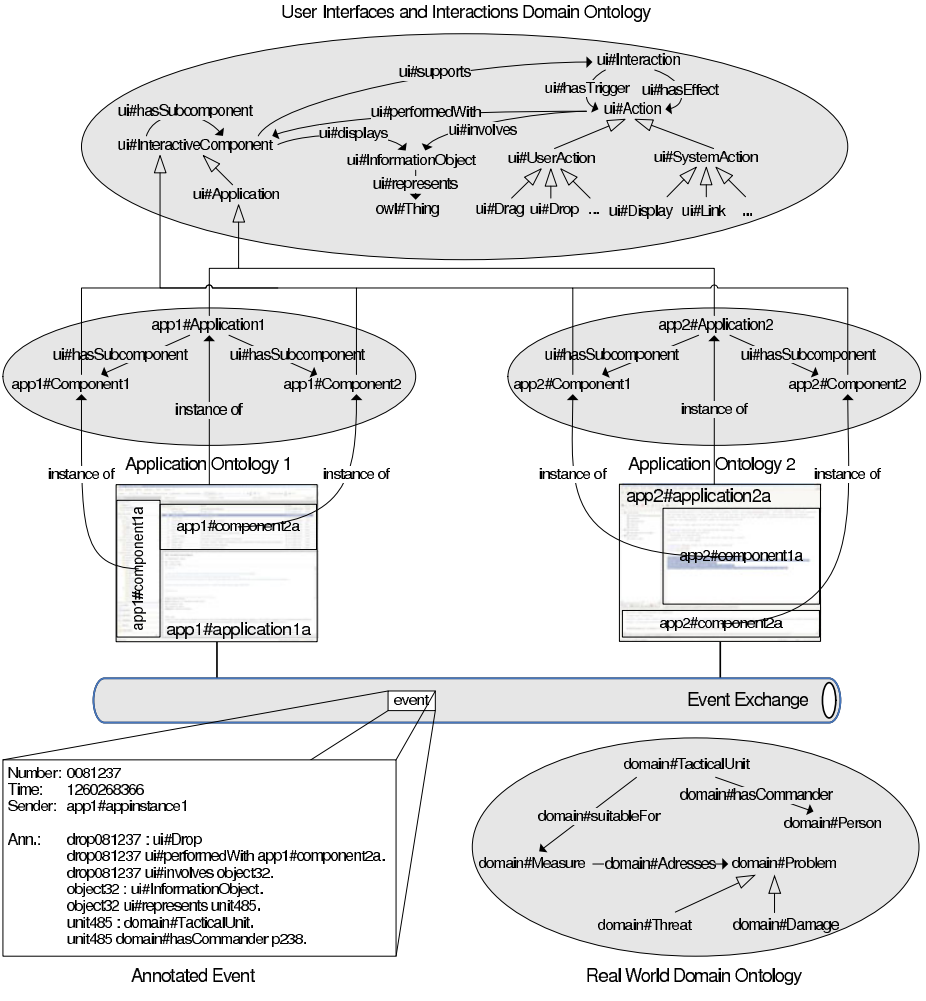



**Fig. 1.** The use of ontologies in our framework. Parts of the UI and interaction domain ontology, real world domain ontology and two application ontologies as well as their relations are shown. The event is annotated by using concepts from the different ontologies. A real world domain ontology from the emergency response domain is used in this example.

workflows (e.g. the creation of an object in one application requires entering data in another one), or dragging and dropping objects from one application to another one. In the following, we will introduce an example taken from the emergency response domain, which is implemented in the SoKNOS system [21].

The emergency response domain deals with entities like damages (such as fires, floodings, etc.), measures addressing those damages (such as fire fighting, building dams, etc.), and tactical units (such as fire brigade cars, helicopters, etc.). A useful interaction rule could be the following: if the user drops a tactical

unit on a measure, the unit is allocated to that measure (which may e.g. result in issuing an order to that unit), given that the unit is suitable for fulfilling that measure. The (non-trivial) decision whether a tactical unit is suitable for a measure or not requires a certain amount of knowledge of the emergency response domain [22].

To formalize such a rule, we use three types of ontologies, as depicted above: an ontology of the user interfaces and interactions domain (prefixed *ui*), an ontology of the real world domain (here: emergency response, prefixed *domain*), and the application ontology of the integrated application which should support the interaction[2] (prefixed *app*). Each interaction rule contains a triggering event (the $E$ in $ECA$, see section 5), an action to be performed as an effect (the $A$ in $ECA$), and some conditions under which the interaction can be performed (the $C$ in $ECA$). The example rule facilitating the drag and drop interaction, formalized in first order logic, looks as follows:

$$
\begin{aligned}
\forall c, t, io_1, io_2, u, m: \quad & app\#DisplayMeasureComponent\,(c) \wedge ui\#DropAction\,(t) \\
& \wedge ui\#InformationObject\,(io_1) \wedge domain\#TacticalUnit\,(u) \\
& \wedge ui\#InformationObject\,(io_2) \wedge domain\#Measure\,(m) \\
& \wedge ui\#represents\,(io_1, u) \wedge ui\#represents\,(io_2, m) \\
& \wedge ui\#performedWith\,(t, c) \wedge ui\#involves\,(t, io_1) \\
& \wedge ui\#displays\,(c, io_2) \wedge domain\#suitableFor\,(u, m) \\
\rightarrow \quad & \exists i, e: sys\#Interaction\,(i) \wedge ui\#LinkAction\,(e) \\
& \wedge ui\#supports\,(app, i) \wedge ui\#involves\,(e, io) \\
& \wedge ui\#hasTrigger\,(i, e) \wedge ui\#hasEffect\,(i, e) \qquad (1)
\end{aligned}
$$

The rule states that whenever a triggering event $t$ is processed which states that an information object $io_1$ representing a tactical unit $u$ is dropped on a component $c$ displaying an information object $io_2$ representing a measure $m$, the resulting interaction $i$ will have the effect $e$ of linking the tactical unit to the measure. The last term of the rule's body – $domain\#suitableFor\,(u, m)$ – involves the usage of real world domain knowledge (i.e. the conditions under which a tactical unit is suitable for a measure). Thus, the semantics of the event and the objects contained therein are used to provide an intelligent processing of that event. The rule also contains statements such as $ui\#displays\,(c, io_2)$, which need information on the system's state to be evaluated. In section 4.2, we will show alternatives of providing this information to the reasoner and rule engine.

An *intelligent* drag and drop interaction mechanism would not only allow the drag and drop itself, but also highlight the possible drop targets when the

[2] As the example shows, the interaction is triggered by the drop action, not by the drag action. Therefore, the application supporting the interaction is the one where the object is dropped, not the one from which it is dragged. The interaction as it is defined above works regardless of which application the object is dragged from.

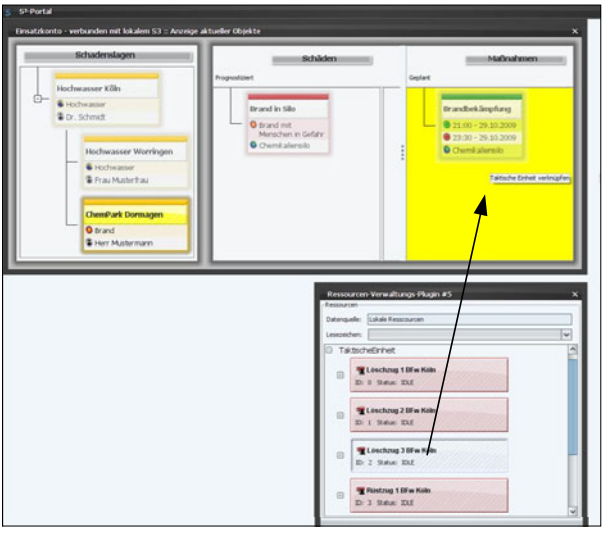



**Fig. 2.** A screenshot of intelligent drag and drop. Two integrated applications are shown: one for managing problems and measures addressing those problems (top), the other one for managing tactical units (bottom). If the user starts dragging a tactical unit (indicated by the arrow), the corresponding drop locations are highlighted. In addition, a tooltip is shown indicating the effect of dropping the object.

user starts dragging an object. Such a behaviour can be formalized as another interaction rule:

$$
\begin{aligned}
\forall c, t, io: \quad & ui\#InteractiveComponent\,(c) \wedge ui\#DragAction\,(t) \\
& \wedge ui\#InformationObject\,(io) \wedge ui\#involves\,(t, io) \\
& \wedge (\exists t_{hyp} : ui\#DropAction\,(t_{hyp}) \wedge ui\#involves\,(t_{hyp}, io) \\
& \quad \wedge \quad ui\#performedWith\,(t_{hyp}, c) \\
& \quad \rightarrow \quad \exists i_{hyp} : ui\#Interaction\,(i_{hyp}) \wedge ui\#hasTrigger\,(i_{hyp}, t_{hyp})) \\
\rightarrow \quad & \exists i, e : ui\#Interaction\,(i) \wedge ui\#HighlightAction\,(e) \\
& \wedge ui\#hasTrigger\,(i, e) \wedge ui\#hasEffect\,(i, e) \\
& \wedge ui\#performedWith\,(e, c) \qquad (2)
\end{aligned}
$$

This rules states that for any drag action $t$, if a corresponding (hypothetical) drop action $t_{hyp}$ on a component $c$ would serve as a trigger for *any* (hypothetical) interaction $i_{hyp}$, then this component is to be highlighted as an effect $e$ of that drag action[3]. This rule is only defined once and fires for *every* drag and drop interaction performed with *any* object from *any* application, as no concepts from

[3] Note that such a rule cannot be defined directly in most rule-based systems, due to the nested implication statement in the body. Therefore, further steps such as breaking down the rule into two or more rules are necessary for the actual implementation.

the domain ontology nor from any specific application ontology are referred to. It can thus be included in the user interfaces and interactions ontology. By using the natural language representation of the actions and objects involved in the computed events, the highlighted drop locations may also be augmented with tooltips (see Fig. 2).

## 4 Implementation Variants

We have tested different possible implementation variants with the framework described in section 3. For each variant, we have measured the average processing time for events, which is the main factor in the perceived reactivity of the integrated system. Throughout the experiments, we have varied some parameters, such as the number of instances of integrated applications that are used in parallel[4], or the number of integration rules per application[5].

### 4.1 Centralized vs. Decentralized Processing

Semantic event processing involves operations such as event filtering or the creation of new events from those that are already known. Such operations may be performed either by one central unit, or in a decentralized way by each participant involved in the event exchange [2]. The two variants are depicted in Fig. 3.

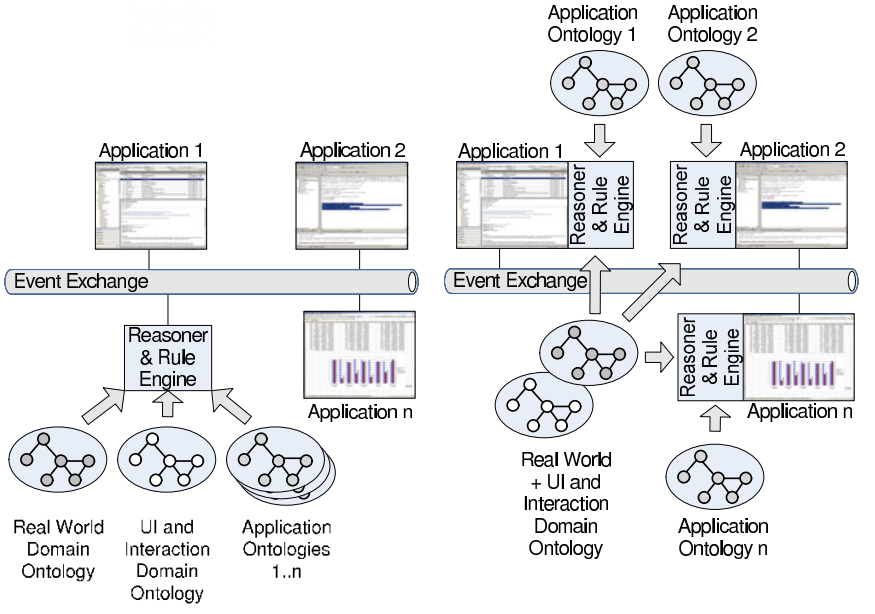
Both approaches have advantages and drawbacks. A centralized event processing unit needs to know about each application ontology including the event processing rules defined therein (see Fig. 3(a)), thus leading to a large number of rules to be processed by one unit. This unit may become a bottleneck, and cross-dependencies between rules can slow the whole process down.

With decentralized event processing, on the other hand, each event has to be analyzed and processed multiple times, even if the result of such processing is that the event is discarded in most cases. Furthermore, common domain knowledge, which is an essential ingredient of semantic event processing, has to be replicated and taken into the processing process each time (see Fig. 3(b)). Those operations can also have negative impact on the overall event processing performance.

The measurements depicted in Fig. 4 reflect these mixed findings. Both approaches scale about equally well to larger numbers of integrated applications (and thus, larger total numbers of integration rules). For integrated applications with a smaller number of integration rules per application, global processing is about 40% faster than local processing; with a growing number of integration

[4] Note that this is not the number of applications that are integrated into one system, but the number of instances of those applications that are used in parallel, where the latter is usually lower than the former.

[5] The tests have been carried out on a Windows XP 64Bit PC with an Intel Core Duo 3.00GHz processor and 4GB of RAM, using Java 1.6 and OntoBroker 5.3.




(a) Implementation with global event processing

(b) Implementation with local event processing

**Fig. 3.** Framework architecture using global vs. local event processing. The global variant uses one central reasoning and rule engine which processes all domain and *all* application ontologies. The local variant uses several reasoning and rule engines which each process all domain ontologies and *only one* application ontology.

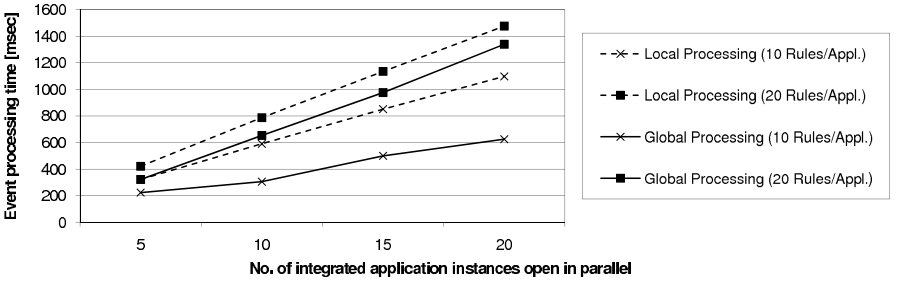



**Fig. 4.** Event processing performance comparison between using centralized and decentralized processing. Event processing time has been measured for 5 to 20 instances of integrated applications used in parallel, with 10 to 20 integration rules each.

rules per application, the difference is not as significant, but global processing is still slightly faster.

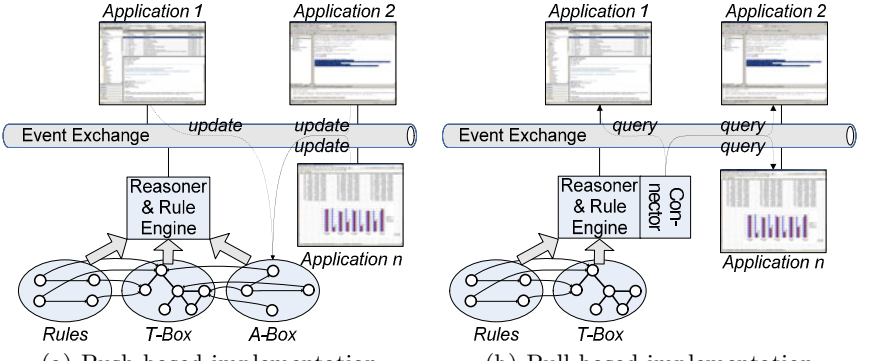
The main reason why global event processing turns out to be faster is that each event has to be processed only once, not once per receiving application – this advantage is not trumped by the larger number of rules in a centralized approach.

### 4.2 Pushing vs. Pulling of Instance Data

Ontologies have two parts, a T-Box, which contains the definitions of classes and relations, and an A-Box, which contains the information about instances of those classes. Reasoning about an integrated UI and events requires information about the system at run time, such as the application instances that are currently open, the components that constitute them, and the information objects they currently process (such as the example rule no. 1, which uses information about which components display which information objects as a condition). These kind of information are part of the ontologies' A-boxes.

There are two possible ways of implementing this A-box. A straight forward approach is to use an instance store for the instance data (see Fig. 5(a)). In this approach, integrated applications are responsible for sending (i.e. *pushing*) regular updates to assure that the instance store is always synchronized with the system its instances represent.

Another approach is to use the integrated system itself as an instance store. In this approach, an instance store connector is used to answer the reasoner's queries when needed. When called upon a query, it passes the query to the applications, collects (i.e. *pulls*) their answers and returns the instance data to the reasoner (see Fig. 5(b)).




(a) Push-based implementation (b) Pull-based implementation

**Fig. 5.** Framework architecture using a push-based vs. a pull-based approach. Both variants are demonstrated in an implementation combined with global event processing. In comparison to Fig. 3, the individual ontologies' rules, T-boxes and A-boxes are subsumed in this figure.

Both approaches have their advantages and drawbacks. Pushing instance data into an instance store causes redundancy, since the data contained in the instance store is also contained in the integrated system itself. Furthermore, to make sure that each query is answered correctly, the updates sent by the applications and the queries issued by the reasoner need to be properly serialized. This may result in slower query execution if a larger number of updates is queued before the query. These problems are avoided when using a pull-based approach.

On the other hand, a pull-based approach includes that instance data is retrieved from external systems while a query is processed. Depending on the reactivity of those systems, the overall query processing time might also increase.

A comparison of both implementations is shown in Fig. 6, using applications with ten integration rules each, and varying the number of applications that are integrated at the same time. For applications issuing five updates per second on average, both approaches scale equally well. For applications issuing ten updates per second, the pushing approach does not scale anymore.

With a pushing approach, the updates from the individual applications form an ever growing queue, slowing down the whole system until a total collapse. This behavior will occur with every reasoning system as soon as the frequency of updates exceeds the inverse of the time it takes to process an update. Thus, only approaches using the pulling approach scale up to larger integrated systems.

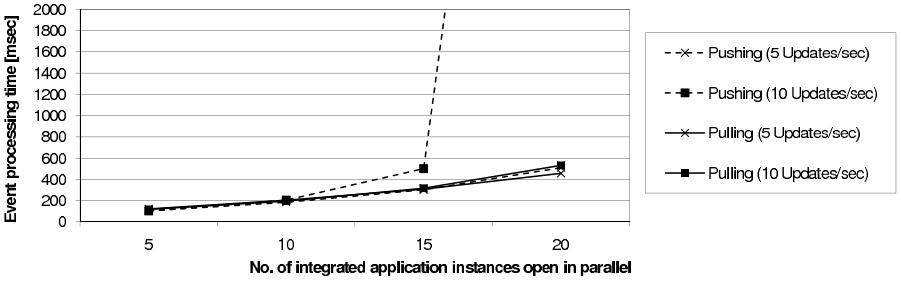



**Fig. 6.** Event processing performance comparison between pushing and pulling instance data. Event processing time has been measured for 5 to 20 instances of integrated applications used in parallel, with ten integration rules each, working at an update frequence of 5 to 10 updates per second. The figure has been cut at the two second mark where reasonable work is assumed to become impossible, as indicated in the HCI literature [7,8]. Systems integrated from applications issuing 10 updates per second collapsed when integrating more than 15 applications using the pushing approach.

## 5 Related Work

Few works exist which inspect the efficiency and scalability issues of using semantic events. In the field of *event detection*, the approach described in [6] uses modular event ontologies for different domains to enable more sophisticated semantic event processing mechanisms based on ECA rules. The authors propose to use reasoning to detect more complex event patterns in a stream of events described by ontologies, and to tell important events from non-important ones. The authors name scalability and real time processing as challenges, although no actual numbers are presented.

The idea of modular event ontologies is further carried out in [24] and [25]. The authors propose a modular approach where not only different ontologies, but also different languages can be used for individual parts of ECA rules. In this

very versatile approach, applications on the semantic web can register their rules as well as the corresponding processing units. Annotated events are then processed in a distributed fashion by dynamically calling the registered processing units. As such an approach involves (possibly remote) method calls during the event processing procedure, it may not be suitable in applications with real-time requirements.

The work discussed in [26] also uses events described with ontologies. The authors focus on mining information from a large database of events, allowing queries about characteristics such as the social relationships between the actors involved in events as well as the temporal and spatial relationships between events. Although real-time processing is not a necessary property in this case, the authors discuss the use of high-performance triple stores to allow fast query answering. A similar approach is described in [27], where operators for detecting complex events from a database of atomic events are introduced, based on the model by Westermann and Jain mentioned above.

In the field of *event processing*, there are a few examples of using semantic event processing in the user interface area. One approach is described in [28]. Annotated web pages can be used to create events that also carry information about the semantics of the involved objects. The authors present an approach for producing, processing, and consuming those events, and for constructing complex events from atomic ones. In their approach, the annotations may be used to formulate rules, but no reasoning is applied in processing the events. The approach is evaluated with respect to performance (although no variants are discussed) by means of the example of context-sensitve advertising, and it proves high scalability, but at the price of not incorporating domain knowledge and reasoning in the event processing mechanism at all (and thus not facilitating semantic event processing as defined above).

Another application of semantic event processing in the user interface area is shown in [29]. Here, it is employed for run time adaptation of user interfaces, allowing the incorporation of domain knowledge in the reasoning process. The authors present an e-government portal and show the use case of presenting appropriate content based on the users context, and evaluate their approach with regard to run time, reporting event processing times between a few and a few thousand milliseconds, depending on the number of rules and the number of instances that are used to answer a query. The authors present performance measures that allow a direct comparison to our work: While the performance marks are about the same, our approach allows for formulating arbitrary rules on the ontologies, the approach presented in that paper allows only the use of the class hierarchy. Thus, examples as shown in section 3.2 would not be possible with that approach.

## 6 Conclusion and Outlook

Application integration on the user interface level is an emerging area of research. With the example of an intelligent drag and drop mechanism, we have

shown how semantic event processing – i.e. event processing incorporating domain knowledge – can be employed to build more intelligent integrated systems.

Incorporating domain knowledge makes the event processing task more complex and thus more time consuming. As user interfaces require fast reactivity, our main focus was on the performance of the approaches, i.e. on the time to process an event. In this paper, we have introduced a framework for UI integration, which is capable of integrating Java-based as well as non-Java-based user interfaces, using ontologies and rules for the integration. Based on this framework, we have analyzed different variants of implementing a semantic event exchange based on the commercial off-the-shelf reasoner OntoBroker. We have introduced architectures for centralized and decentralized event processing, and for pushing and pulling instance data needed by the reasoning and rule engine.

While there are only minor differences between centralized and decentralized event processing, the pushing approach using a continuously updated instance store does not scale at all to larger and more complex integrated systems. Therefore, the key finding is that high-performance semantic event processing can only be implemented using the pulling approach.

For the work introduced in this paper, we have used a very basic and straight forward connector implementation (which still outperforms the pulling approach). In the future, we aim at improving this implementation, e.g. using more sophisticated caching approaches and rule sets that minimize the number of connector invocations. These actions may further improve the performance measures.

The framework for UI integration introduced in this paper is also subject to further research work. Current research addresses problems such as mediating between heterogeneous class models based on semantic annotation, and on incorporating more detailed context information, which can then be used to formulate more concise interaction rules (e.g. including restrictions based on the users' rights and context).

In summary, we have shown approaches to implement efficient semantic event processing, and discussed its implementation in the scenario of application integration on the user interface level.

## Acknowledgements

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grant no. 01ISO7009. The author would like to thank the reviewers for their helpful and constructive remarks.

## References

1. Linthicum, D.S.: Enterprise Application Integration. Addison-Wesley, Reading (1999)
2. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66 (2007)

3. Paulheim, H.: Ontologies for User Interface Integration. In: [30], pp. 973–981
4. Myers, B.A., Rosson, M.B.: Survey on user interface programming. In: CHI 1992: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 195–202. ACM, New York (1992)
5. Westermann, U., Jain, R.: Toward a Common Event Model for Multimedia Applications. IEEE MultiMedia 14(1), 19–29 (2007)
6. Teymouriana, K., Paschke, A.: Towards semantic event processing. In: DEBS 2009: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, pp. 1–2. ACM, New York (2009)
7. Miller, R.B.: Response time in man-computer conversational transactions. In: AFIPS 1968 (Fall, part I): Proceedings of the fall joint computer conference, part I, December 9-11, pp. 267–277. ACM, New York (1968)
8. Shneiderman, B.: Response Time and Display Rate in Human Performance with Computers. ACM Computing Surveys 16(3), 265–285 (1984)
9. Schmidt, K.U., Anicic, D., Stühmer, R.: Event-driven Reactivity: A Survey and Requirements Analysis. In: Proceedings of the 3rd International Workshop on Semantic Business Process Management (2008)
10. Wang, J., Jin, B., Li, J.: An ontology-based publish/subscribe system. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 232–253. Springer, Heidelberg (2004)
11. Skovronski, J., Chiu, K.: An Ontology-Based Publish Subscribe Framework. In: Proceedings of the 8th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2006 (2006)
12. Murth, M., Kühn, E.: Knowledge-based coordination with a reliable semantic subscription mechanism. In: SAC 2009: Proceedings of the 2009 ACM symposium on Applied Computing, pp. 1374–1380. ACM, New York (2009)
13. Anicic, D., Stojanovic, N.: Towards Creation of Logical Framework for Event-Driven Information Systems. In: Cordeiro, J., Filipe, J. (eds.) ICEIS 2008 - Proceedings of the Tenth International Conference on Enterprise Information Systems, Barcelona, Spain, June 12-16, vol. ISAS-2, pp. 394–401 (2008)
14. Paschke, A., Kozlenkov, A., Boley, H.: A Homogenous Reaction Rules Language for Complex Event Processing. In: International Workshop on Event Drive Architecture for Complex Event Process (2007)
15. Angele, J., Lausen, G.: 2. International Handbooks on Information Systems. In: Ontologies in F-Logic, pp. 29–50. Springer, Heidelberg (2004)
16. Wege, C.: Portal Server Technology. IEEE Internet Computing 6(3), 73–77 (2002)
17. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing 12(5), 44–52 (2008)
18. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: Meersman, R., Tari, Z., Stevens, S.M. (eds.) Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, January 4-8. IFIP Conference Proceedings, vol. 138, pp. 351–369. Kluwer, Dordrecht (1999)
19. Software, V.: JFlashPlayer Web Page (2009), http://www.jpackages.com/jflashplayer
20. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A framework for rapid integration of presentation components. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 923–932. ACM, New York (2007)

21. Paulheim, H., Döweling, S., Tso-Sutter, K., Probst, F., Ziegert, T.: Improving Usability of Integrated Emergency Response Systems: The SoKNOS Approach. In: Proceedings 39. Jahrestagung der Gesellschaft für Informatik e.V (GI) - Informatik 2009. LNI, vol. 154, pp. 1435–1449 (2009)
22. 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)
23. Eick, S.G., Wills, G.J.: High Interaction Graphics. European Journal of Operational Research 84, 445–459 (1995)
24. Alferes, J.J., Eckert, M., May, W.: Evolution and Reactivity in the Semantic Web. In: Bry, F., Maluszynski, J. (eds.) Semantic Techniques for the Web. LNCS, vol. 5500, pp. 161–200. Springer, Heidelberg (2009)
25. Behrends, E., Fritzen, O., May, W., Schenk, F.: Combining ECA Rules with Process Algebras for the Semantic Web. In: RULEML 2006: Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web, Washington, DC, USA, pp. 29–38. IEEE Computer Society, Los Alamitos (2006)
26. Aasman, J.: Unification of Geospatial Reasoning, Temporal Logic, & Social Network Analysis in Event-Based Systems. In: DEBS 2008: Proceedings of the Second International Conference on Distributed Event-Based Systems, pp. 139–145. ACM, New York (2008)
27. Rafatirad, S., Gupta, A., Jain, R.: Event composition operators: ECO. In: EiMM 2009: Proceedings of the 1st ACM International Workshop on Events in Multimedia, pp. 65–72. ACM, New York (2009)
28. Stühmer, R., Anicic, D., Sen, S., Ma, J., Schmidt, K.U., Stojanovic, N.: Lifting Events in RDF from Interactions with Annotated Web Pages. In: [30], pp. 893–908
29. Schmidt, K.U., Dörflinger, J., Rahmani, T., Sahbi, M., Thomas, L.S.S.M.: An User Interface Adaptation Architecture for Rich Internet Applications. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 736–750. Springer, Heidelberg (2008)
30. Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.): ISWC 2009. LNCS, vol. 5823. Springer, Heidelberg (2009)

# Usage Policies for Document Compositions

Sebastian Speiser and Rudi Studer

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
Institute of Applied Informatics and Formal Description Methods (AIFB)
Karlsruhe Service Research Institute (KSRI)
firstname.lastname@kit.edu

**Abstract.** The availability of contents and information as linked data or Web services, i.e. over standardized interfaces, fosters the integration and reuse of data. One common form of information integration is the creation of composed documents, e.g. in form of dynamic Web pages. Service and data providers restrict allowed usage of their resources and and link it to obligations, e.g. only non-commercial usage is allowed and requires an attribution of the provider. These terms and conditions are currently typically available in natural language which makes checking, if a document composition is compliant with the policies of the used services, a tedious task. In order to make it easier for users to adhere to these usage policies, we propose to formalize them, which enables policy-aware tools that support the creation of compliant compositions. In this paper we propose an OWL model of document compositions and show how it can be used together with the policy language AIR to build a policy-aware document composition platform. We furthermore present a use case and illustrate how it can be realized with our approach.

## 1 Introduction

More and more data on the Internet is published with standardized interfaces, i.e. as Web services or as linked data[1]. Legacy systems and data sources can be exposed with standardized interfaces by using tools, such as D2R [1] which publishes relational databases as linked data. It is also possible to integrate data from legacy systems with services, e.g. with IBM Mashup Center[2] which includes support for spreadsheets and SAP systems. This development fosters the reuse of information. One way to realize reuse is building dynamic document compositions. This means that a document description is created that specifies how the reused resources are composed to form the final document. A dynamic document composition system can then use this description in order to retrieve data copies from the specified resources and combine them to the composed document. Such systems are in wide use, for example in form of dynamic web pages and are conceptually similar to mashups.

[1] See e.g. http://webservices.seekda.com and http://linkeddata.org
[2] http://www.ibm.com/software/info/mashup-center/

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 75–89, 2010.

Standardized interfaces create unlimited possibilities on a technical level to combine services and data into new applications. However in most scenarios there will be restrictions, such that not all services are freely available to everyone and not all output data belongs to the public domain. These restrictions that apply when accessing a service or using its output data are currently typically available in natural language as terms and conditions. The evaluation of these terms requires high manual effort and makes it a tedious task to check if a composed document is compliant with the policies of the used services. Usage policies are a formalization of the terms and conditions of services that can be used to support the creation of compliant compositions with policy-aware tools.

Resources may for example be only allowed to be used by specific user groups, e.g. defined by the internal structure of a company. Consider for example a resource that provides detailed financial information about a company. Access to this resource should be restricted to managers of the company. This restriction also applies to data retrieved from the resource after initial access was granted, meaning that a manager who has access to the financial information is not allowed to pass the data to third parties.

Granting someone access to data does not imply that he can use it for any purpose. Usage policies also restrict data usage after access was granted, e.g. a real-time stock quote service might prohibit to display the quotes on a public homepage. Another aspect is that a usage permission can be linked with an obligation that has to be fulfilled. An example is a weather service that requires attribution of the provider whenever a weather forecast is shown. Other typical obligations include payments or to "share-alike" (i.e. a composition of resources has to specify the same policy as its components with a share-alike obligation).

In this paper we propose that usage policies are a requirement for dynamic document compositions that are used in environments that do not only consist of resources in the public domain. Reusability of services and easy access to data sources are key features of such systems.

The contributions of this paper include (i) a formal model for usage policies for dynamic document compositions (Section 3.1), (ii) a description how a platform for document compositions can support usage policies (Section 3.3), and (iii) an analysis of shortcomings of existing policy formalisms, including a proposal how they can be overcome (Section 4.4).

The rest of the paper is organized as follows: Section 2 presents a use case including a dynamic document composition platform that illustrates the requirements for a usage policy language. In Section 3 we introduce our approach to usage policies and show in Section 4 how it can be applied to the use case. Finally Section 5 gives an overview of related work and we conclude the paper and give an outlook to future work in Section 6.

## 2 Use Case

We consider a company that deploys a system that allows to include data from Web services and linked data sources as contents into composed documents. The

system is used to create documents and Web pages for internal and external use. Identification of users is realized by a centralized system maintaining the following hierarchy of groups: public (all users), internal (employees of the company), and manager.

The document system includes a service repository where both internally provided services and wrappers for external services are described. Note that the term service in this context also denotes linked data sources in the sense of data services. The document manager supports the search of services in the repository and the insertion of data pieces obtained through service calls into documents. It automatically checks usage policies and in case of non-compliance gives the reasons and hints how compliance can be achieved. We consider the following services:

1. A service provided by the manager Bob that delivers a text motivating every employee to increase the shareholder value. The text was obtained externally and was published under a Creative Commons attribution license (CC-BY[3]).
2. A real-time stock quote service, which delivers the current stock price of the company. It is provided by the external provider Powerquote. It is only available for company internal use and requires a payment of $ 1.00 per call.
3. A service delivering delayed stock quotes which is also provided by Powerquote. It can be used for external documents but requires attribution of the provider.
4. A service giving access to linked data about the financial status of the company, which is only available to managers.
5. A service providing a restricted view on the same financial data. The output of this service can be made available to the public.

Based on these services, we describe in the following two documents that employees of the company want to build.

First is a Web page for the intranet that shows the current stock quote and the motivational text. The builder of the document wants help of the document composition tool in order to define a policy for the Web page that is compliant with the policies of the used services. Furthermore in case that the document is classified as non compliant, he expects the tool to give him the reason for the decision.

The second document should be distributed to the public in order to convince investors to buy the shares of the company. The document's policy must allow use by the public and therefore only services should be made available that do not have a contradicting policy. The document should contain a stock quote of the company and a description of the financial status.

These tasks pose different requirements to the policy engine. The content of the first document is predetermined and the resulting policy should be modified accordingly. In the case of the second document, the desired policy is given and the content is chosen dependently.

[3] `http://creativecommons.org/licenses/by/3.0/`

## 3 Formal Usage Policies

This section describes our approach to formal usage policies for dynamic document compositions. First we propose in Section 3.1 a model of data sources, services and compositions that is useful from a policy view. Then in Section 3.2 we present requirements for the policy formalism and introduce AIR, a general purpose policy language by Kagal et al. [2], that is used by our approach. Furthermore we define some generally applicable policies using our proposed model. Finally in Section 3.3 we describe which operations a dynamic document composition platform must implement in order to support usage policies.

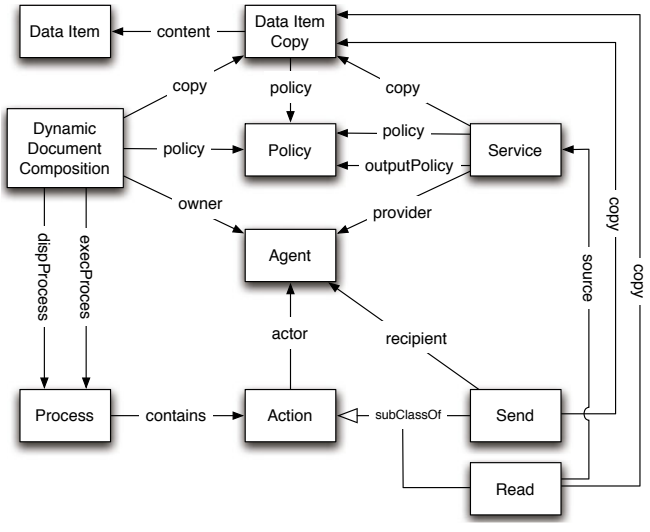



**Fig. 1.** Dynamic Document Composition Model for Usage Policies

### 3.1 A Model for Dynamic Document Compositions

Our model of dynamic document compositions is visualized in Figure 1. The basic component in our model is the data item, which can be for example an image, a text or a stock quote. A copy of a data item denotes a digital manifestation of the data item with an attached usage policy. This can be realized for example by annotating an HTML document with an RDFa representation of the policy, storing it in the metadata of a picture file or referring to an external file in an XML document. In our model we abstract from the technical realization and assume that a copy of a data item has some content and a usage policy. The usage policy of a data item copy regulates the transmission of the copy to other agents. We assume for now that there are no derivations of data items and the only possible actions are viewing it and transmitting it to others. Viewing a data item is assumed to be allowed for those who have a copy of it. Therefore

only transmission is regulated which is done by the action *send* that defines the actor who is sending it, the recipient and the corresponding data item copy. Note that data item copies are sent, which means that also the attached policy is transmitted.

A service is a provisioning of a specific data item copy, by a specific agent (the service provider) over a standardized interface, e.g. as a Web service, or as linked data. An agent (the service user) can obtain the data item copy of a service by executing a *read* action with the service as its *source*. The allowed read actions are regulated by the *policy* of the service and the obtained data item copy will be associated with the *output policy* of the service. The counterpart of reading a service is sending its data item copy, an action executed by the service provider with the service user as *recipient*. This implies that the policy of the service can only allow read actions for which the provider has the right to execute the corresponding send action. The policy of the service however does not have to be the same but can be more restrictive.

A composed document is a collection of data item copies and a layout function. The layout function merges the copies into a digital artifact that is sent to the viewer of the document. We consider the layout function for now as irrelevant for the policies and therefore treat the viewing of a document as a sequence of send actions where the document provider sends each data item copy to the document viewer. Obtaining the data item copies of the document is the execution process which consists of reads of services. Viewers of the document get the results of the display process which consists of sends of the data item copies. Therefore a document consists of a number of data item copies, a document owner, an execution process, a display process and a layout function. In the following we will refer to dynamic document compositions also as compositions, as this matches our view on these documents from a policy standpoint.

The display process is defined in terms of send operations that have an agent as recipient that is defined by the display process as its user role. This agent is bound during an execution to the specific user that views the document. The possible agents that can bind the user role are restricted by the composition's read policy.

Listing 1 formalizes the model as description logic formulas that can be represented in OWL[4]. This approach was taken as it supports interoperability in a Web scenario and existing reasoners can be used to infer implicit knowledge, e.g. a condition requiring a user to be a manager can be inferred to be fulfilled by a user that is the head of a department.

### 3.2 Policy Formalism

A usage policy does not only regulate the possible recipients but also required and forbidden actions in the process that uses an action. So basically a policy classifies whether a process is compliant or not. Such policies can be formalized as rules. In case of non-compliance the process creator wants to find out the reason for this

[4] OWL: Web Ontology Language, `http://www.w3.org/2004/OWL/`

**Listing 1.** Ontology about Dynamic Document Compositions

$$
\begin{aligned}
\texttt{DataItemCopy} &\sqsubseteq \forall{=}1\ \texttt{content.DataItem} \sqcap \forall{=}1\ \texttt{policy.Policy}\\
\texttt{Service} &\sqsubseteq \forall{=}1\ \texttt{policy.Policy} \sqcap \forall{=}1\ \texttt{outputPolicy.Policy} \sqcap\\
&\quad \forall{=}1\ \texttt{provider.Agent} \sqcap \forall{=}1\ \texttt{copy.DataItemCopy}\\
\texttt{Composition} &\equiv \forall\texttt{copy.DataItemCopy} \sqcap \forall{=}1\ \texttt{owner.Agent} \sqcap\\
&\quad \forall{=}1\ \texttt{execProcess.Process} \sqcap\\
&\quad \forall{=}1\ \texttt{dispProcess.Process} \sqcap \forall{=}1\ \texttt{policy.Policy}\\
\texttt{Process} &\sqsubseteq \forall\texttt{contains.Action}\\
\texttt{Action} &\sqsubseteq \forall{=}1\ \texttt{actor.Agent}\\
\texttt{Read} &\sqsubseteq \texttt{Action} \sqcap \forall{=}1\ \texttt{source.Service} \sqcap \forall{=}1\ \texttt{copy.DataItemCopy}\\
\texttt{Send} &\sqsubseteq \texttt{Action} \sqcap \forall{=}1\ \texttt{copy.DataItemCopy} \sqcap \forall{=}1\ \texttt{recipient.Agent}
\end{aligned}
$$

classification. This can be realized by a rule-system that supports dependency tracking, which means that the derived facts which led to a conclusion are saved. The owner of a non compliant document display process can then for example see that the reason for this decision is that an image is sent to the document viewer without sending a data item which attributes the image creator. As we assume that when checking a process for policy compliance we know all facts about the involved agents, actions and policies, we can evaluate the rules with a local closed world assumption. Otherwise an open world engine would not classify a process which is missing a required payment action as non compliant as long as there is the possibility that its existence can still be stated, e.g. in another file. As all knowledge about a document is maintained by the document manager in a central place, we can disregard these possibilities.

Accountability in RDF (AIR) is a policy language that comes with a reasoner that supports our requirements [2]. Policies are specified as RDF models in the N3 syntax extended with expressions to introduce quantified variables. An `air:Policy` defines via the `air:rule` relation one or more rules. Rules can use variables which are introduced by `@forSome :VAR1` or `@forAll :VAR2` statements. A rule evaluates its `air:if` clause, if it finds a binding for used variables, it further processes its `air:then` clause, which either specifies the next rule to evaluate or it `air:asserts` new axioms, e.g. that a variable is `air:compliant-with` the policy. Rules can also have `air:else` clauses which are evaluated when the if-clause does not have a binding. Besides asserting new axioms and specifying the next applicable rule, an if- or else-clause can also include an `air:description` which is a natural language explanation of the decision, which can use the bound values of variables. One or more policy files can be used to classify objects defined in one or more knowledge bases. Only explicitly stated individuals in the knowledge bases and those inferred from the policy rules are considered for the classifications.

The policies can also assert axioms other than (non-)compliance relations. For example the policy of a composition can require the actor that reads the

composition to belong to a specific class and assert that the agent playing the user role in the composition's display process therefore always belongs to this class.

The following is a general policy that applies the `process-action-rule` to all instances of `Process` (see `process-rule`). The `process-action-rule` states that if a process `:P` contains an action with a policy `:POLICY` to which `:P` is not compliant, the whole process is not compliant with the `ProcessPolicy`. If such an action does not exist, then `:P` is compliant with `ProcessPolicy`:

```
@forAll :P.
:ProcessPolicy a air:Policy ;
  air:rule :process-rule .
:process-rule a air:BeliefRule;
  air:if { :P a a:Process. };
  air:then [air:rule :process-action-rule].

:process-action-rule a air:BeliefRule;
  air:if { @forSome :ACTION, :POLICY.
           :P a:contains :A.
           :A a:policy :POLICY.
           :P air:non-compliant-with :POLICY. };
  air:then [air:assert [air:statement
              { :P air:non-compliant-with :ProcessPolicy .}]];
  air:else [air:assert [air:statement
              { :P air:compliant-with :ProcessPolicy .}]].
```

Further general rules include propagation rules, which state:

- A read operation assigns its data item copy the same policy as the output policy of the read service.
- A send operation has to fulfill the policy of the data item copy that is sent.
- All actions in the processes of a composition have as actor the owner of the composition.

### 3.3 Policy-Awareness for Document Composition Platforms

In the following we describe the usage policy view of a platform for dynamic document compositions. The two main components are the document manager and the service repository.

The service repository stores descriptions of all available services. The non-functional part of a service description includes usage policies which are split in three files: (i) a knowledge base defining the service, its provider, its data item copy, (ii) a policy, and (iii) an output policy. The repository supports the retrieval of a description for a given service. Another functionality could be to search services by specifying queries on the service descriptions, including restrictions on usage policies.

The document manager maintains for each composition two usage policy relevant files. One is the usage policy of the composition itself, which regulates allowed read operations on the document. The other file is a knowledge base

which contains definitions of the document, its owner, its data item copies and its processes. If a new service call is inserted in a composition, the following actions are taken:

1. Import the service knowledge base in the knowledge base of the composition.
2. Add a new `DataItemCopy` .
3. Add a new `Read` action having the new service as `source` to the document's execution process, the `actor` is the `owner` of the document, and the `copy` is the newly created `DataItemCopy`.
4. Add a new `Send` action to the display process, which sends the data item copy to the user role.
5. Check compliance for the execution and display processes. If they are not compliant show the document owner the justifications and help him to make the right changes.

The document manager allows a composition to be in a non-compliant state, which can occur in the process of building or changing a document. However it only executes and displays compliant compositions.

## 4 Realization of Use Case

In this section we describe how the services and documents presented in Section 2 can be realized with our approach to usage policies. First we explain the policies for the single services and afterwards we show the steps of document creation that involve decisions by the policy engine.

### 4.1 Motivational Text with CC-BY License

The service itself is provided by Bob freely for every user. Therefore we do not describe the service policy but concentrate on the output policy which is a formalization of the Creative Commons attribution license. The service definition including the data item and the required attribution in N3 syntax:

```
@prefix : <http://.../bob/motivation#>.
@prefix pol: <http://.../bob/motivationPolicy#>.

:MotivationService a m:Service;
   m:provider users:Bob;
   m:policy pol:MotivatonServicePolicy;
   m:outputPolicy pol:MotivationOutputPolicy.
:AttributionData a m:DataItemCopy;
   m:content "Created by XYZ".
```

The part of the policy file, defining the output policy:

```
@prefix f: <http://.../bob/motivation#>.
@prefix : <http://.../bob/motivationPolicy#>.

@forAll :P.
:MotivationOutPolicy a air:Policy;
   air:rule :PolicyApplicable.
```

Thus far it is stated that the policy requires the evaluation of the `:PolicyApplicable` rule, which is defined in the following and specifies that all applicable processes are subject to the `:AttributionRule`. A process is considered applicable if it contains a send action referring to a data item copy with a `:MotivationOutPolicy`.

```
:PolicyApplicable a air:BeliefRule;
   air:if { :P a m:Process;
               m:contains [a m:Send;
                           m:copy [m:policy :MotivationOutPolicy]]. };
  air:then [ air:rule :AttributionRule ].

:AttributionRule a air:BeliefRule;
   air:if { :P m:contains [a m:Send;
               m:copy f:AttributionData]. };
   air:then [ air:assert [ air:statement
              { :P air:compliant-with :MotivationOutPolicy.}]];
   air:else [ air:description( :P " has to attribute XYZ!." );
              air:assert [ air:statement
              { :P air:non-compliant-with :MotivationOutPolicy.}]].
```

The `:AttributionRule` checks if the process contains an action sending the attribution text. If this is the case, the process is compliant. If not, it is classified as non-compliant and the decision is justified by a natural language description requiring attribution. One future goal is to be able to represent also the other Creative Commons licenses. One example would be the non-commercial clause, which could be interpreted as granting compliance only if a process does not include payment actions and its policy does not require any payments. However it is currently not entirely clear what users consider as commercial usage[5].

### 4.2 Financial Status Report Services

The public financial status report is available for everybody and can be sent to everybody, therefore its usage policy is not of interest for us, as it would just classify every process as compliant.

The internal report however is only available to managers. In the following we just show the policy fragment checking identities of senders and recipients of the document and assume that applicability was already checked by previous rules and the variables `:P` (process) and `:S` (send action) are already bound.

```
:internal-report-auth-rule a air:BeliefRule;
   air:if { :S m:actor [a users:Manager]. };
   air:then [air:rule :internal-report-rec-rule ];
   air:else [air:description("Only Managers can access the report");
             ... :P air:non-compliant-with :InternalReportPolicy. ].
```

[5] Creative Commons has commissioned a study about the understanding of commercial vs. non-commercial use. An interim report can be found at: `http://mirrors.creativecommons.org/nc-study/NC_Use_Study_Interim_Report_20090501.pdf`

```
:internal-report-rec-rule a air:BeliefRule;
  air:if { :S m:recipient [a users:Manager]. };
  air:then [... :P air:compliant-with :InternalReportPolicy.];
  air:else [air:description("Report can only be sent to Managers");
            air:assert [ ... :P air:non-compliant-with ...]].
```

Additionally to the report's policy, also the providing service has an policy allowing only managers to read from the service, which is very similar to the `:internal-report-auth-rule`, but restricting a `Read` instead of a `Send` action.

### 4.3 Stock Quote Services

The delayed stock quote service has a policy which is similar to that of the motivational text with the difference that Powerquote has to be attributed.

In the following we concentrate on the policy part of the real-time stock quote service that requires the payment, as authentication and recipient restrictions were already shown in the policy of the internal financial report. We assume that `:P` (process) is already bound and a payment action is defined in some payment ontology:

```
:real-time-pay-rule a air:BeliefRule;
   air:if { :P m:contains [a pay:Payment;
                           m:recipient users:Powerquote;
                           pay:amount "$ 1.00" ]. };
   air:then [ ... :P air:compliant-with :RealTimePolicy. }]];
   air:else [ air:description("Service requires payment of $1.00");
              ... :P air:non-compliant-with :RealTimePolicy ].
```

### 4.4 Document Creation

Making a new dynamic document composition involves creating two files, one that contains a knowledge base describing the document, its data item copies and processes, and one policy file. The following is an exemplary description that is common for both documents of the use case:

```
:Document1 a m:Composition;
   m:policy pol:Document1Policy;
   m:execProcess :ExProc;
   m:dispProcess :DispProc.
:ExProc a m:Process.
:DispProc a m:Process;
   m:userRole :UserRole.
:UserRole a m:Agent.
```

The corresponding policy file defines at least two rules. The `:document1-user-rule` propagates the user restrictions that are defined in the `:document1-read-rule` to the user role of the display process, which is the user that represents all possible recipients of the send actions.

```
@forAll :M, :P, :U, :R.
:Document1Policy a air:Policy;
   air:rule :document1-user-rule;
   air:rule :document1-read-rule.
:document1-user-rule a air:BeliefRule;
  air:if { :M a m:Document;
             m:dispProcess :P.
           :P m:userRole :U. };
  air:then [air:assert [ air:statement { :U a users:Public. }]].
:document1-read-rule a air:BeliefRule;
  air:if { :P a m:Process;
             m:contains :R.
           :R a m:Read;
             m:source [m:policy :document1-policy];
             m:actor :U. };
  air:then [air:rule :document1-read-auth].
:document1-read-auth a air:BeliefRule;
  air:if { :U a users:Public . };
  air:then [ ... :P air:compliant-with :document1-policy. }]];
  air:else [ ... :P air:non-compliant-with :document1-policy. }]].
```

In the following we discuss how these files have to be modified for both documents during their construction.

**Document 1: Intranet Motivation Page.** First step is the insertion of the motivational text service into the document. This is realized by importing the knowledge base describing the service and adding the following statements to the knowledge base describing the composition:

```
:Motivation a m:DataItemCopy.
:Document1 m:copy :Motivation.
:ExProc m:contains [a m:Read;
                    m:source f:MotivationService;
                    m:copy :Motivation ].
:DispProc m:contains [a m:Send;
                      m:copy :Motivation;
                      m:recipient :UserRole ].
```

Afterwards the policy engine classifies `:DispProc` as non-compliant and gives as justification ":DispProc has to attribute XYZ!." and refers to the `:AttributionRule`. The composition owner now inserts an attribution, which is reflected in the document's knowledge base as:

```
:DispProc m:contains [a m:Send;
                      m:copy f:AttributionData;
                      m:recipient :UserRole ].
```

As a consequence the policy engine classifies the process as compliant. The next step is to insert the real-time stock quote service and adding the corresponding statements to the knowledge base. The consequences are that again the display process is marked as non-compliant because the stock quote is sent to the

user role which is only restricted to `users:Public`. As a consequence the document owner changes the policy by replacing the user role propagation and read actor restriction to `users:Internal`. However the display process is still non-compliant with the new reason "Real-time service requires payment of $ 1.00". Therefore the user adds a payment to the execution process, which is reflected by the following changes to the document description, that lead to a compliance classification and allow the execution and displaying of the document:

```
:ExProc m:contains [a pay:Payment;
                    m:recipient users:Powerquote;
                    pay:amount "$ 1.00" ].
```

**Document 2: Public Investor Advertisement.** For this document the delayed stock quote service and the public financial status report are inserted. The only requirement of their policies is that an attribution of Powerquote is inserted. How this works was already discussed for the previous document. The interesting part here is to select the right services that do not require a change in the document's read policy, which disqualifies the internal financial report and the real-time stock quote service. Such a reasoning over properties of policies is currently not supported by the AIR policy engine. For different usage policy scenarios this is a required functionality and in future we plan to develop an extension that supports this.

This is also useful for increasing the expressiveness of policies. For example a data item's policy could classify processes as non-compliant if they send out the item and have a policy that requires a payment (a possible interpretation of a non-commercial use clause).

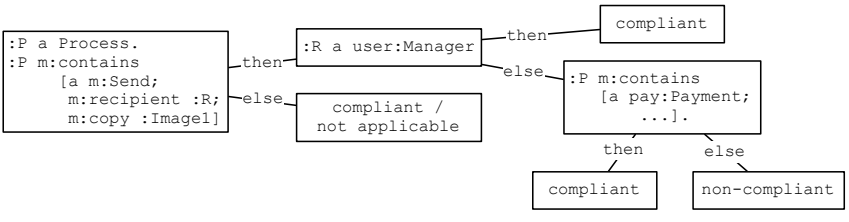



**Fig. 2.** Representing a policy as a tree of nested rules

In order to check such properties we treat policies with their nested rules as trees that have nodes representing the restrictions imposed by the if clauses and as leafs the (non-)compliance classifications. Figure 2 shows the tree of a rule that requires either a payment or restricts the possible recipients of a send action. These trees can then be used to check if a restriction is required by the policy, by checking if all paths from the tree root to a compliance leaf state the restriction. With negations several other properties can be checked, e.g. checking if the negation of a restriction is not required by the policy is equivalent to checking if there is a possible path to a compliance leaf including the restriction.

The above examples could be realized in the following ways: for checking if a policy restricts the possible users, we can check if it requires a restriction that states that the user belongs to a class that is not a super-class of the desired user class. The absence of required payments can be ensured by checking if the policy has a path to a compliance leaf that does not include a payment action.

**Discussion.** This section showed how the use case can be realized with our approach. Based on the example services, we illustrated how different policy aspects can be formalized using the proposed model. The process of creating the second document revealed that currently not all required policy operations are supported, namely the reasoning about properties of policies. We explained how this can be fixed without a need to change our model. The use case of the first document showed how service composition tools can support the building of compliant documents using formal policies and natural language explanations.

## 5 Related Work

Usage policies control access and usage of resources. Traditional access control mechanisms such as the role based model build abstractions which implicitly define an access control matrix. The matrix specifies for each combination of subject and resource if the access is allowed or not [3]. Such an approach can also be used with credentials instead of subject identities [4]. Access control provides proven and useful methods which are part of usage control, but is missing two aspects which are important for dynamic document compositions and services in general: obligations and usage control after initial access was granted.

Park et al. present the $\text{UCON}_{\text{ABC}}$ approach which is a theoretical approach that handles usage control, including obligations [5]. While the $\text{UCON}_{\text{ABC}}$ model gives an interesting base for diverse applications such as digital rights management, privacy and usage policies, it does not provide a general purpose solution that can just be applied to specific problems. Therefore it complements our approach with a theoretical foundation.

Creative Commons[6] defines several licenses for digital objects and an RDF formalization [6]. Such formal licenses are a form of usage policies as defined in this paper. However the language is restricted to the Creative Commons licenses, which have a limited scope. They are generally addressed to the public and therefore do not allow to specify rights for single users or user groups. Also the set of actions is restricted and does for example not include payments.

More general languages are used for digital rights management (DRM), e.g. ODRL [7]. These languages are based on XML and allow the specification of complex policies. They are mainly used for protecting digital media files and therefore contain actions from this area and cover areas such as encryption, media encodings and other domain specific aspects that are not applicable to dynamic document compositions. Another aspect is that these languages are

[6] `http://creativecommons.org`

lacking formal semantics which can lead to ambiguities when several heterogeneous service providers and consumers exist.

Gangadharan et al. developed an ODRL profile for service licenses (ODRL-S) [8]. Such a profile adapts ODRL for a specific application area. The defined actions are rather technical, such as for example composition, which can be used to generally allow or forbid the usage of the service in combination with other services. It is for example not possible to express that a document can only be combined with other documents or a translation service. There is also no support for data usage rights in ODRL-S.

A recent work by Seneviratne et al. presents a solution for a similar scenario as ours [9]. They analyze the use of images that have a Creative Commons license in Web pages. The work also uses an approach based on AIR policies. The key difference is that Senevirante et al. concentrate on copyright issues in static pages. In contrast our work presents an approach for dynamic documents and their creation process. Also we focus on usage policies that express more explicit restrictions based on user identification and also commercial aspects. Due to these differences our work complements their approach. Interesting overlaps do exist, as it is shown in Section 4 where we modeled a CC-BY license with our method.

## 6 Conclusions and Future Work

Dynamic document compositions are an powerful approach to the integration of data sources and services. The underlying model is flexible and treats widely used systems such as dynamic Web pages as a special case. In order to motivate service, data and content providers to participate in such scenarios they have to be ensured that their rights and restrictions about data and service usage are respected. We presented an approach to formal usage policies that can be used by providers to express their terms and conditions. We discussed how platforms for dynamic document compositions and service repositories can be extended to support users to adhere to usage policies.

Our approach is based on an OWL ontology, modeling the domain of dynamic document composition, which is used to describe the usage policy view of services and documents. Policies are expressed with rules that are evaluated in a local closed world environment and classify composed documents as compliant or non compliant. Evaluation is done by the AIR policy engine which not only returns the classifications but also the justifications for its decisions. This is used to give the user reasons why his document is not compliant and hints how he can change this.

For future work we plan to extend our model to include the collaborative creation of documents. For this we want to extend the simple composed document model to include more complex processes and services like for example translation and publishing services, and processes to assign the creation of new contents to providers. For a description of a powerful collaborative document creation system, we refer the reader to the work of Schuster et al. on document mashups [10].

Based on a planned implementation at hand, we want to extend the document manager's capabilities to automatically adjust a document or its policy in order to match certain policy requirements of used services. For example a user could specify that required attributions or user restrictions should be included automatically.

Another important point is to extend the capabilities of the policy engine to reason about properties of policies. For this it is necessary to inspect not only one trace through a policy's rules that leads to a compliance decision, but to have a look at the whole tree of policy rules and argue about the existence of paths with specific properties. We have argued that this is necessary in order to gain expressivity that is needed for common policy clauses, such as for example when a content creator wants to forbid that somebody is charging money for the creator's work.

**Acknowledgments.** The authors wish to thank Nelly Schuster, Sudhir Agarwal, Robin Fischer and Christian Zirpins for the useful discussions. This work was supported by the European project SOA4All.

## References

1. Bizer, C., Cyganiak, R.: D2R Server Publishing Relational Databases on the Semantic Web. In: WWW 2003, Posters (2003)
2. Kagal, L., Hanson, C., Weitzner, D.: Using Dependency Tracking to Provide Explanations for Policy Management. In: 2008 IEEE Workshop on Policies for Distributed Systems and Networks, pp. 54–61 (2008)
3. Lampson, B.W.: Protection. In: Proc. Firth Princeton Symposium on Information Sciences and Systems, pp. 437–443. Princeton University, Princeton (1971)
4. Agarwal, S., Sprick, B.: Specification of Access Control and Certification Policies for Semantic Web Services. In: E-Commerce and Web Technologies, pp. 348–357 (2005)
5. Park, J., Sandhu, R.: The $\text{UCON}_{\text{ABC}}$ Usage Control Model. ACM Transactions on Information and System Security 7, 128–174 (2004)
6. Abelson, H., Adida, B., Linksvayer, M., Yergler, N.: ccREL: The Creative Commons Rights Expression Language. W3C Submission (2008)
7. Iannella, R.: Open Digital Rights Language (ODRL) Version 1.1 (2002)
8. Gangadharan, G.R., D'Andrea, V., Weiss, M.: Service Licensing Composition and Compatibility Analysis. Int. J. Cooperative Inf. Syst. 17(3), 301–317 (2008)
9. Seneviratne, O., Kagal, L., Weitzner, D., Abelson, H., Berners-Lee, T., Shadbolt, N.: Detecting Creative Commons License Violations on Images on the World Wide Web. In: WWW 2009 (2009)
10. Schuster, N., Zirpins, C., Tai, S., Battle, S., Heuer, N.: A Service-Oriented Approach to Document-Centric Situational Collaboration Processes. In: 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (2009)

# The Impact of Multifaceted Tagging on Learning Tag Relations and Search

Fabian Abel, Nicola Henze, Ricardo Kawase, and Daniel Krause

IVS – Semantic Web Group & L3S Research Center,
Leibniz University Hannover, Germany
{abel,henze,kawase,krause}@L3S.de

**Abstract.** In this paper we present a model for multifaceted tagging, i.e. tagging enriched with contextual information. We present TagMe!, a social tagging front-end for Flickr images, that provides multifaceted tagging functionality: It enables users to attach tag assignments to a specific area within an image and to categorize tag assignments. Moreover, TagMe! maps tags and categories to DBpedia URIs to clearly define the meaning of freely-chosen words. Our experiments reveal the benefits of these additional tagging facets. For example, the exploitation of the facets significantly improves the performance of FolkRank-based search. Further, we demonstrate the benefits of TagMe! tagging facets for learning semantics within folksonomies.

## 1 Introduction

Tagging systems like Flickr or Delicious organize and search large item collections by utilizing the Web 2.0 phenomena: users attach tags to resources and thereby create so-called tag assignments which are valuable metadata. However, imprecise or ambiguous tag assignments can decrease the performance of tagging systems regarding search and retrieval of relevant items.

For example a tag assigned to an image may describe the image with respect to different dimensions referring to the owner, an opinion or the usage context (task) and thus is only valid for a user-specific point of view [1]. Not all tags can be used for search, but most often tags describing the content of a document are applicable for search [2]. However, even those descriptive tags might characterize just a small part of an image and hence cannot be used to derive the overall topic of the image correctly. Finally, tag assignments suffer from ambiguity in natural languages. For disambiguation, approaches like MOAT [3] exist, which support users to attach URIs describing the meaning of a tag to a particular tag assignment analogously to semantic tagging in Faviki[1].

In this paper, we extend the common folksonomy model by flexible, contextual tagging facets. We present the TagMe! system[2] that introduces novel tagging facets: tag assignments are enriched with a DBpedia URI [4] to disambiguate the

[1] http://faviki.com
[2] http://tagme.groupme.org

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 90–105, 2010.

meaning of a tag. So-called *area tags* enable users to tag a specific part of an image (spatial tagging). Furthermore, a *category* dimension is offered to categorize tag assignments. We propose different ranking strategies that exploit the additional context by extending the FolkRank [10] algorithm. In our evaluation we show how the different strategies improve search significantly (see Section 4.2) and analyze the impact of the additional context on learning semantic relations among tags (see Section 4.3).

## 2 Context in Folksonomies

Folksonomies evolve over time when users annotate resources with freely chosen keywords. Research in the area of folksonomy systems most often focusses on lightweight models where folksonomies are basically considered as collections of tag assignments. In this section we introduce approaches for modeling context in folksonomy systems.

### 2.1 Folksonomy Models

Traditional folksonomy models describe the relations between users, tags and resources. According to [5], a folksonomy can be defined as follows.

**Definition 1.** *A* folksonomy *is a quadruple* $\mathbb{F} := (U, T, R, Y)$*, where* $U$*,* $T$*,* $R$ *are finite sets of instances of* users*,* tags*, and* resources*.* $Y$ *defines a relation, the* tag assignment*, between these sets, that is,* $Y \subseteq U \times T \times R$*.*

Some systems imply a folksonomy model that incorporates additional information indicating in which context a tag was assigned to a resource. In particular, such context might be formed by groups, which are finite sets of resources [6]. In Definition 2 we introduce a more generic folksonomy model that considers arbitrary type of context.

**Definition 2.** *A* context folksonomy *is a tuple* $\mathbb{F} := (U, T, R, Y, C, Z)$*, where:*

- $U$*,* $T$*,* $R$*,* $C$ *are finite sets of instances of* users*,* tags*,* resources*, and* context information *respectively,*
- $Y$ *defines a relation, the* tag assignment *that is,* $Y \subseteq U \times T \times R$ *and*
- $Z$ *defines a relation, the* context assignment *that is* $Z \subseteq Y \times C$

Given the context folksonomy model, it is possible to attach any kind of context to tag assignments. For example, the context folksonomy allows for tagging tag assignments. TagMe! [7], a tagging and exploration front-end for Flickr pictures, introduces three types of context: (i) spatial information describing to which part of a resource a tag assignment belongs to, (ii) categories for organizing tag assignments, and (iii) DBpedia URIs that describe the semantic meaning of a tag assignment. Such contextual information is simply assigned to a tag assignment by the relation $Z$.


**Fig. 1.** User tags an area within an image and categorizes the tag assignment with support of the TagMe! system

The spatial information as well as the categories are explicitly provided by the end-users via the tagging interface of TagMe! (see Figure 1). For each tag assignment a user can enter one or more categories that classify the annotation. While typing in a category, the users get auto-completion suggestions from the pre-existing categories of the user community (see bottom in Figure 1). When a user categorizes a tag assignment $y = (u, t, r) \in Y$ into category $c$ then this is modeled as relation $(y, c) \in Z$ where $c \in C$ can actually be an arbitrary tag, i.e. $c \in T$. Additionally, users are enabled to perform *spatial tag assignments*, i.e. to attach a tag assignment to a specific area, which they can draw within the picture (see rectangle within the photo in Figure 1) similarly to *notes* in Flickr or annotations in LabelMe [8]. In the context folksonomy a spatial tag assignment is simply modeled via a relation $(c_a, y) \in Z$ where $c_a$ refers to the context information that describes the area that is tagged. TagMe! automatically assigns DBpedia URIs to tag assignments by exploiting the DBpedia lookup service[3] (cf. Section 4.1). Hence, tag assignments have well-defined semantics so that applications, which operate on TagMe! data using the Linked Data[9] infrastructure, can clearly understand the meaning of the tag assignments. In particular, RDF data can be obtained by accessing resources via an alternative URI, e.g. http://tagme.groupme.org/TagMe/resource/220/rdf, where each RDF representation links to other RDF representations of users, tags, resources, and DBpedia concepts.

[3] http://lookup.dbpedia.org

## 3 Ranking Algorithms

In this section we present the ranking algorithms that we apply in our experiments (Section 4) to reveal the benefits of exploiting context information. We first outline the FolkRank [10] algorithm, which we use as baseline strategy for ranking resources in our search experiments. Later we present different algorithms (based on FolkRank) that leverage context folksonomies to improve ranking quality.

### 3.1 FolkRank

The FolkRank algorithm [10] operates on the folksonomy model specified in Definition 1. FolkRank transforms the hypergraph that is spanned by the tag assignments into a weighted tripartite graph $\mathbb{G}_{\mathbb{F}} = (V_{\mathbb{F}}, E_{\mathbb{F}})$, where an edge connects two entities (user, tag, or resource, i.e. $V_{\mathbb{F}} = U \cup T \cup R$) if both entities occur together at a tag assignment within the folksonomy: $E_{\mathbb{F}} = \{\{u,t\},\{t,r\},\{u,r\}|(u,t,r) \in Y\}\}$. The weight of an edge corresponds to the amount of the entities' co-occurrences. For example, the weight of an edge connecting a tag $t$ and a resource $r$ is defined as $w(t,r) = |\{u \in U : (u,t,r) \in Y\}|$ (cf. Definition 1) and thus corresponds to the number of users, who have annotated $r$ with $t$. The constructed graph $\mathbb{G}_{\mathbb{F}}$ serves as input for an adaption of the Personalized PageRank [11]: $\boldsymbol{w} \leftarrow dA_{\mathbb{G}_{\mathbb{F}}}\boldsymbol{w} + (1-d)\boldsymbol{p}$, where the adjacency matrix $A_{\mathbb{G}_{\mathbb{F}}}$ models the folksonomy graph $\mathbb{G}_{\mathbb{F}}$, $\boldsymbol{p}$ allows to specify preferences (e.g. for a tag) and $d$ enables to adjust the influence of the preference vector. FolkRank applies the adapted PageRank twice, first with $d = 1$ and second with $d < 1$ (in our evaluation we set $d = 0.7$ as done in [10]). The final vector, $\boldsymbol{w} = \boldsymbol{w}_{d<1} - \boldsymbol{w}_{d=1}$, contains the *FolkRank* of each folksonomy entity.

### 3.2 Category-Based FolkRank

The category-based FolkRank algorithm operates on a context folksonomy (see Definition 2) where the context is given by categories that are attached to tag assignments. The algorithm relates folksonomy entities via the category assignments and the main hypothesis is that entities sharing the same category are related to each other. Similarly to GFolkRank, the category-based FolkRank introduces an alternative approach for creating the weighted folksonomy graph $\mathbb{G}_{\mathbb{F}}$ (cf. Section 3.1). Categories are treated as tags ($c \in T_C$ where $T_C \subseteq T$) so that the set of nodes is extended with $T_C$: $V_{\mathbb{F}_{new}} = V_{\mathbb{F}} \cup T_C$. For each category assignment $(y,c) \in Z$, new edges are created to connect the given category $c$ with the resource and tag of the tag assignment $y$: $E_{\mathbb{F}_{new}} = E_{\mathbb{F}} \cup \{\{c,r\},\{c,t\}|c \in T_C, t \in T, r \in R, ((u,t,r),c) \in Z\}$. The weight of an edge $(c,r)$ corresponds to the frequency the category $c$ is assigned to a tag assignment that refers to $r$: $w(c,r) = |\{(u,t,r) \in Y : (u,t,r) \in Y, ((u,t,r),c) \in Z\}|$. Weights of $(c,t)$-edges are accordingly computed by counting the tag assignments that refer to $t$ and are categorized using $c$.

### 3.3 Area-Based FolkRank

While the categories are used to enrich the folksonomy graph with further edges and possibly also with further vertices, the area-based FolkRank merely modifies the weights of edges in $\mathbb{G}_{\mathbb{F}}$ (cf. Section 3.1). In particular, it emphasizes the weight of an edge between a tag $t$ and a resource $r$, i.e. $(t,r)$-edges, whenever $t$ and $r$ occur within a tag assignment $(u,t,r) \in Y$ to which spatial context information is attached to. The amplification is based on the size of the corresponding area as well as on the distance of the midpoint of the area to the center of the resource (in our experiments we examine pictures).

- **size** Our hypothesis is that the larger the size of an area the more important is also the corresponding tag for the given resource, i.e. the larger the area that is attached to $(u,t,r) \in Y$ is the more relevant $t$ is for $r$. The size of an area is measured relatively to the size of the resource. For example, if an area is associated to a tag assignment $(u,t,r)$ and the relative size of the area is $s = 0.4$, i.e. the area covers 40% of the resource, then we use $s^{-1}$ to emphasize the weight $w(t,r)$. As different users might attach differently sized areas to $(u,t,r)$, we use the average size $\bar{s}$ of those areas to finally compute the new weight of $(t,r)$-edges: $w(t,r)_s = \bar{s}^{-1} \cdot w(t,r)$.
- **distance** The second hypothesis is that tag assignments which are according to the spatial information relevant to the center of a resource are more important for the resource than tag assignments which are associated to the margin of a resource. The distance $d$ is also measured relatively and the weight $w(t,r)$ is emphasized with the average distance $\bar{d}$ of the areas attached to $(u,t,r) \in Y$: $w(t,r)_d = \bar{d}^{-1} \cdot w(t,r)$.

Finally, the weight of the edges $(t,r)$ is simply the average of $w(t,r)_s$ and $w(t,r)_d$: $w(t,r)_{area} = 0.5 \cdot w(t,r)_s + 0.5 \cdot w(t,r)_d$.

### 3.4 URI-Based FolkRank

The URI-based FolkRank operates on meaningful URIs instead of tags. Hence, the construction of the folksonomy graph $\mathbb{G}_{\mathbb{F}} = (V_{\mathbb{F}}, E_{\mathbb{F}})$ is modified as follows. The set of vertices is $V_{\mathbb{F}} = U \cup URI \cup R$, where $URI \subseteq C$ (cf. Definition 2) denotes the set of URIs that describe the meaning of the tag assignments. The set of edges is $E_{\mathbb{F}} = \{u, uri\}, \{uri, r\}, \{u, r\} | u \in U, uri \in URI, r \in R, ((u,t,r), uri) \in Z\}$ whereas there should only exists exactly one URI assignment $(y, uri) \in Z$ for each tag assignment $y$. The weights of the edges are computed in the same way as done by the traditional FolkRank algorithm.

The URI-based FolkRank algorithm is therewith resistant against ambiguous tags as well as synonymic tags. For example, given two tag assignments $y_1 = (u_1, t_1, r_1)$ and $y_2 = (u_2, t_2, r_2)$ as well as two context assignments $(y_1, uri_1)$ and $(y_2, uri_1)$, the URI-based FolkRank algorithm would replace the synonymic tags $t_1$ and $t_2$ by the unique URI $uri_1$ that clearly defines the meaning of the tags. It therewith, e.g., relates $r_1$ and $r_2$ as it constructs the edges $(uri_1, r_1)$ and $(uri_1, r_2)$. As the TagMe! system utilizes DBpedia URIs to define the meaning

of tags, we denote the URI-based FolkRank as *DBpedia FolkRank* in our search experiments in Section 4.

## 4 Evaluation

In this Section we analyze the benefits of multi-faceted tagging resulting in context folksonomies (see Definition 2), i.e. contextual information that is embedded in the folksonomy model. In particular, we analyze the three context types introduced by the TagMe! system.

1. *Categories* for organizing tag assignments.
2. *Spatial information* (areas) describing to which part of a resource a tag assignment belongs to
3. *DBpedia URIs* that describe the semantic meaning of a tag assignment.

From the TagMe! data it seems that users appreciate those tagging facets, e.g. 899 of the 1264 tag assignments, which were performed within the three weeks after the launch of the system, were categorized and 657 times the users assigned a tag to a specific area within a picture. In the evaluation period the system was used by 30 students with background in computer science where half of the participants also had a Flickr account and thus were able to tag own pictures while the other half did not had a Flickr account.

In our evaluation we examine the impact of the additional context generated by the multi-faceted tagging on search and mining tag relations. In particulr, the key questions we would like to answer are the following.

1. Does the exploitation of the additional context improve the search and ranking performance?
2. Do the different context types offer potential to learn advanced types of relations among tags?

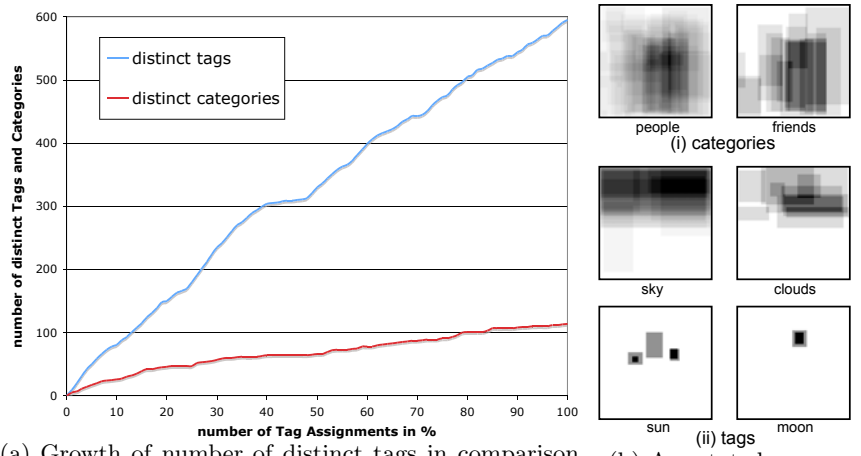
To better understand the context types available in TagMe!, we first present the results of analyses in which we investigated potential benefits of the categories and areas and examined strategies for mapping tags to DBpedia URIs (see Section 4.1. In Section 4.2 we afterwards summarize the results of our experiments that reveal the positive impact of the TagMe! context folksonomy on search. Finally, we examine the possibilities of learning semantic relations between tags in the TagMe! folksonomy.

### 4.1 Preliminary Analysis

In our preliminary analysis we try to gain first insights into the characteristics of the three TagMe! context types. We target the following questions.

1. How are *categories* used in comparison to tags?
2. What are the benefits of assigning tags to specific *areas* within an image (spatial information)?
3. How accurate can tags (and categories) be mapped to *DBpedia URIs* describing the meaning of the annotations?

**Analysis of Category Usage and Benefits.** Figure 2(a) shows the evolution of the number of distinct tags and categories: Although categories can be entered freely like tags, they grow much less than tags. Further, only 31 of the 87 distinct categories (e.g., "car" or "sea") have also been used as tags, which means that users seem to use different kinds of concepts for categories and tags respectively.




(a) Growth of number of distinct tags in comparison to distinct categories.

(b) Annotated areas.

**Fig. 2.** Annotation behavior in TagMe!

The TagMe! system supports users in assigning categories by means of auto-completion (see Figure 1). During our evaluation we divided the users into two groups: 50% of the users (*group A*) got only those categories as suggestion, which they themselves used before, while the other 50% of the users (*group B*) got categories as suggestions, which were created by themselves or by another user within their group. This small difference in the functionality had a big impact on the alignment of the categories. The number of distinct categories in group A was growing 61.94% stronger than in group B. Hence, the vocabulary of the categories can be aligned much better if categories, which have been applied by other users, are provided as suggestions as well.

**Analysis of Spatial Tagging Information.** Categories can be differentiated according to their usage. For example, some categories have never or very seldomly been used when a specific area of an image was tagged (e.g., "time", "location", or "art") while others have been applied almost only for tagging a specific area (e.g., "people", "animals", or "things").

The areas, can moreover be used to learn relations among categories and tags. Figure 2(b) shows (i) the areas that have been annotated whenever the categories "people" and "friends" have been used (the darker an area the more tags have been assigned to that area). As the areas that have been tagged in both categories

strongly correlate and as category "people" was used more often than category "friends" one can deduce that "friends" is possibly a *sub-category* of "people" even if both categories would never co-occur at the same resource. Relations between tags can also deduced by analyzing the tagged areas. Figure 2(b) shows (ii) the areas that were tagged with "sky", "clouds", "sun", and "moon"[4] and via the size and position of the area it is possible to learn that an entity *is part of* or *contained in* another entity (e.g., "sun, moon, and clouds are contained in sky"). The learned relations among tags and categories can moreover be used to learn and refine relations between URIs (ontology concepts) as TagMe! maps tags and categories to DBpedia URIs.

**Mapping to DBpedia URIs.** For realizing the feature of mapping tags and categories to DBpedia [4] URIs we compared the following two strategies.

- **DBpedia Lookup** The naive lookup strategy invokes the DBpedia lookup service with the tag/category that should be mapped to a URI as search query. DBpedia ranks the returned URIs similarly to PageRank [12] and our naive mapping strategy simply assigns the top-ranked URI to the tag/category in order to define its meaning.
- **DBpedia Lookup + Feedback** The advanced mapping strategy is able to consider feedback while selecting an appropriate DBpedia URI. Whenever a tag/category is assigned, for which already a correctly validated DBpedia URI exists in the TagMe! database then that URI is selected. Otherwise the strategy falls back the naive DBpedia Lookup.

The mappings of the naive approach result in a precision of 79.92% for mapping tags to DBpedia URIs and 84.94% for mapping categories considering only those tag assignments where a DBpedia URI that describes the meaning properly exists. The consideration of feedback, which is currently managed by the administrators of TagMe!, improves the precisions of the naive DBpedia Lookup clearly to 86.85% and 93.77% respectively, which corresponds to an improvement of 8.7% and 10.4%. As the mapping accuracy for categories is higher than the one for tags, it seems that the identification of meaningful URIs for categories is easier than for tags. Moreover, the precision of the category mappings, which are determined by the strategy that incorporates feedback, will further improve, because—fostered by TagMe!'s category suggestion feature—the number of distinct categories seems to converge (cf. Figure 2(a)). A more detailed evaluation of th DBpedia URI mapping is described in [13] and the implementation of advanced mapping strategies is part of our future work.

Overall, the DBpedia mapping reduces the number of distinct tags and categories within TagMe! by 14.1% and 20.9% respectively, which promises a positive impact on the recall when executing tag-based search. For example, while some users assigned the tag "car" to pictures showing cars other users chose "auto" to annotate other pictures that show cars. As both kinds of tag assignments

[4] The visualizations are based on 25 ("sky"), 10 ("clouds"), 6 ("sun"), and 2 ("moon") tag assignments respectively.

are mapped to "http://dbpedia.org/resource/Automobile", TagMe! can simply search via the DBpedia URI whenever users search via "car" or "auto" to increase recall of the tag-based search operations. With the *URI-based FolkRank* we present an advanced ranking algorithm that exploits the DBpedia URIs to improve the ranking performance (see evaluations in Section 4.2).

**Synopsis.** In summary, the context types available in the TagMe! folksonomy have a positive impact on identifying correlations between the folksonomy entities (e.g., identifying similar tags). Further, categories and areas enable the extraction of additional semantic relations between tags. As tags are mapped to DBpedia URIs that describe the meaning of a tag assignment, it is possible to deduce rich semantics from the context folksonomy available in TagMe!. The results of our preliminary analysis can be summarized as follows.

- The usage of categories differs from the usage of tags: Even for those users, who did not benefit from the category suggestions, the number of distinct categories is growing slower than the number of distinct tags.
- The spatial tag assignments can be used to learn typed relations among tags and categories such as *sub-category*, *sub-tag*, *part-of*, or *contained-in* relations. As tags and category assignments are mapped to meaningful URIs (ontological concepts), it is possible to propagate the learned relations to ontologies.
- A naive DBpedia lookup allows us to map tags and categories to ontological concepts (DBpedia URIs) with a high precision of 79.92% (tags) and 84.94% (categories). The consideration of feedback improves the accuracy of the mapping of tags and categories to 86.85% and 93.77% respectively.

This preliminary analysis already delivers insights into the potential of the context information available in the TagMe! folksonomy. In the next section we will evaluate whether categories, the size and position of areas, and the DBpedia URI assignments can be applied to improve the quality of search and ranking. Hence, we evaluate the ranking algorithms proposed in Section 3.

### 4.2 Search Evaluation

In our search evaluation we examine the impact of the advanced semantics provided by the TagMe! context folksonomy on search. In particular, we apply the FolkRank algorithm (see Section 3.1) as well as the Category-, Area-, and URI-based FolkRank adaptions to search and rank Flickr images and investigate how the different context types can help to improve the search performance. We evaluate the algorithms with respect to the following task.

> **Resource Ranking Task.** *Given a keyword query (tag), the task of the ranking strategy is to compute a ranking of resources so that resources that are most relevant to the keyword query appear at the top of the ranking.*

Our primary goal is to determine whether the additional context information has a positive impact on the ranking task. We further examine the characteristic strengths and weaknesses of the different context types by comparing the ranking performance of the corresponding FolkRank-based strategies.

**Data Set and Test Set.** We conducted our experiments on the TagMe! data set that evolved during the first month after the launch of the system. In this period the users created 1264 tag assignments where 899 tag assignments where also enriched with a category and 657 tag assignments were attached to a specific area of a Flickr resource. As outlined in Section 4.1, the number of distinct tags was growing faster than the number of distinct categories. Finally, the TagMe! data set contained distinct 610 tags and 118 distinct categories. The distribution of the usage frequency of tags shows a typical distribution: While some tags are used very often, the majority of tags is used just once.

The DBpedia URI assignments that were automatically attached by TagMe! were validated by hand so that the data set on which we performed the experiments did not contain wrong URI assignments. The cleaned data set finally contained 360 distinct DBpedia URIs referenced by tags and 92 DBpedia URIs referenced by categories. For 17% of the tag assignments there did not exist a correct DBpedia URI mappings.

Regarding relevance assessment, we selected 24 representative tags (according to the usage frequency) as keyword queries and asked TagMe! users to rate the relevance of a picture to a given query on a five-point scale: *yes*, *rather yes*, *rather no*, *no*, and *don't know*. Thereby, we obtained for each of the query all relevant resources in the TagMe! data set. On average, for each query there were nearly 30 resources in the data set that were rated as relevant (*yes*). However, four of the queries had below 10 relevant (*yes*) resources. For all the 24 tag-based queries there was a proper DBpedia URI available in the data set.

**Method and Metrics.** The *ranking task* defined at the beginning of the section requires the ranking strategies to rank those resources at the top of the ranking that are most relevant to the given query. We analyzed the ranking algorithms presented in Section 3 that are applicable to the TagMe! context folksonomy: FolkRank, Category-based FolkRank (CategoryFolkRank), Area-based FolkRank (AreaFolkRank), and URI-based FolkRank (DBpediaFolkRank). Each ranking strategy then had to compute a resource ranking for each of the 24 representative keyword queries. We measured the quality of the rankings using the precision at rank k ($P@K$), which represents the average proportion of *relevant* items within the top k. For our experiment we considered an item as *relevant* iff the average user judgement is at least "yes".

In addition to the FolkRank-based approaches we also consider a ranking algorithm (denoted as "F+C+A+D") that combines all four ranking strategies: Given the list of weighted resources as computed by the different algorithms it utilizes the average ranking weight to rank the resources.

We tested the statistical significance of our results with a two-tailed t-Test and a significance level of $\alpha = 0.05$. The null hypothesis H0 is that some strategy $s_1$ is as good as another strategy $s_2$, while H1 states that $s_1$ is better than $s_2$.

**Results.** Figure 3 shows the precisions (P@10 and P@20) of the different ranking strategies. Those algorithms that make use of contextual information embedded in the folksonomy perform better than the traditional FolkRank algorithm that considers only the tag assignments without any additional context. Between DBpediaFolkRank and FolkRank there seems to be no remarkable performance difference. However, as noted in Section 4.2, the DBpediaFolkRank is operating on 215 fewer tag assignments than the other algorithms. It is thus remarkable that DBpediaFolkRank still performs slightly better than FolkRank. The CategoryFolkRank performs good results especially for the precision within the top 20 (P@20). Hence, the hypothesis raised in Section 3.2 seems to hold: Category assignments can be used to relate resources. By exploiting the category context, the algorithm detects relevant resources that are not directly related via tag assignments to the given query. The AreaFolkRank algorithm, which exploits the size and position of spatial information attached to the tag assignments, is—with respect to P@10—the best algorithm among the core ranking strategies (P@10 = 52.9%). However, there is no significant difference between the FolkRank and the Area-, Category-, and DBpedia-based FolkRank.

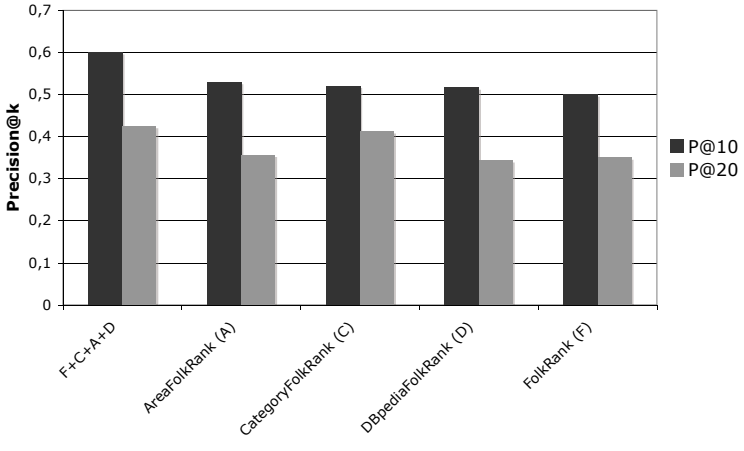



**Fig. 3.** Precisions of FolkRank-based search algorithms

The strategy "F+C+A+D", which combines all four core ranking strategies (i.e., FolkRank, CategoryFolkRank, AreaFolkRank, and DBpediaFolkRank), is the most successful strategy. It performs significantly better than the FolkRank algorithm regarding the P@10 and P@20 metrics. The combined strategy improves the precision of FolkRank by 20.0% and 21.4% with respect to the precision within the top 10 and top 20 respectively.

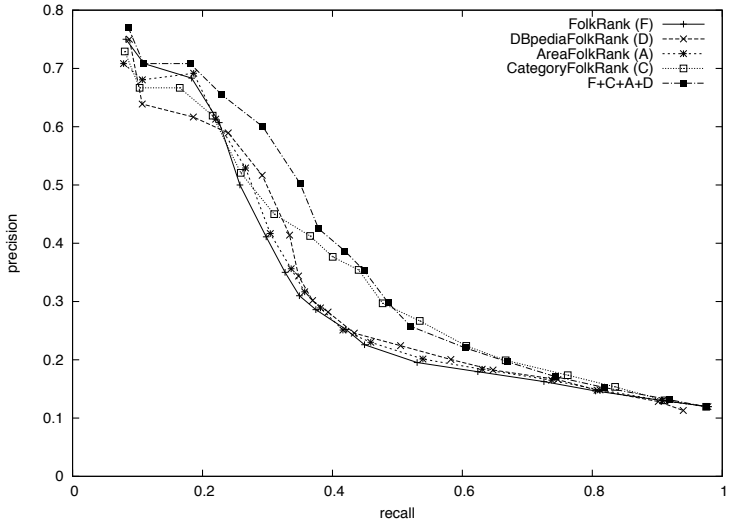



**Fig. 4.** Precision-recall diagram of FolkRank-based search algorithms

Figure 4 depicts the precision-recall diagram of the different FolkRank-based ranking algorithms. It underlines that the context-based approach, which combines FolkRank with the strategies that exploit the category, area, and DBpedia context, is the best performing ranking strategy as it results in the best precision-recall ratio. In the low recall interval, i.e. within the very top of the resource rankings, FolkRank can compete with the other algorithms. For example, the probability that a relevant resource appears at the first rank is 75.0% for FolkRank and 79.2% for the combined strategy. However, with higher recall values, the precision of FolkRank drops significantly stronger than the one of the Category-based FolkRank or the combined strategy F+C+A+D: At a recall level of 0.5 the precision of F+C+A+D and CategoryFolkRank is 0.29 and therewith significantly higher (approx. 45%) as the precision of FolkRank.

In summary, the exploitation of context embedded in the folksonomy is beneficial for ranking resources. While the size and position of the area helps to improve the precision particularly at the top of the resource rankings, the DBpedia and category context successfully contribute to improve the recall. And by combining the different context types we are able to improve the ranking performance of FolkRank significantly.

### 4.3 Learning Tag Relations

In addition to the exposed benefits in terms of search we hypothesize that the additional context is also helpful for learning fine-grained relations among tags. Different studies showed already that it is possible to learn hierarchical relations among tags by exploiting traditional folksonomy structures. For example, Halpin et al. extract sub-class relations by exploiting correlations between tags induced

by co-occurrence of tags [14] and Mika mines narrower and broader concepts by detecting communities and sub-communities within the tripartite graph spanned by traditional folksonomies [15] (cf. weighted tripartite graph in Section 3.1). Others apply association rule mining to detect super-sub-concept relations [16]. Given tags from Flickr, Schmitz detects tag pairs where one tag subsumes the other [17].

In this section we analyze the impact of the additional context types on learning tag relations. Therefore, we again apply the ranking strategies introduced in Section 3. However, instead of ranking resources we analyze their performance regarding ranking tags:

> **Tag Ranking Task.** *Given a tag $t$, the task of the ranking strategy is to compute a ranking of tags so that tags that are most related to $t$ appear at the top of the ranking.*

We run our experiments on the TagMe! data se as described in Section 4.2. For each of the 610 tags the ranking algorithms had to perform the ranking task defined above. For each ranking algorithm, we then accumulated the 610 rankings of tags to obtain a global ranking of tag pairs $(t_a, t_b)$, where the ranking score was computed as the sum of the FolkRank of $t_b$, when searching with $t_a$, plus the FolkRank of $t_a$, when searching for $t_b$ (cf. Section 3). We measure the success of the ranking strategies by means of the precision within the top k ranked tag pairs, i.e. the percentage of tag pairs within the top k that are—from our perspective and without considering the tag usage but just the words themselves—truly related to each other.

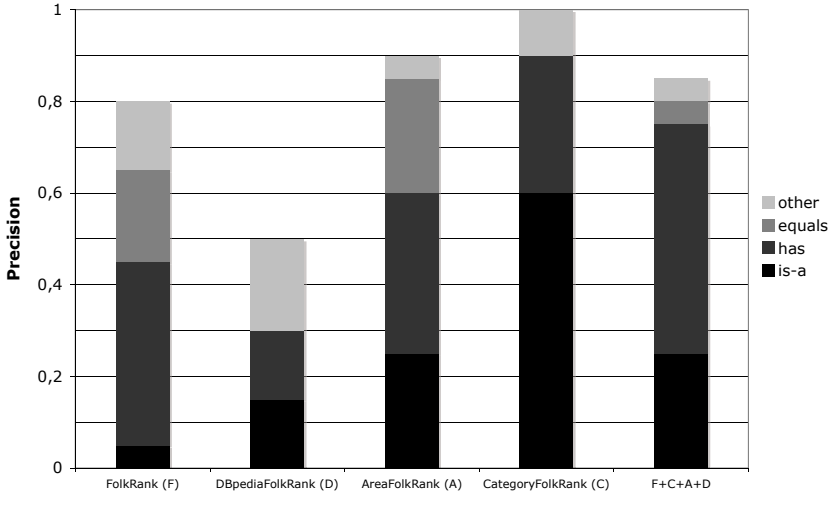



**Fig. 5.** Precision of tag pairs successfully classified as *related* as well as the distribution of relation types for the different strategies within the top 20

**Results.** Figure 5 shows the average performance of the different ranking strategies for identifying tag relations. When applying the ranking strategies as described above, the CategoryFolkRank is the most successful strategy as all tag pairs within the top 20 of the ranking are truly related to each other. FolkRank and the URI-based FolkRank (DBpediaFolkRank) perform worse as 20% and respectively 50% of the returned tags are really related. The bad performance of the URI-based FolkRank might be explained by the URI mapping: the actual FolkRank is computed on basis of the URIs to which the tags were mapped and the ranking of tag pairs is constructed by mapping the URIs back to tags. The latter step causes that errors are propagated, for example, the non-related tag pairs *(pink rose, bird)*, *(red rose, bird)*, *(white rose, bird)*, and *(rose, bird)*, which are each mapped to *(dbpedia:Rose, dbpedia:Bird)*, are each wrongly ranked at the same high level whereas FolkRank just ranks *(rose, bird)* at a high rank. AreaFolkRank as well as the combined approach (F+C+A+D) perform very well with a precision of 90% and 85% respectively. The combined strategy is negatively influenced by DBpediaFolkRank: omitting DBpediaFolkRank (*F+C+A*) leads to a precision of 95%.

Further, we examined the kind of tag pairs that were identified by the algorithms. Therefore, we considered the following types of relations.

- ***is-a*:** This relation refers to a sub/super-class or instance-of relation. For example: church is a building, horse is an animal, etc.
- ***has*:** Denotes that an entity consists of (is part of) or contains (is contained in) another entity. For example: car has wheel, sky contains cloud, etc.
- ***equal*:** Relation stating that two tags refer to the same thing (cf., `sameAs` relation in OWL). For example: spinne equals spider, marienkäfer equals ladybug (German/English).
- ***other*:** Other types of relations such as instances that belong to the same class or instances that care part of the same concept. For example: batman and superman, rooftop and façade, etc.

Figure 5 summarizes the types of relations that exist among the tag pairs returned within the top 20 ranking of the different ranking strategies. For example, regarding CategoryFolkRank 60% of the tag pairs can be typed as *is-a* relation, 30% as *has* relations and 10% denote other types of relations while tags that refer to *equal* concepts do not appear at the top of the ranking. The high frequency of *is-a* relations can possibly explained by the usage style of categories (cf. Section 4.1), i.e. categories are more broader and were often used to classify tags. The AreaFolkRank identifies all types of relations and is the strategy that identifies the most tag pairs that refer to *equal* concepts (25% of the tag pairs are typed as *equals*): different tags assigned to the same area of an image are thus likely to refer to the same concept.

In summary, the strategies that exploit the area and category context of tag assignments perform better (12.5% and 25% respectively) than the strategy that just considers the traditional tag assignments to identify related tags. Further, the exploitation of categories most often induces super/sub-class relations between tags while the area context has potential to infer *sameAs* relations.

## 5 Conclusion

In this paper we proposed an approach for multifaceted tagging and a corresponding model for embedding context into folksonomies. We implemented the model into the TagMe! system, a tagging and exploration interface for Flickr pictures that introduces three types of context: spatial information, categories and semantic meanings (URIs). We further developed and evaluated different search and ranking algorithms that exploit the contextual facets of the folksonomy. The algorithm that considered the different contextual facets significantly improved the precision of the baseline FolkRank algorithm by 20.0% and 21.4% with respect to the precision of the search result rankings. Relying on these results we demonstrated that contextual information can significantly improve search. Moreover, we were able to apply the ranking strategies for identifying semantic relations among tags. We showed that the exploitation of spatial information and categories attached to tag assignments improves the precision of inferring related tags by 12.5% and 25% respectively.

**Acknowledgments.** This work is partially sponsored by the EU FP7 project GRAPPLE (http://www.grapple-project.org/).

## References

1. Golder, S.A., Huberman, B.A.: The structure of collaborative tagging systems. The Computing Research Repository (CoRR) **abs/cs/0508082** (2005)
2. Bischoff, K., Firan, C., Paiu, R., Nejdl, W.: Can All Tags Be Used for Search? In: Proc. of Conf. on Information and Knowledge Management 2008, November 2008, pp. 193–202. ACM, New York (2008)
3. 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 (April 2008)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: 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)
5. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: BibSonomy: A Social Bookmark and Publication Sharing System. In: Proc. First Conceptual Structures Tool Interoperability Workshop, Aalborg, pp. 87–102 (2006)
6. Abel, F., Henze, N., Krause, D., Kriesell, M.: On the effect of group structures on ranking strategies in folksonomies. In: Baeza-Yates, R., King, I. (eds.) Weaving Services and People on the World Wide Web, July 2009, pp. 275–300. Springer, Heidelberg (2009)
7. Abel, F., Kawase, R., Krause, D., Siehndel, P.: Multi-faceted Tagging in TagMe! In: 8th International Semantic Web Conference (ISWC 2009) (October 2009)
8. Russell, B.C., Torralba, A.B., Murphy, K.P., Freeman, W.T.: LabelMe: A Database and Web-based tool for Image Annotation. International Journal of Computer Vision 77(1-3), 157–173 (2008)

9. Berners-Lee, T.: Linked Data - design issues. Technical report, W3C (May 2007), http://www.w3.org/DesignIssues/LinkedData.html
10. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
11. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project (1998)
12. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. In: Web Semantics: Science, Services and Agents on the World Wide Web (July 2009)
13. Abel, F., Kawase, R., Krause, D., Siehndel, P., Ullmann, N.: The Art of Multifaceted Tagging – inteweaving spatial annotations, categories, meaningful URIs and tags. In: 6th Int. Conf. on Web Information Systems and Technologies (WEBIST), Valencia, Spain (April 2010)
14. Halpin, H., Robu, V., Shepard, H.: The Dynamics and Semantics of Collaborative Tagging. In: Proc. of the 1st Semantic Authoring and Annotation Workshop (SAAW 2006) (November 2006)
15. 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)
16. Schmitz, C., Hotho, A., Jäschke, R., Stumme, G.: Mining Association Rules in Folksonomies. In: Data Science and Classification, pp. 261–270. Springer, Heidelberg (2006)
17. Schmitz, P.: Inducing Ontology from Flickr Tags. In: Proceedings of the Workshop on Collaborative Tagging co-located with 15th Int. World Wide Web Conference (WWW 2006), Edinburgh, Scotland (May 2006)

# OKBook: Peer-to-Peer Community Formation

Xi Bai[1], Wamberto Vasconcelos[2], and Dave Robertson[1]

[1] School of Informatics, University of Edinburgh, UK
[2] Department of Computer Science, University of Aberdeen, UK
xi.bai@ed.ac.uk, w.w.vasconcelos@abdn.ac.uk, dr@inf.ed.ac.uk

**Abstract.** Many systems exist for community formation in extensions of traditional Web environments but little work has been done for forming and maintaining communities in the more dynamic environments emerging from *ad hoc* and peer-to-peer networks. This paper proposes an approach for forming and evolving peer communities based on the sharing of choreography specifications (Interaction Models (IMs)). Two mechanisms for discovering IMs and collaborative peers are presented based on a meta-search engine and a dynamic peer grouping algorithm respectively. OKBook, a system allowing peers to publish, discover and subscribe or unsubscribe to IMs, has been implemented in accordance with our approach. For the meta-search engine, a strategy for integrating and re-ranking search results obtained from Semantic Web search engines is also described. This allows peers to discover IMs from their group members, thus reducing the burden on the meta-search engine. Our approach complies with principles of Linked Data and is capable of both contributing to and benefiting from the Web of data.

## 1 Introduction

Nowadays, providers care more about how communities (e.g. via eBay) rank their products and services than how search engine giants rank them. On the other hand, service requesters trust recommendations by other peers in social communities more than advertisements from service providers. Therefore, the community plays a more and more important role in addressing service discovery issues. Several systems exist for community formation in extensions of traditional Web environments but little is known about how communities might be formed and maintained in the more dynamic environments emerging from *ad hoc* and peer-to-peer networks. A difficulty in establishing communities in traditional peer-to-peer systems is that there is no structure that can be used as a basis for forming communities; there is nothing analogous to the relations used behind the scenes in Web-based social networking stems in order to infer community information (such as friend-of-a-friend relationships). Some recent peer-to-peer knowledge sharing systems have, however, used languages for specifying choreography between peers that can be used to provide the relations to build social networks. The OpenKnowledge[1] project has developed a peer-to-peer knowledge sharing system in

[1] http://www.openk.org/

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 106–120, 2010.

which peer interactions are described as Interaction Models (IMs, encoded in Lightweight Coordination Calculus (LCC) [1]). This paper proposes an approach for forming and evolving peer-to-peer communities based on their shared interactions with the assistance from the IM republication system [5]. Based on this approach we have developed OKBook - an open online platform for this new form of peer-to-peer communities.

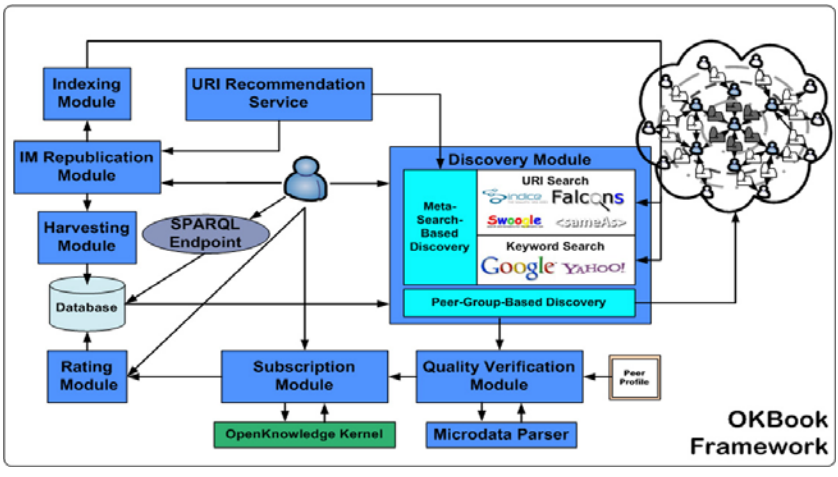



**Fig. 1.** Overlook of OKBook modules

Figure 1 shows the main components of the OKBook system. The *IM Republication Module* assists IM publishers in publishing IMs on Web pages using embedded micro-data such as RDFa [15]. Then the *Indexing Module* gets these IM pages indexed by informing search engine giants such as Google and Semantic Web Search Engines such as Sindice[2]. The *Discovery Module* offers a meta-search engine built on top of several SWSEs and ranks the search results by combining rankings from these search engines. Indices from Google and Yahoo!Search are used for assisting users in refining and adjusting their keyword-based queries when few URIs related to these queries are returned by SWSEs. Moreover, the *Discovery Module* also allows users to discover IMs that meet their requirements in terms of groups formed dynamically based on peers' historical interactions. Group members usually have some interests in common and this discovery mechanism can significantly cut down the IM search space compared with the meta-search-based mechanism. The *URI Recommendation Service* provides appropriate URIs for both the IM publisher within the annotation process and the IM requester within the IM querying process. The *Quality Verification Module* is in charge of discovering which role a peer is qualified to play by querying IM triples harvested by the micro-data parser as well as triples from the profile previously submitted by this peer. The *harvesting Module* gleans RDFa from the

[2] http://www.sindice.com

republished IM Web pages and stores triples into the back-end database. The *Group Discovery Module* analyzes the historical interactive data and discovers peer groups based on the criteria which will be discussed in Section 4. If a peer is qualified to play a role in an IM, it can subscribe to this IM through the *Subscription Module* which will display all roles this peer is capable of playing and subscription information will be finally forwarded to the OpenKnowledge Kernel (on a randomly selected peer) that coordinates this IM. After the execution, the *Subscription Module* forwards execution results from the OpenKnowledge Kernel to the *Rating Module* through which the peer can give some feedback about the IM based on its satisfaction with the execution. The ratings further inform IM selection beyond the ranking provided by the *Discovery Module*. Meanwhile, the execution result will be stored as one of historical interactions of this peer into the database. Moreover, a SPARQL endpoint is exposed and users can query and reuse triples (e.g., creating mashups) stored on OKBook via this endpoint so as to create more linked data and Web services.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to the LCC choreography language used in OpenKnowledge and OKBook. Section 3 describes our way of representing knowledge such as capabilities of peers and links from IMs to the Web of data in OKBook. Section 4 proposes two mechanisms for discovering IMs helpful to users based on the meta-search engine and dynamically formed peer groups, respectively. The connection between the OKBook platform and the OpenKnowledge Kernel also is detailed in this section. Focusing on group-based IM discovery, Section 5 experiments with our approach and presents case studies that demonstrate how peers may acquire desired IMs from their group members, thus reducing the burden on the meta-search engine. Section 6 presents related work. Section 7 draws conclusions and outlines our future work.

## 2 Choreography Description on OKBook

OKBook uses the structure of interactions shared between peers to provide the basis for inferring social linkage. An interaction model is a specification of a peer interaction. In OKBook this is specified using the LCC language. LCC was developed in the OpenKnowledge project for describing choreographies for peer-to-peer networks. Compared with WS-CDL [11], LCC is a choreography description language but also executable. The LCC syntax is described elsewhere (e.g. in [1]) and a detailed definition of its semantics is not essential to this paper. For our purposes, all readers need know is that it is a process language that differentiates the specific roles taken by different peers in a given interaction and specifies the overall interaction through definitions (in the form of message sequences) of each role.

## 3 Knowledge Representation for OKBook

OKBook is an online platform built on top of the OpenKnowledge Kernel based on the Drupal framework. Through this platform, peers can import, publish,

discover and subscribe/unsubcribe to IMs and join social groups which can help them to find out desired models for interactions and corresponding collaborators. The main modules of the system are described in this section.

### 3.1 Peer Capability Description and Its Storage

In the OpenKnowledge-based peer-to-peer community, each peer has a profile, in RDF, which will be uploaded to the discovery service as soon as this peer signs up to OKBook. A profile contains its owner's public information such as which roles it can play and proprietary information such as which constraints it can solve and corresponding methods in the form of OpenKnowledge Components (OKCs). OKBook can be regarded as a discovery server. In the peer-to-peer network, a discovery server behaves like a super peer and more than one of this type of peers may co-exist. When a peer registers on one of them, other discovery servers will also get copies of this peer's profile and the peer can log on to any of them using an identical username. Currently, OKBook supports the OpenID [18] standard which allows users to log on to different services with the same digital identity.

Centralized storage of triples is inefficient and incompliant with the peer-to-peer network for two reasons: firstly, it normally takes several hours to load in a billion RDF triples [16] so the centralized storage will be slow and impracticable; secondly, some data are proprietary and sensitive, which should be stored separately when copyright and security issues are taken into considerations. OKBook stores peers' profiles in a distributed way. Since Peer profiles are more concerned with providing information that can benefit interactions, they comply with choreographies in the whole OpenKnowledge system.

### 3.2 Linking Elements of Interaction Models to the Web of Data

In [5], a micro-data-based approach for republishing IMs has been proposed and a prototype has been implemented. OKBook further improves this prototype and integrates it as a module aimed at linking IMs to the Web of Data. As described in [5], currently IMs are republished using RDFa on XHTML Web pages and this task is fulfilled by the *IM Republication Module* as shown in Figure 1.

Wikipedia is a worldwide encyclopedia which has 262 language editions and naturally provides a knowledge base represented by natural languages. DBpedia [2] extracts structured information from Wikipedia and can be taken as a knowledge base represented by structured data such as CSVs, RDF triples which link structured knowledge to the World Wide Web. DBpedia assigns *http://dbpedia.org/resource/Name* (*Name* is taken from Wikipedia's *http://en.wikipedia.org/wiki/Name* like URLs) like URIs to all entities that have been crawled from Wikipedia. A URI is used for identifying a specific entity so we can also make use of URIs generated by DBpedia to annotate elements in IMs. However, sometimes more than one URI indicate the same individual and they will introduce differently within the annotation process in picking up a suitable URI and within the discovery process in matching annotations against the peer'

profile. In order to allow IM publishers to efficiently find appropriate URIs to annotate elements in IMs, we take advantage of the DBpedia Lookup service which employs Lucene's string-similarity-based ranking and relevance metric having been discussed in [3] to do the URI recommendation. As shown in Figure 1, the *URI Recommendation Service* wraps the above functionality. This IM republication adheres to the four principles of Linked Data because of the followings: URIs are used as annotations and they are dereferenceable HTTP URIs; employed URIs are curated by DBpedia and each of them can be provided with an RDF file and a human-readable Web page from Wikipedia; each IM finally becomes an RDF resource on the Web of data and it is also assigned with a URI linking to other URIs of annotations. The *Discovery Module* (discussed in Section 4) also uses this lookup service.

We employ the property *rdfs:comment* to allow publishers to add more human-readable details about published IMs. Sometimes, users want to use diverse vocabularies or even their own. Therefore, three types of search services are offered to users for selecting diverse URIs to annotate IMs. The first service wraps in SW search engines such as Sindice, Falcons[3], Swoogle[4], which crawl and index URIs from the Web continuously. The second service wraps in general search engines such as Google and Yahoo!Search, which can assist users in refining and adjusting their keyword-based queries when too few relevant URIs are returned by above SW search engines. The third service wraps in a co-references search engine sameAs[5], which can help users find equivalent URIs for a specific URI inputed by themselves.

When users publish new IMs, the embedded triples will be harvested automatically by the *Harvesting Module* as shown in Figure 1 and stored in the back-end database by OKBook using the ARC2 library[6]. Obtained triples will be also exposed to users via a SPARQL endpoint based on HTTP bindings. This gives an opportunity for others to reuse the RDF repository derived from republished IMs and establish their own applications of interest (e.g., IM mashups).

## 4 Discovery of Interaction Models and Collaborative Peers

IMs describe choreographies between peers as protocols which guide peers to interact with one another and achieve collaborations with different service providers. Peers collaborate by subscribing to an IM but finding an appropriate one for guiding peers' interaction is still a challenge. The keyword-based IM publication [4] limits IM discovery in the OpenKnowledge system. Therefore, we connect to the broader Semantic Web discovery effort by using URIs to republish IMs. Based on this strategy, two mechanisms for discovering desired IMs and collaborative peers are proposed in this section: meta-search-based discovery and peer-group-based.

[3] http://iws.seu.edu.cn/services/falcons/
[4] http://swoogle.umbc.edu/
[5] http://www.sameas.org/
[6] http://arc.semsol.org

The former mechanism is a generic solution for discovering IMs. The latter one discovers other collaborative peers which have common interests with the requester. The meta search works well especially when requesters' desired IMs are out of the scope of their groups' interests. The *Discovery Module* in Figure 1 is in charge of this IM discovery task.

### 4.1 Meta-search-Based Discovery

RDFa is a type of serialization for RDF so RDF triples parsed from the RDFa-embedded page can be indexed by SW search engines. OKBook provides a meta-search engine that allows users to input queries and access several SW search engines. When a user submits a new IM to OKBook, the submission will also trigger a request message to be sent to SW search engines. For example, Sindice supports the RPC ping API that is developed according to the specification of the Pingback [17] mechanism. So an IM submission or a submission of its revised version will ping Sindice for indexing or re-indexing this IM. On the OKBook query interface, users submit their queries to our meta-search engine by typing in several keywords. Normally, different SW search engines use different ranking mechanisms so ranking results are combined before being displayed to users. Suppose $Q$ denotes a querying phrase that a user types into our meta-search engine and $U$ denotes the URL minted for a specific IM indexed by one or more SW search engines. $S_i$ denotes the $i$th SW search engine. The rank of this URL returned by $S_i$ is denoted by $rank_{S_i}(U, Q)$. If $S_k$ has not previously indexed $U$, then $rank_{S_k}(U, Q) = 0$. We take the weighted average of ranks of $U$ returned by different search engines as the overall rank of $U$ on our meta-search engine, which is calculated by the following equation:

$$rank(U, Q) = \frac{\sum_{i=0}^{N} \chi_i \times rank_{S_i}(U, Q)}{N} \qquad (\sum_{i=0}^{N} \chi_i = N)$$

Here, $N$ denotes the overall number of search engines on which our meta-search engine is built. $\chi_i$ denotes the weight on the rank returned by $S_i$ and it actually reflects the user's preference. $\chi_i$ is equal to one by default. However, if a user prefers some search engines over others, he or she can inform our meta-search engine of this by changing values of weights on the OKBook search panel. Using this equation, ranks of all URLs will be synthesized and shown to users, in descending order of relevance. If more than one URL has identical ranking value, they will be displayed together as a group on the result page. Nonetheless, their provenances will be retained in case users want to further explore relevant information (e.g., snippets of search results) provided by diverse SW search engines.

On the search panel of OKBook, the URI recommendation service having been used in the *IM Republication Module* is employed as well within the query suggestion process. When users type in keywords, the recommendation service will forward them to the DBpedia Lookup service which will calculate the similarities between the keywords and stored URIs and select out the most suitable URIs

for users to refine their queries. Since we use the same recommendation service in both the *IM Republication Module* and the *Discovery Module*, it is more likely that both the IM publisher and the IM requester will choose the same URI for the same individual. Needless to say, this will benefit the IM discovery because under this circumstance (a single URI is used for both annotating and querying), OKBook can just do the precise match between annotations and refined queries without employing any ontology matching or reasoning algorithms.

On the other hand, if the IM publisher and the IM requester represent a single object using heterogenous URIs, we align them by employing the sameAs service which has collected predicates of co-reference such as *owl:sameAs*, *skos:exactMatch* and *skos:closeMatch* from diverse vocabularies. Many more sophisticated techniques have been proposed for ontology matching and these also could be used on the OKBook platform for assisting users in discovering more useful services. Discussion of this is outside the scope of this paper.

### 4.2 Peer-Group-Based Discovery

Our peer-to-peer community is established based on peers' historical interactions. As mentioned earlier, when enough peers fill roles inside an IM, the *Submitting Module* will forward the IM and relevant subscription information to a coordinator randomly selected by the OpenKnowledge Kernel. After executed, the IM and the original subscription information will be sent back to OKBook which maintains a table to record the subscription history for each registry peer in the database. Peer groups will be discovered by analyzing the interaction history on the fly. For instance, the authorized peer will know peers with which it has been involved in executions of IMs and what other IMs these peers have subscribed to. This will facilitate IM discovery and peers can find more IMs with which they may potentially interact. After letting OKBook analyze IMs in which its group members and itself have been involved, a peer can subscribe to IMs by claiming to play a specific role inside. This method is actually derived from the Friends-Of-A-Friend (FOAF) idea. If the $PeerA$ has interacted with the $PeerB$, it is possible that the $PeerA$ will be also interested in other interactions not having involved itself but have involved the $PeerB$ and vice versa. Therefore, we use this method to group peers and name it the Interactions-From-An-Interaction (IFAI) method. The algorithm for the peer-group-based discovery is described in Algorithm 1. It is notable that Algorithm 1 describes a situation in which just peers that $PeerAP$'s friends know directly are considered. Actually, peers are allowed to do a deeper search depending on their preferences by invoking our IFAI method, which is helpful especially for peers who newly registered and have not gotten many friends. This can be also achieved by making use of newly discovered friend peers and running Algorithm 1 recursively.

### 4.3 Subscription Information Submissions and Feedback

After an appropriate IM is discovered, peers can subscribe to it via the *Subscription Module* as shown in Figure 1. In the OpenKnowledge Kernel, the

**Algorithm 1.** IFAI Algorithm (single step)

**Input**: the URI of current authorized peer, *apeer_uri* and the historical interaction record, *record*.
**Output**: URIs of group members, *fpeer_uris* and URIs of IMs these members were involved in, *im_uris*.

```
begin
    IM_URIs = getInvolvingIMs (apeer_uri, record);
    for each im_uri ∈ IM_URIs do
        partner_peer_uris = getInvolvedPeerURIs(im_exec);
        for each peer_uri ∈ partner_peer_uris do
            if apeer_uri equals peer_uri then
                continue;
            else
                fpeers = fpeers ∪ {peer_uri};
                IM_URIs' = getInvolvingIMs(peer_uri, record);
                for each im_uri' ∈ IM_URIs' do
                    if im_uri' ∈ im_uris then
                        continue;
                    else
                        im_uris = im_uris ∪ {im_uri'};
end
```

Distributed Discovery Service (DDS) will finally send subscription-related data (such as which peer fills which role) and the subscribed IM itself to a randomly selected coordinator, which takes charge of bootstrapping the interaction and executing the LCC protocol. In order to comply with this submission of subscription information and the OpenKnowledge Kernel, when a peer decides to subscribe or unsubscribe to an IM, OKBook will record relevant information such as the peer's user account, the URL of the IM, the time when this subscription occurs as well as some auxiliary information such as the peer's contact details. Before IM submissions, OKBook also checks if each role in the submitted IM is filled by at least one peer.

## 5 Experiments

After the execution of an IM, each involved peer may have a chance to extend its group information by making friends with other involved peers. This interaction-based community expansion will encourage peers to interact with others and also make them benefit from these interactions. In this section we describe some preliminary experiments on peer community formation using our approach.

### 5.1 Acquiring IMs from Discovered Group Members

OKBook provides peers with not only interactions in which they were involved but also interactions in which their participants were involved. These interactions can be actually taken as expansion seeds via which peers are likely to interact

with more others whom they are difficult if not possible to know only based the searching mechanism offered by Web sites (e.g., eBay). Suppose Alice bought a product from Bob via the trade IM described in Figure 2 via OKBook. This figure depicts an interaction in which a client purchases a product referenced by a product code from a shop using his or her credit card.

/∗ A client, $C$, sends out a message to a potential shop, $S$, in order to buy a product with a code $PC$ using his or her credit card $CC$. Then $S$ sends the receipt of the product back to $C$. When $S$ receives the message from $C$, it checks if $CC$ has enough credit to buy the product. If the credit is enough, $S$ completes $C$'s order by generating a receipt and send it back to $C$. There is one peer playing the client role and may be more than one peer playing the shop role. ∗/

```
r(client, initial, 1, 1)
r(shop, necessary, 1)

a(client(PC, CC), C)::
   buy(PC, CC)⇒ a(shop, S)← payby(CC)∧ lookup(S) then
   receipt(R)⇐ a(shop, S)

a(shop, S)::
   buy(PC, CC)⇐ a(client(_),C) then
   receipt(R)⇒ a(client(_), C)← enough_credit(CC, PC)∧
                                complete_order(PC, CC, R)
```

**Fig. 2.** Simple trade IM in LCC

Focused on this IM, the sequence diagram in Figure 3 shows that how the group-base IM discovery drives peer interactions. Suppose Bob is a retailer who bought the same product from another peer Carol (the original manufacturer of this product) using cash via another IM similar to the above one previously. By logging on to OKBook, Alice can reach and subscribe to the latter IM based on the automatically discovered peer groups. Therefore, when Alice intends to buy the same product next time, she has a chance to interact with other peers such as Carol via the latter IM instead of with Bob via the former IM. It is likely to happen that Alice will get a lower price this time. On the other hand, from the perspective of Carol, OKBook assists her in discovering a new customer. Once their interaction finishes, Alice's group will be enlarged by absorbing a new group member and a new IM.

We experimented with the IFAI algorithm by simulating the peer interactions. Firstly, 100,000 peers and 10 IMs were generated for our experiment. Since each interaction will involve at least two peers, we assumed each IM owns two roles and 80,000 interactions occurred in the end. In each interaction, two peers and one IM were randomly selected to play roles and model the interaction respectively. Moreover, a bidirectional link was generated to connect every two collaborative peers. Then we made each peer randomly select an IM to subscribe to and calculated how many peers can find desired IMs making use of our peer-group-based discovery rather than the assistance of the meta-search engine on OKBook.

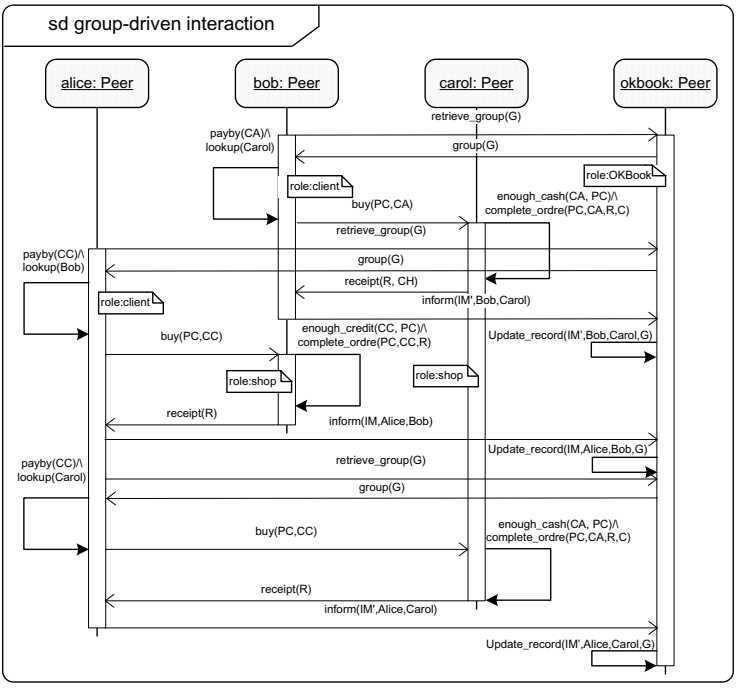
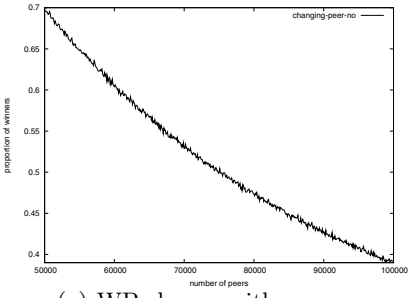



**Fig. 3.** Sequence diagram for a bunch of interactions driven by peer groups

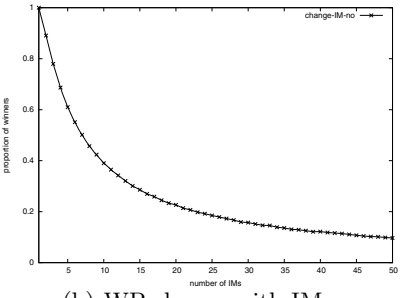
We name the percentage of all peers the peers that can get desired IMs from their group members account for as the Winning Proportion (WP). Then we ran this experiment 1,000 times and found that on average, 38.99% of peers got desired IMs from their own group members without searching on the meta-search engine. The result indicates that peer groups can reduce the burden on OKBook as well as other SWSEs which have indexed republished IMs.

Secondly, since this proportion was probably related to the parameters such as the number of peers, the number of IMs and the number of interactions, we redid the above experiment by changing one parameter but keeping other two fixed and calculating the WP in each case. The results are depicted in Figure 4(a), Figure 4(b) and Figure 4(c). From them, we can see that although the augmentation of the whole peer-to-peer network is uncontrollable, peers can still try to employ as small numbers of IMs as possible to maximize the WP. Moreover, as time goes by, more and more interactions will occur in the peer-to-peer community and the WP will also go up in terms of our experimental results.

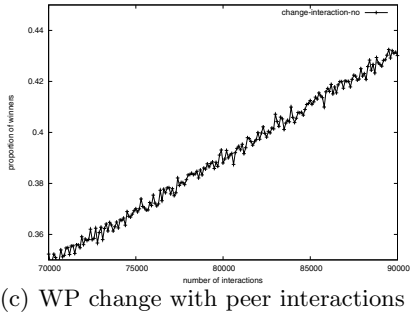
Thirdly, IMs may contain more than two roles and the WP value in reality is probably higher than the above one because under this circumstance, more group members and links may come up after each interaction. Base on the previous configuration of our experiment (100,000 peers, 10 IMs and 80,000



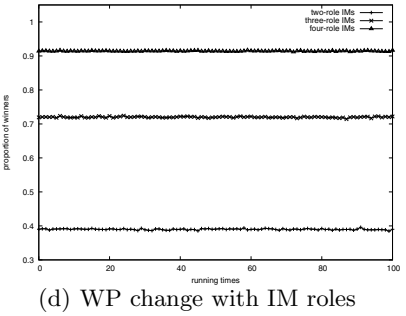

(a) WP change with peers




(b) WP change with IMs




(c) WP change with peer interactions




(d) WP change with IM roles

**Fig. 4.** Experimental results of the group-based IM discovery

interactions), we also investigated the performance of the peer-group-based discovery in the situations that interactions involved three peers or four peers. The discovery programme was ran for 200 times in each case and the result is shown in Figure 4(d). For the three-role case, on average, 72% of peers got desired IMs from their own group members and for the four-role case, this proportion went up to 91.43%. Therefore, in each interaction, if more peers are involved, more community members will gain in the near future.

### 5.2 Peer Subscriptions and IM Consumptions

Conventionally, when a user accesses to a shopping Web site such as eBay, he or she searches a desired product by typing in relevant keywords via the search UI. But this is based on the precondition that providers have already logged on to this Web site and published adverts for this kind of products on the server. On the other hand, keyword-based search has its natural limitations due to the synonymity and the ambiguity of phrases. Employed as annotations, URIs can provide more disambiguous identifiers to concepts that convey meanings users want search services to be truly knowledgable about. In the peer-to-peer network, peers are more autonomous and there is no central server in this distributed environment. On the OKBook platform, all a user has to do is search for an appropriate IM (recommended by OKBook) and subscribe to

it no matter if collaborative peers exist or not. Then OKBook will try to find other peers automatically who can collaborate with this peer and fulfill the interaction. Even if there is no provider providing the desired product for the time being, the subscription of this user will be still valid for a period of time (each subscription has an expiry time). As soon as enough collaborative peers have subscribed to an IM, this user will be informed of this and meanwhile, this IM goes into the execution procedure. However, for most conventional Web sites, this temporal"no result" is likely to end up with a page indicating the HTTP 404 error.

```
<html
xmlns=‘‘http://www.w3.org/1999/xhtml’’
xmlns:openk=‘‘http://homepages.inf.ed.ac.uk/s0896253/openk.owl#’’
xmlns:dbpedia=‘‘http://dbpedia.org/resource/’’
>
...
<div typeof=‘‘openk:InteractionModel’’>
 <span property=‘‘openk:has_declaration’’>r(client, initial, 1, 1)
 </span><br/>
 <span property=‘‘openk:has_declaration’’>r(shop, necessary, 1)</span>
 <div rel=‘‘openk:has_role’’>
  <div typeof=‘‘openk:Role’’ property=‘‘openk:has_roletype’’
  content=‘‘initial’’>a(<span property= ‘‘openk:has_name’’>client
  </span>(<span rel=‘‘openk:has_arg’’ typeof=‘‘openk:Argument
  dbpedia:Universal_Product_Code’’>
  <span property=‘‘openk:has_name’’>PC</span></span>), C)::<br/>
  <span rel=‘‘openk:sendout’’>
   <span typeof=‘‘openk:Message’’>
    <span property= ‘‘openk:has_name’’>buy</span>(
    <span rel=‘‘openk:has_arg’’>
     <span typeof=‘‘openk:Argument dbpedia:Universal_Product_Code’’>
      <span property=‘‘openk:has_name’’>PC</span></span>,
     <span typeof=‘‘openk:Argument dbpedia:Credit_card’’>
      <span property=‘‘openk:has_name’’>CC</span></span>)
...
```

**Fig. 5.** Republished trade IM in XHTML

Figure 5 gives the excerpt of proportional source codes of a Web page on which the trade IM described in Figure 2 has been republished. In Figure 6, OKBook helped a peer consume a Web page on which an IM has been republished and informed this peer of which roles it can play when the *Analyze* button was pushed. As shown at the bottom right of this figure, peers can also choose to make OKBook do the analysis as soon as they get the search result through the search panel. As mentioned previously, this analysis was based on the local profile of the peer as well as the harvested micro-data. With the movement of Linked Data, more and more peer-side applications will come up, which can parse and harness micro-data in a variety of ways but further discussion of this is outside the scope of this paper.

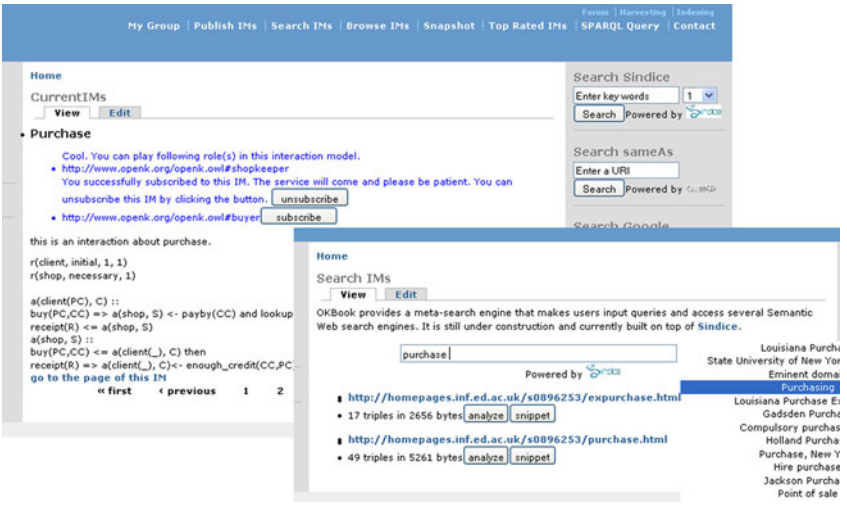



**Fig. 6.** Consumption of a republished IM

## 6 Related Work

From the perspective of orchestration, several approaches have been proposed for semantically describing Web services, such as OWL-S [6], WSDL-S [7] and SAWSDL [8]. Accompanying these, several matchmakers have been built, such as OWLS-MX [9] and SAWSDL-MX [10]. However, insufficient attention has been paid to semantically enhancing service descriptions from the perspective of choreography. WS-CDL [11] has been proposed for describing Web service choreography but it lacks appropriate support for semantics. Moreover, existing SWS choreography description languages such as WSMO [12] are heavyweight for portable devices such as mobile phones and personal digital assistants whose computation capabilities have been hampered by limitations of batteries and memories. OKBook employs LCC which is a lightweight and compact but expressive language for describing services provided by peers from the perspective of choreography.

Micro-data provides us with a more lightweight way of adding semantics to Web content for the purpose of making the content both human-readable and machine-readable. Several solutions for embedding micro-data into Web pages have been proposed. Davis proposed eRDF (embedded RDF) [13] from Talis, which can be used with any version of HTML but it restricts itself to the existing HTML attributes and does not support full RDF (e.g. there is no data type and no blank node). Microformat [14] also makes use of existing HTML and XHTML tags to convey metadata and other attributes but new formats require new data models. RDFa was proposed by Adida and Birbeck, which not only takes advantage of existing HTML attributes but also invents several new attributes in XHTML for flexibility and disambiguation. It supports full RDF and

can reuse data models created for RDF. In OKBook, a micro-data-based strategy is employed within the process of IM republication and users are encouraged to use URIs maintained by DBpedia to annotate IMs. This strategy works for any RDF vocabularies because the micro-data we employ in OKBook is RDFa that supports the full RDF data model. When peers face IMs republished using diverse URIs indicating the identical individual, they can use the sameAs service from OKBook or any local ontology matchmakers to fulfill the alignment task.

Compared with other online services such as eBay and Facebook, OKBook can dynamically form peer groups which assist peers in discovering more IMs of interest and corresponding collaborative peers. In the distributed environment, there is no central peer and peers are more autonomous. Therefore, with OKBook, users only have to search and subscribe to desired IMs and leave the remaining work to OKBook and the OpenKnowledge Kernel.

## 7 Conclusions

This paper proposes an approach to forming peer communities based on peer interactions. A system has been created as a preliminary implementation of this approach built on top of the OpenKnowledge Kernel. This system provides peers with a platform for publishing, discovering and subscribing or unsubscribing to IMs. Within the publishing process, RDFa is used for annotating elements inside IMs that will be finally published on Web pages. Within the discovery process, two mechanisms are proposed based on our meta search engine and dynamic peer grouping algorithm respectively. In the established peer community, within the subscription process, peers are allowed to discuss on and add rates to IMs through modules such as forums and voting. Nowadays, more and more service providers begin to look for customers instead of waiting for customers to look for their services. So we believe that in the near future, we will not search for services but services will find us in one way or another (e.g., through the peer community). OKBook is a prototype trying to achieve this goal via the distributed knowledge sharing environment.

## References

[1] Robertson, D.: Multi-agent coordination as distributed logic programming. In: Demoen, B., Lifschitz, V. (eds.) ICLP 2004. LNCS, vol. 3132, pp. 416–430. Springer, Heidelberg (2004)

[2] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: 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] Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., Lee, R.: Media meets Semantic Web - How the BBC uses DBpedia and Linked Data to make connections. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 723–737. Springer, Heidelberg (2009)

[4] Kotoulas, S., Siebes, R.: Adaptive routing in structured peer-to-peer overlays. In: Proceedings of the 3rd International IEEE Workshop on Collaborative Service-Oriented P2P Information Systems (WETICE 2007). IEEE Computer Society, Los Alamitos (2007)
[5] Bai, X., Robertson, D.: Service choreography meets the Web of data via micro-data. In: Proceedings of the AAAI Spring Symposium on Linked Data Meets Artificial Intelligence (LINKEDAI 2010). AAAI Press, Menlo Park (2010)
[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. W3C Member Submission (2004), http://www.w3.org/Submission/OWL-S/
[7] Akkiraju, R.: Web service semantics-WSDL-S (Version 1.0) (2005), http://www.w3.org/Submission/WSDL-S/
[8] Farrell, J., Lausen, H.: Semantic annotations for WSDL and XML schema. W3C Recommendation (2007), http://www.w3.org/TR/2007/REC-sawsdl-20070828/
[9] Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with OWLS-MX. In: Proceedings of the Internatinoal Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2006), pp. 915–922. ACM Press, New York (2006)
[10] Klusch, M., Kapahnke, P.: Semantic web service selection with SAWSDL-MX. In: Proceedings of the International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR$^2$ 2008) at ISWC 2008, pp. 3–18 (2008), CEUR-WS.org
[11] Kavantzas, N., Burdett, D., Ritzinger, G., Lafon, Y.: Web services choreography description language version 1.0. W3C Working Draft (2004), http://www.w3.org/TR/ws-cdl-10
[12] Lara, R., Roman, D., Polleres, A., Fensel, D.: A conceptual comparison of WSMO and OWL-S. In: Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 254–269. Springer, Heidelberg (2004)
[13] Davis, I.: RDF in HTML (2005), http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml
[14] Suda, B.: Using microformats. O'Reilly Press, Sebastopol (2006)
[15] Adida, B., Birbeck, M.: RDFa primer (2008), http://www.w3.org/TR/xhtml-rdfa-primer/
[16] Large triple stores (2009), http://esw.w3.org/topic/LargeTripleStores
[17] Langridge, S., Hickson, I.: Pingback 1.0 (2002), http://hixie.ch/specs/pingback/pingback-1.0
[18] Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In: Proceedings of the 2nd ACM Workshop on Digital Identity Management (DIM 2006), pp. 11–16. ACM Press, New York (2006)

# Acquiring Thesauri from Wikis by Exploiting Domain Models and Lexical Substitution

Claudio Giuliano[1], Alfio Massimiliano Gliozzo[2],
Aldo Gangemi[3], and Kateryna Tymoshenko[4]

[1,4] FBK, Via Sommarive 18, Trento, Italy
[2,3] STLab-CNR, Via Nomentana 56, 00161 Rome (RM) Italy
giuliano@fbk.eu, alfio.gliozzo@istc.cnr.it,
aldo.gangemi@istc.cnr.it, tymoshenko@fbk.eu

**Abstract.** Acquiring structured data from wikis is a problem of increasing interest in knowledge engineering and Semantic Web. In fact, collaboratively developed resources are growing in time, have high quality and are constantly updated. Among these problems, an area of interest is extracting thesauri from wikis. A thesaurus is a resource that lists words grouped together according to similarity of meaning, generally organized into sets of synonyms. Thesauri are useful for a large variety of applications, including information retrieval and knowledge engineering. Most information in wikis is expressed by means of natural language texts and internal links among Web pages, the so-called wikilinks. In this paper, an innovative method for inducing thesauri from Wikipedia is presented. It leverages on the Wikipedia structure to extract concepts and terms denoting them, obtaining a thesaurus that can be profitably used into applications. This method boosts sensibly precision and recall if applied to re-rank a state-of-the-art baseline approach. Finally, we discuss how to represent the extracted results in RDF/OWL, with respect to existing good practices.

## 1 Introduction

Acquiring structured knowledge from unstructured data is a crucial issue in the field of semantic technologies. Lexical knowledge plays a crucial role for applications dealing with natural language. Such knowledge, as it is usually processed by linguists, is typically represented following a structural paradigm in which *concepts are represented by a set of synonymous terms*. When one term refers to two different concepts, the term is ambiguous, while the fact that the same concept is expressed by two or more terms is called variability. Most computational lexicons are structured in this way, as variability and ambiguity are the more pervasive phenomena in language.

While defining concepts as sets of (nearly-) synonymous terms is a clear and pragmatic assumption among computational linguists, this is far less clear and pragmatic among domain experts, and even less among ontology engineers. However, the usefulness of lexicons like WordNet for ontology engineering techniques (ontology design, automatic mapping of ontologies) has eventually made that assumption acceptable, when revised appropriately. We will see how to reconcile the semantics of the results obtained through that assumption, with the formal semantics expected for ontologies.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 121–135, 2010.

Typically, the main weakness of computational lexicons is their lack of coverage when compared to actual usage of language. Lexical resources can be crafted by hand, however they typically suffer of poor coverage as all possible variants are typically not found by lexicographers. Corpus-based approaches can be used to enhance coverage, for example, to extract domain specific terminology or to find semantic relations among words. However, state-of-the-art approaches are not accurate enough, so domain experts are frequently involved in the process to provide additional validation.

The process of inducing terminology and paradigmatic relations among terms (e.g., synonymy and hyponymy) has been called *thesaurus induction* [6]. To this end, pattern-based approaches have been used [15], as well as distributional measures of lexical similarity based on word co-occurrence in documents [8] or in syntactic contexts [7]. Even though these approaches are scalable and cost-effective, their main limitation is that they can be used to discover relations among words, while semantic relations are defined between concepts. The term *thesaurus induction* is again a bit misleading when compared to what is known as a thesaurus in the Semantic Web literature. What computational linguists call thesauri are in fact a coarse approximation of mainstream thesauri known on the Semantic Web for the ability to be reengineered, e.g., through the SKOS vocabulary[1]. A SKOS-compliant thesaurus is similar to linguists' thesauri in the sense that it is made of concepts, which have preferred and alternative labels. But realistic thesauri (that SKOS describes) are actually richer than that, and include a whole range of semantic relations between concepts: *broader*, *related*, *mapping*, etc. The results of (near-) synonymy, that thesaurus induction techniques show, should actually be matched against the richer model of thesauri. However, that classification detail cannot be easily achieved with purely statistical techniques, as precision hardly exceeds 40% with the best model.

In order to make thoroughly clear what are the potential benefits that might derive from bootstrapping thesauri, and formalizing them, we propose a novel technique to automatically acquiring thesauri from the analysis of the structure and content of wikis. Our work is contextualized in the area of research aimed at extracting knowledge bases from Wikipedia (see for example the DBPedia [1] and YAGO [21] projects).

In contrast to the previous corpus-based approaches, our technique is able to identify senses for each term, and then to extract (near-) synonyms specifically for each different sense. Another major limitation of previously proposed approaches is their limited lexical coverage. In fact, to measure the distributional similarity between two words, it is necessary to collect multiple occurrences of those words in a given corpus to define reasonably good context representation. This is particularly problematic when dealing with domain specific terms, whose occurrences are somewhat rare. In contrast, our method is scalable and ensures high coverage, as it relies on two large-scale resources: Wikipedia and the Web 1T 5-gram corpus [2].

Our technique is based on two fundamental linguistic properties characterizing paradigmatic relations between words: the *domain restriction hypothesis* and the *lexical substitutability*. The former states that paradigmatic relations (e.g., synonymy and hyperonymy) holds mainly among terms belonging to the same domain, where domains represent regions in the language characterized by topical similarity and, in our case,

[1] http://www.w3.org/TR/skos-reference/skos.rdf

modeled by latent semantic analysis (LSA). The latter claims that terms are synonyms if they can be substituted in a particular context preserving the original meaning. Relying on these assumptions, we adapted the lexical substitution technique proposed by [11] to the problem of finding synonymous terms for terms in context. For example, the term *book* in the sentence "All Gold Canyon is a book written by Jack London" can be substituted with *novel* preserving the original meaning. It means that *book*, in this particular sense, is a (near-) synonym of *novel*.

Another innovation of our approach is that we enhance the thesaurus induction process by relying on the existing conceptual structure provided by Wikipedia, assuming that each article identifies a different concept that can be expressed by one or more terms. Our goal is then to discover sets of (near-) synonymous terms describing that concept. To this end, we extract the set of candidate (near-) synonyms from the Wikipedia article and its incoming wiki links. Then, we choose all those terms belonging to the domain of the article by applying LSA similarity. Finally, we rank each term by applying the lexical substitution technique. The result is a set of synonyms that can be used as a thesaurus entry for the concept.

We apply this method to detect all possible synonyms for subset of the most frequently visited articles in Wikipedia, and we evaluated our approach by comparing the results with respect to the corresponding synonym lists provided by the Oxford American Writers Thesaurus, after an handcrafted mapping of these articles to the corresponding dictionary senses. Experiments show that our method is accurate and improves the baseline approach.

The rest of the paper is structured as follows. In Section 2, we identify the analogies between the structures of wikis and thesauri. Section 3 describes the two basic building blocks of our algorithm for detecting paradigmatic relations: semantic domains and lexical substitutability. Section 4 reports the experimental results, Section 5 describes a reengineering pattern to formalize linguistic thesauri in OWL-RDF, Section 6 presents related work, and Section 7 concludes the paper suggesting some interesting applications of our method that we are going to explore in the future.

## 2 Deriving Conceptual Structures from Wikis

The wiki technology is changing the Web, leading to the success of Web 2.0 applications, such as Wikipedia. Wikis are currently adopted in many applications, including corporate knowledge management systems, collaborative development of lexical and encyclopedic resources, social networks, e-learning systems, and so on. The attractive feature of wikis is that their users are implicitly asked to collaboratively create a shared conceptual space, where each concept is described by a Web page mostly composed by highly focused natural language text concerning that concept. When a new concept of interest for the domain of the wiki is identified, the user creates an "empty" page where other users or her/himself can explain it in detail. On the other hand, when the concept of interest is already in the wiki, the user can link to it. The anchor text of the link may be a synonym or an equivalent expression of the term adopted to name the target page describing the concept.

Therefore, the wiki can be used as a basis to derive a network of relations between concepts (the Web pages) and terms (anchor texts of the internal links present in other wiki pages). More formally, we can describe the conceptual structure derived by a wiki as follows. Let $C$ be the set of concepts, and $T$ be the vocabulary of terms. From each wiki page we derive a concept, while terms, which are anchor texts of links to the page, become elements of $T$. Each wikilink describes a relation $M(t, c)$ between a term and a concept. For a given term $t_i$, the set of its possible senses is $S(t_i) = \{c_j \in C | M(t_i, c_j)\}$. For a given concept $c_i$, the set of its possible lexicalizations is returned by $L(c_i) = \{t_j \in M(t_j, c_i)\}$. The number of links connecting a given term to a given concept can be used to infer a prior probability for the term $t_i$ having sense $c_j \in S(t_i)$, by adopting the Formula 1.

$$P(c_j|t_i) = \frac{|M(t_i, c_j)|}{|M(t_i, c_z)|c_z \in S(t_i)|} \tag{1}$$

The structure described above is a thesaurus, as synonyms are represented and connected to concepts. In fact, [19] and [16] followed this approach to extract a thesaurus out of the wikilink structure.

On the other hand, wikis are collaboratively developed resources, most of the time composed very quickly by non-professional users. As a consequence, errors and bad practices are very frequent. For example, the article describing the concept *book* is linked to by the following terms: *monograph*, *manual*, *work*, *novel*, *booklet*, *father*, *guide*, *literary work*, *textbook*, *folio*, *books*, *treatise*, *print*, and *bookshop*, *book*. But, for example, only *manual*, *novel*, and *treatise* are listed as synonyms in the Oxford American Writers Thesaurus. The list obtained from wiki structure supplies new synonyms like *textbook*, but it introduces obvious erros like *father* as well.

Therefore, the use of the Wikipedia structure is not a viable solution for inducing high quality thesauri, even if it provides a rich starting point. In particular, we identified the following limitations.

- Ttypographical mistakes in the anchor texts. E.g., *politik* linked to `wiki/Politics`.
- Wrong links assigned to terms. E.g. terms *Article* and *Money* were linked by someone to the wiki page `wiki/Philosophy`.
- Bad selection of the term to link to the wiki page. E.g., *where prohibited* was made an anchor text for `wiki/Book`.
- Coreferences linked to concepts instead of terms. E.g., *this guy* linked to `wiki/Pope`.
- Terms having different syntactic roles are actually referred to the same page. For example, *political* is an anchor for the page `wiki/Politics`.
- Many possible lexical variants of concepts are missing, as users typically do a search before deciding whether to create a new wiki page or linking to an existing one. E.g., *herb* is missing among the lexical variants of the concept *Plant*.

In the following sections, we propose a method relying on semantic domains and lexical substitution to solve all the above problems.

**Table 1.** Ranked lists of possible synonymous terms for the concept *book* extracted adopting different criteria (from top to bottom): Wiki links, ranked by frequency (baseline), nouns in the article text, ranked by frequency, union of wikilinks and terms ranked by LSA, and by the combined method exploiting lexical substitution and LSA

| Method | (Near-) Synonyms |
|---|---|
| Wiki Links | book, monograph, manual, work, novel, booklet, father, guide, literary work, textbook, folio, books, treatise, print, bookshop, book ’ |
| Page Terms | ANSI, Abaca, Ages, . . . , E-book, book, bookbinder, . . . , novel, novels, number, numbers, . . . , purposes, queen, quo, quotation, . . . , stem, stems, stock, stories, story, . . . , values, varies,vellum, vergilius, version, vine, vines, virgil, vitriol, vivarium, volume, word, . . . , year, young, yun-ui |
| LSA | Book, book, Books, books, story, Reading, reading, Publication, publication, reader, Author, author, stories, readers, Publishers, publishers, Novel, novel, reprint, work, . . . |
| Lexical Subst. | book, story, magazine, novel, publication, chapter, writing, new, people, print, work, literature, reading, time, author, young, history, edition, collection, character, link, page, place, reference, english, press, review, note, account, kind, publisher, . . . |

## 3 Discovering Synonymy

We aim at capturing two characteristic properties of paradigmatic relations for synonymy detection:

- *Domain properties*: if a semantic relation between two terms $X$ and $Y$ holds, both $X$ and $Y$ tend to belong to the same semantic domain [14].
- *Lexical substitutability*: if two terms $X$ and $Y$ are paradigmatically related, they can be mutually substituted in text preserving the meaning.

In the following two subsection we describe each of them and we show their exploitation for thesauri induction.

### 3.1 The Domain Restriction Hypothesis

Semantic domains are at the basis of lexical semantics. Domain properties, also known as domain restriction hypothesis, are illustrated in Figure 1, where the probability for two terms to be related in WordNet [9] by a paradigmatic relation is contrasted to their domain similarity, measured by computing the cosine similarity between their corresponding vectors in the LSA space.

The main advantage of adopting semantic domains for thesauri extraction is that they allow us to impose a domain restriction on the set of candidate pairs of related terms. In fact, semantic relations can be established mainly among terms in the same semantic domain, while terms belonging to different fields are mostly unrelated.

Domain similarity between terms can be captured by measuring the LSA similarity between them. For our purposes, we defined a domain model by unsupervised learning on the Wikipedia text, following the procedure described in [13]. $\boldsymbol{c_i}$ and $\boldsymbol{t_j}$ are defined in the LSA space, estimated by the Singular Valued Decomposition for the term by

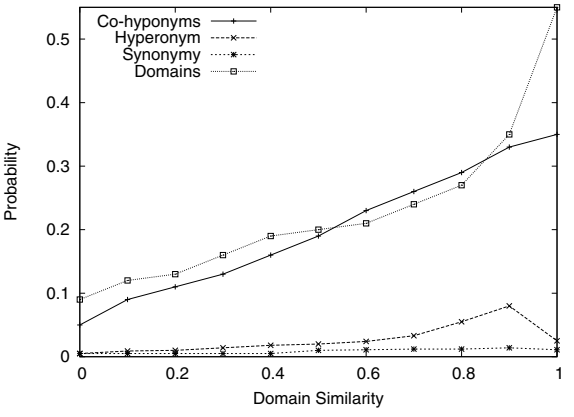



**Fig. 1.** Probability of finding paradigmatic relations in WordNet contrasted to domain similarity estimated in the British National Corpus [14]

document matrix extracted from the subset of the 500,000 most visited pages of the English Wikipedia. As a result, the similarity between a concept $c_i$, described by the corresponding Wikipedia article, and a term $t_j$, can be estimated by $cos(\boldsymbol{c_i}, \boldsymbol{t_j})$[2].

Domain similarity is then used to identify sets of candidate terms to fill the thesaurus entry representing each concept $c$. We first collect a set of the candidate lexicalizations $L(c) \subset T$ for a target article (concept) $c$ by merging the results of the following two strategies:

1. Selecting all nouns extracted from the analysis of the text in the Wikipedia article associated to $c$. For example the second row of Table 1 shows the list of terms obtained after the analysis of the article corresponding to the concept *book*.
2. Selecting words used to link the target page from other articles, i.e. the lexicalizations $L(c)$ extracted by the wiki structure described in Section 2. For example, the first row of Table 1 shows the list of terms obtained from the wikilinks pointing to the article *book*. We adopted this strategy to define our baseline.

Then, we ranked all terms in the set $t_i \in L(c)$ according to their LSA similarity with respect to the target concept $cos(\boldsymbol{c}, \boldsymbol{t_j})$ , and we filtered out all those terms having similarity below a threshold, that we fixed to 0.5 in all experiments described in this paper. Using such threshold we were able to filter out up to 80% of the terms in the set. This step is crucial to boost the efficiency of the synonymy induction step, as the final ranking, based on lexical substitution, is computationally intensive and constitute a bottleneck for the overall efficiency of our method.

### 3.2 Finding Synonyms by Lexical Substitution

As a second step, the candidate terms having high domain similarity with respect to the target concept are re-ranked by lexical substitution. The lexical substitution is a textual

[2] The SVD process exploited 400 dimensions at unsupervised learning time and took less than 2h on a dual core processor exploiting 4GB of RAM.

entailment subtask in which the system is asked to provide one or more terms that can be substituted to $w$ in a particular context $H_w = H^l w H^r$, generating a sentence $H_e = H^l e H^r$ such that both $H_w \rightarrow H_e$ and $H_e \rightarrow H_w$ hold, where $H^l$ and $H^r$ denote the left and the right context of $w$, respectively.

[17] recently proposed this task in the context of the SemEval-2007 evaluation campaign. The state-of-the-art system is quite accurate and totally unsupervised [12], as it only exploits an existing dictionary and a huge language model. For each target word, a set of possible candidate words is extracted by the dictionary, and then substituted to the target word generating new sentences. Then, the "plausibility" of each sentence is measured by looking for the frequency of the generated sentences in the Web 1T 5-gram corpus, and the candidate words are then ranked according to that plausibility score.

In order to apply this technique to our problem we cannot rely on the availability of existing dictionaries containing candidate terms, as our goal is to discover such terms. In addition, we are not interested in substituting terms into a single occurrence of a word in context, but we are rather interested in the combined results coming from the substitution of candidate words into a large set of occurrences, i.e., all wiki links pointing to a particular article.

For a given concept $c$ and a set of its candidate lexicalizations $L(c)$, our substitution algorithm works as follows:

*Step 1.* For each concept $c$, we collect all the sentences $S$ containing a wiki link $i$ pointing to the Wikipedia article associated to $c$. For instance, a sentence containing a link to the concept *book* is "The most printed *book* in history"

*Step 2.* Then, for each sentence $s \in S$, we derive a hypothesis phrase by replacing the link $i$ with a candidate synonym $j \in L(c)$. For instance, from the previous example, we derive "The most printed *novel* in history".

*Step 3.* For each hypothesis phrase $h_j$, we calculate a plausibility score $v_j$ using a variant of the scoring procedure defined in [11]. In our case, $v_j$ is given by the sum of the pointwise mutual information (PMI) of all the n-grams ($1 < n \leqslant 5$) that contain $j$ divided by the self-information (SI) of the right and left contexts. PMI and SI are defined by the Equations 2 and 3, respectively.

$$PMI = \log \frac{p(t_1, t_2 \ldots t_l)}{p(t_1)p(t_2)\ldots p(t_l)} = v_j \tag{2}$$

$$SI = -\log p(t_1, t_2 \ldots t_l) = v_j \tag{3}$$

Dividing by the self-information allows us to penalize the hypotheses that have contexts with a low information content, such as sequences of stop words. The frequency of the n-grams is estimated from the Web 1T 5-gram corpus. For instance, from the hypothesis phrase "The most printed *novel* in history", we generate and score 10 n-grams, some of them are: "The most printed *novel* in", "printed *novel* in history", "*novel* in history"

*Step 4.* Finally, to obtain an overall score $v_t$ for the term $t$, we sum the scores obtained by the substitution in all sentences, as defined in Equation 4.

$$v_t = \sum_{l=1}^{N} v_l, \tag{4}$$

where $N$ is the number of sentences containing a wiki link to the target article $c$.

We used the this procedure to rank all candidate terms selected in the domain restriction phase. The top ranked terms for each concept provide a different thesaurus entry, associated to the corresponding Wikipedia article.

## 4 Evaluation

For the evaluation, we selected the target concepts among the most visited articles in Wikipedia, and we automatically extracted thesaurus entries for each of them by contrasting our method with a baseline, provided by a replication of the method described in [19].

The list of concepts evaluated so far is the following: *Sex, Pornography, Love, Politics, Book, Earth, Map, Computer, Cat, Game, Dictionary, Ejaculation, Lesbian, Sport, Cancer, God, Virus, Animal, Heroin, Horse, Film, Human, Condom, Snake, Flower, Evolution, Beer, Statistics, Religion, Communication, Management, Insomnia, Erection, Death, Earthquake, People, Insurance, Graffiti, Research, Puberty, Cannabis, Marriage, Gun, Socialism, Narcissism, Twilight, Sleep, Architecture, Stroke, Leaf.*

### 4.1 Gold Standard

We handcrafted a gold standard thesaurus entry by mapping each concept into the appropriate sense of the Oxford American Writers Thesaurus. For simplicity, we did not consider multi-words and parts of speech other than nouns. Below, we report a sample of the gold standard: the first word is the name of the Wikipedia article, and the following words are its synonyms extracted from the appropriate sense in the dictionary.

**book** hardback, storybook, volume, treatise, publication, e-book, manual, novel, title, tome, paperback, anthology
**earth** planet, world, globe
**computer** laptop, terminal, mainframe, pc, desktop
**dictionary** wordbook, lexicon, glossary, thesaurus
**cancer** lymphoma, sarcoma, melanoma, malignancy, myeloma, tumor, carcinoma
**virus** contagion, disease, bug, infection

The output of our thesaurus induction method is a ranked list of terms for each concept, that we compared with the gold standard, reporting precision and recall figures.

### 4.2 Baseline

As a baseline method, we implemented the heuristic described in Section 2. We extracted all terms contained by wikilinks pointing to each article, and we ranked them by frequency (i.e., the number of times they have been used to link to the target article). For each term, we computed the precision at different levels of recall, following an information retrieval paradigm.

The micro-average figures, reported in Figure 2, show that this method is not very accurate. In fact, the maximum precision is close to 0.2 at very low recall. The break-even point is 0.18.

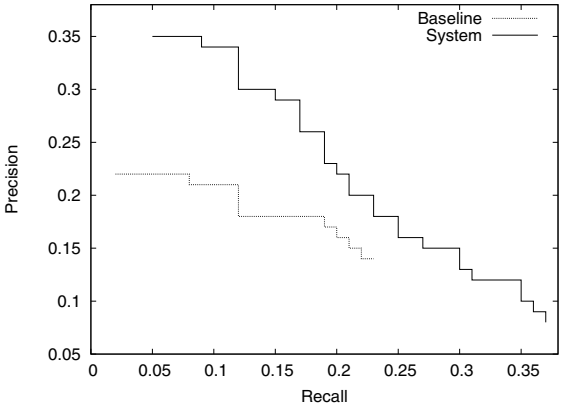



**Fig. 2.** Precision/Recall curves for the baseline and our system

### 4.3 Advanced Methods

We built the domain model from the 200,000 most visited Wikipedia articles. After removing terms that occur less than 5 times, the resulting dictionaries contain about 300,000 terms. We used the SVDLIBC package[3] to compute the singular value decomposition, truncated to 100 dimensions. All the experiments were performed using the LIBSVM package [5].

Then, for each article corresponding to a concept in the gold standard, we adopted the bag of word vector described above to provide a list of candidate terms, that we merge to the set of words returned by the baseline method. Then, we filtered out from this set all those terms having a cosine similarity with the target article's vector below 0.5. This threshold allows us to drastically reduce the number of candidates but preserving with high probability the (near-) synonyms. The third row of Table 1 exemplifies the list of terms obtained so far for the concept *book*.

Then we collected all sentences containing wiki links to the target article, and we generated new sentences by substituting all candidate terms collected above. Finally, we computed a plausibility score for each of those sentences, on the basis of which we derived the final score for each candidate synonymous term by adopting Formula 4. The resulting sets are then ranked by applying the lexical substitution algorithm described in Section 3.2.

Results are reported in Figure 2, we compare the two methods by means of the precision/recall curves. The lexical substitution method combined with the domain restriction hypothesis significantly boosts the quality of the baseline..

### 4.4 Qualitative Evaluation

From a qualitative analysis of the results, it seems that most terms that do not match the gold standard entry are not actually weird errors, but rather they follow under the definition of (near-) synomymy. For example, looking at the fourth row of Table 1 and

[3] `http://tedlab.mit.edu/~dr/svdlibc/`

comparying it to the gold standard reported in Section 4.1, many high quality terms such as print, monograph, booklet, folio, guide, textbook, treatise, do not match the gold standard. However, they include hyperonymy, hyponymy and co-hyponymy terms. Such terms are very useful in developing computational thesauri, especially as far as applications like semantic information retrieval are the final goal. For example, the terms acquired so far can be used to improve search in Wikipedia itself, or to expand the lexical part of DBpedia, bridging knowledge to language in a much smarter way.

## 5 RDF-izing Linguistically-Induced Thesauri

A linguistically-induced thesaurus is currently understood as a "poor" thesaurus, which only contains concepts and (preferred or alternate) labels. The linguistic semantics, by which a concept is equivalent to a set of near-synonymous terms can be represented in different ways, and we show here three possible solutions: one based on SKOS, one on WordNet OWL, and a hybrid one.

However, before exemplifying the solutions, we should note that the assumption of near-synonymy between the terms (found as statistically meaningful), associated with a concept (i.e. the entities Wikipedia pages are about) can be counter-intuitive for lexical semantics.

Near-synonymy is expected to catch a special kind of similarity between terms, which makes them *substitutable* in a certain class of contexts, modulo certain local constraints. For example, *book* can be substituted with *volume* in many generic contexts (referring to physical objects), or with *script* in more specialistic contexts (drama playing), but a lexicographer would hardly make it substitutable with *magazine* or *writing*: the last two would be considered similar in a different sense (i.e. hyponymy, overlap, or association).

The notion of substitutability assumed in thesaurus induction is broader than the one assumed in lexical semantics, and the representations we present have to be adjusted to this broader notion, which also covers what lexicographers would classify as e.g. hyponymy or association.

The first candidate representation consists in using the relation between a skos: Concept and its `skos:label`(s): this representation basically uses literals as values for the skos:label property, and therefore has problems when we need to provide attributes to the terms. In a new extension of SKOS[4], a label is represented as an individual from the class skosxl:Label, with which literal values can be associated. This gives also room for representing a large set of data about preferred or alternative labels: creators, dates, provenance, etc.

For example, the relation between: **Book** and {book, novel, magazine, textbook, publication, writing} can be represented as a set of RDF triples:

```
:Book a skos:Concept ; skosxl:prefLabel :book ;
  skosxl:altLabel :novel ; skosxl:altLabel :magazine ;
skosxl:altLabel :textbook ; skosxl:altLabel :writing ;
                          skosxl:altLabel :publication
```

with `:book a skosxl:Label`, etc.

[4] http://www.w3.org/TR/skos-reference/skos-xl.rdf

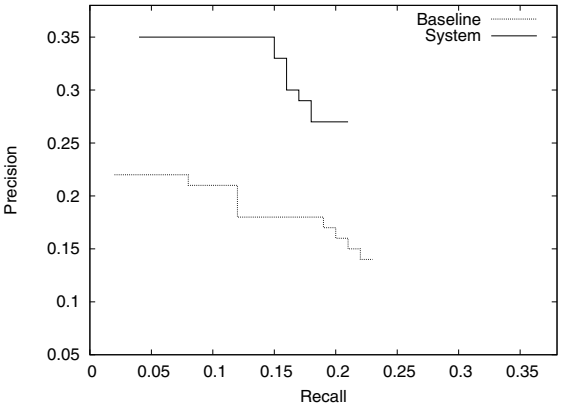



**Fig. 3.** Precision/Recall curves for the baseline and our system used to re-rank the baseline

With this representation, the semantics of thesaurus induction (concepts against relevant associated terms) is represented, since `skos:Concept` and `skosxl:Label` are disjoint classes, but it is confusing for a user to find "labels" of e.g. the concept *Book* that include *novel*, *magazine*, etc., since most people is accustomed to the lexicographic near-synonymy assumption. The intuition for an English speaker would possibly be that the concept `Book` has more specific senses like *novel* and *textbook*, more generic ones like *publication*, overlapping terms like *magazine*, and related senses like *writing*. The most correct representation would be therefore to use the generic `skos:semanticRelation` between concepts, which can be eventually specialized as broader, related, etc. after automatic or manual evolution. In this case, either concepts or terms from thesaurus induction should be classified as `skos:Concept`(s).

However, in this way we are not getting much to the point, since in order to preserve the lexicographic intuition, we lose the original thesaurus induction semantics.

An alternative to using SKOS is to reuse a meta-model that is closer to computational linguists' intuition, i.e. the WordNet OWL vocabulary[5]. In that case, the class `wordnet:Synset` has the intended meaning of a set of near-synonymous word senses, which are linked through the `wordnet:containsWordSense` relation. A `wordnet:WordSense` is on its turn linked to words through the `wordnet:word` relation to wordnet:Word. This representation seems better, since it allows to representing terms either as word senses or words, and still retains the possibility to link them to synsets (concepts). However, even here the exceptions to intuitive near-synonymy mentioned above are still valid: some relations look closer to `wordnet:hyponymOf`, others to `wordnet:hypernymOf`, others do not have any correspondence at all. The relation `wordnet:containsWordSense` has a meaning that conflicts with e.g. `wordnet:hyponymOf`, and in WordNet no relation represents e.g. the one between *Book* and *writing*.

[5] `http://www.w3.org/2001/sw/BestPractices/WNET/tf.html`

A third alternative consists in building an ad-hoc vocabulary for thesaurus induction semantics. This solution can be implemented by merging the positive aspects of both SKOS and OWL-WordNet: concepts are represented as `wordnet:Synsets`, terms as `wordnet:Words` mapped to `skosxl:Label`(s), while new individuals for each word sense of a word can be created, and linked to their synset through the `skos:semanticRelation` relation, for example:

```
:Book a wordnet:Synset ;
skos:semanticRelation :wordsense-book ;
skos:semanticRelation :wordsense-novel ;
skos:semanticRelation :wordsense-magazine ;
skos:semanticRelation :wordsense-textbook ;
skos:semanticRelation :wordsense-publication ;
skos:semanticRelation :wordsense-writing
```

with `:wordsense-novel a wordnet:WordSense ;` and `wordnet:word:novel` which is both a `wordnet:Word,` a `skosxl:Label`, etc.

The third hybrid representation can also be implemented by designing a vocabulary that catches exactly the thesaurus induction semantics, and then by mapping the vocabulary to SKOS and OWL-WordNet.

## 6 Related Work on Thesaurus Construction

Among the approaches to semantic relatedness evaluation and thesauri induction there are the pattern-based approaches [15], as well as distributional measures of lexical similarity based on word co-occurence in documents [8] or in syntactic contexts [7]. In [22] a thesaurus is generated by usage of a syntactically constrained Vector Space Model (VSM). In this approach each dependency from a set of predefined syntactic dependencies is represented by different subsets of VSM. Sets of words related to a given source word are found distinctly in each of the VSM and then overlapped.

Many of the latest approaches to the task are based on usage of Wikipedia as the source of background knowledge. Among the first were [20], who exploited the Wikipedia pages texts and category structure to calculate text overlap, taxonomy and information based relatedness measures. Given two words **i** and **j** they retrieved the corresponding Wikipedia pages (or articles), to content of which text overlap similarity based measures could be applied. The pages were retrieved just by querying a wiki page named **i** or **j**. In the case when one or both retrieved pages are disambiguation pages, disambiguation heuristic was applied. Exploiting sets of categories $C_i$ and $C_j$, to which words belong, [20] obtained least common subsumer for each category pair to be able to calculate information based measures. Finally, sets of paths between the $C_i$ and $C_j$ pairs gave input to taxonomy similarity measures. The correlation with human relatedness evaluation is 0.19-0.48.

[10] proposed Explicit Semantic Analysis (ESA) for measuring semantic relatedness of both words and texts. They build the semantic interpreter, which maps input text/words into a weighted vector of relevant Wikipedia concepts. The relevant

Wikipedia concepts are titles of Wikipedia pages, which contain the words/texts. The semantic relatedness of two texts/words is computed then applying cosine metric to corresponding weighted concept vectors.The correlation with human relatedness evaluation is 0.75.

[18] measure semantic relatedness between the concepts using Wikipedia link structure. Given two terms, they obtain all wikipedia articles to which these terms may refer and compute semantic relatedness between all possible pairs of articles combining two relatedness measures. First one is similar to tfidf , but uses wikipedia links and pages instead of terms and documents. The second is based on the Normalized Google Distance, which takes into account term co-occurences of terms on the web-pages, where links and wikipedia articles are used instead of terms and web-pages correspondingly. The terms relatedness is then evaluated as a combination of the highest relatedness achieved among pairs of articles with term co-occurence frequency in wikipedia anchor texts. The correlation with human relatedness evaluation for words is 0.78.

[19] and [16] propose to use Wikipedia directly as a building material for an association thesaurus. Therefore, they do not encounter the disambiguation problem. In [19], the thesaurus is constructed using *pfibf* (path frequency inversed backward frequency), which calculates the relativity between two concepts I an J represented by two articles $V_i$ and $V_j$. *pfibf* exploits information about the number of paths from $V_i$ to $V_j$, the lengths of these paths and the number of backward links to articles. The latter is used to penalize popular articles, which are highly related to a big number of articles. The concept precision among 10-30 top terms varies within limits 66.7-85.9. In the later work ([16]) they propose a much faster associative thesaurus construction method based on link co-occurrence analysis. The links are considered to be concepts, two links are said to be the same even if they have different anchor texts. In order to calculate relatedness between two links each link is represented as a combined vector (cv). *Cv* is a linear combination of a *link* and *tfidf* vector. *Link* vector is a vector, whose positions are first-order co-occurrences of the given link with other links in the Wikipedia. The first-order co-occurrence of two links is computed using the co-occurrence frequencies of these links with all other links over Wikipedia. Accuracy measure is 0.59-0.69.

In addition, different techniques for acquiring terms, synonyms and semantic relations have been presented in the field of ontology learning [4,3].

## 7 Conclusion and Future Work

In this paper, we presented an innovative method for inducing thesauri from Wikipedia. Our method leverage on the Wikipedia structure to find out target concepts for the thesaurus, which is in itself an innovation if compared to traditional method relying on words only. In addition to that, we applied an innovative method for detecting possible lexicalizations of those concepts, showing that both precision and recall, in this settings corresponding to the quality of the thesaurus entry and to its lexical coverage, can be significantly boosted. A qualitative assessment of the results shows that our method is actually able to find (near-) synonymous terms that can be used to automatically populate thesaurus entries if the strict requirement of synonymy is relaxed.

For the future, we are going to perform an extensive analysis of the full Wikipedia aimed at extracting lexical expressions for each concept represented by a Wikipedia

page. In fact, our method is largely scalable: with the present (prototypical) implementation we where able to process each page in few seconds by using a standard PC. In addition, we are going to extract thesauri for many languages in Wikipedia, as our method is totally language independent and only requires part of speech tagging of texts.

Exploiting such a large scale thesaurus into applications seems very promising, as the main weakness of knowledge based approach for retrieval is very often coverage. For example, the extracted synonyms for each page can be used for improving the index of Wikipedia for (cross lingual) information retrieval.

Last, but not least, our technique provides a explanatory application of two fundamental properties of lexical semantics, the domain restriction hypothesis and the lexical substitutability principle, showing that a combination of them can be almost always used to figure out solutions to problems involving lexical semantics, providing yet another experimental proof of their universal validity.

## Acknowledgments

Claudio Giuliano is supported by the X-Media project (http://www.x-media-project.org), sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978 and the ITCH project (http://itch.fbk.eu), sponsored by the Italian Ministry of University and Research and by the Autonomous Province of Trento. Aldo Gangemi is supported by the NeOn project (http://www.neon-project.org) and the IKS project (http://www.iks-project.eu), sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-2005-027595 and IST 231527, respectively. Kateryna Timoshenko is supported by the ITCH project.

## References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: 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)
2. Brants, T., Franz, A.: Web 1T 5-gram corpus version 1.1. Linguistic Data Consortium, Philadelphia (2006)
3. Buitelaar, P., Cimiano, P. (eds.): Ontology Learning and Population: Bridging the Gap between Text and Knowledge. Frontiers in Artificial Intelligence and Applications, vol. 167. IOS Press, Amsterdam (2008)
4. Buitelaar, P., Cimiano, P., Magnini, B. (eds.): Ontology Learning from Text: Methods, Evaluation and Applications. Frontiers in Artificial Intelligence and Applications, vol. 123. IOS Press, Amsterdam (2005)
5. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm

6. Curran, J.R., Moens, M.: Improvements in automatic thesaurus extraction. In: Proceedings of the ACL 2002 Workshop on Unsupervised Lexical Acquisition, Philadelphia, Pennsylvania, USA, July 2002, pp. 59–66. Association for Computational Linguistics (2002)
7. Dagan, I.: Contextual Word Similarity, ch. 19, pp. 459–476. Marcel Dekker Inc., New York (2000)
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
9. Fellbaum, C.: WordNet. An Electronic Lexical Database. MIT Press, Cambridge (1998)
10. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1606–1611 (2007)
11. Giuliano, C., Gliozzo, A., Strapparava, C.: Fbk-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In: Proceedings of the Fourth International Workshop on Semantic Evaluations, SemEval 2007, Prague, Czech Republic, June 2007, pp. 145–148 (2007)
12. Giuliano, C., Gliozzo, A.M., Strapparava, C.: Fbk-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In: Fourth International Workshop on Semantic Evaluations, SemEval 2007. ACL (2007)
13. Gliozzo, A.: Semantic Domains in Computational Linguistics. PhD thesis, University of Trento (2005)
14. Gliozzo, A., Pennacchiotti, M., Pantel, P.: The domain restriction hypothesis: Relating term similarity and semantic consistency. In: Proceedings of NAACL-HLT (2006)
15. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the Fourteenth International Conference on Computational Linguistics, Nantes, France (July 1992)
16. Ito, M., Nakayama, K., Hara, T., Nishio, S.: Association thesaurus construction methods based on link co-occurrence analysis for wikipedia. In: CIKM 2008: Proceeding of the 17th ACM conference on Information and knowledge management, pp. 817–826. ACM, New York (2008)
17. McCarthy, D., Navigli, R.: Semeval-2007 task 10: English lexical substitution task. In: Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007), Prague, Czech Republic, June 2007, pp. 48–53. Association for Computational Linguistics (2007)
18. Milne, D., Witten, I.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence, WIKIAI 2008 (2008)
19. Nakayama, K., Hara, T., Nishio, S.: A thesaurus construction method from large scaleweb dictionaries. In: 21st International Conference on Advanced Networking and Applications (AINA 2007), pp. 932–939 (2007)
20. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: AAAI. AAAI Press, Menlo Park (2006)
21. Suchanek, F., Kasneci, G., Weikum, G.: Yago - a large ontology from wikipedia and wordnet. Elsevier Journal of Web Semantics 6(3), 203–217 (2008)
22. Yang, D., Powers, D.M.: Automatic thesaurus construction. In: Dobbie, G., Mans, B. (eds.) Thirty-First Australasian Computer Science Conference (ACSC 2008), CRPIT, Wollongong, NSW, Australia, vol. 74, pp. 147–156. ACS (2008)

# Efficient Semantic-Aware Detection of Near Duplicate Resources

Ekaterini Ioannou, Odysseas Papapetrou,
Dimitrios Skoutas, and Wolfgang Nejdl

L3S Research Center/Leibniz Universität Hannover
{ioannou,papapetrou,skoutas,nejdl}@L3S.de

**Abstract.** Efficiently detecting near duplicate resources is an important task when integrating information from various sources and applications. Once detected, near duplicate resources can be grouped together, merged, or removed, in order to avoid repetition and redundancy, and to increase the diversity in the information provided to the user. In this paper, we introduce an approach for efficient semantic-aware near duplicate detection, by combining an indexing scheme for similarity search with the RDF representations of the resources. We provide a probabilistic analysis for the correctness of the suggested approach, which allows applications to configure it for satisfying their specific quality requirements. Our experimental evaluation on the RDF descriptions of real-world news articles from various news agencies demonstrates the efficiency and effectiveness of our approach.

**Keywords:** near duplicate detection, data integration.

## 1 Introduction

A plethora of current applications in the Semantic and Social Web integrate data from various sources, such as from the local file system, from other applications, and from the Web. In this open environment, information is often spread across multiple sources, with the different pieces being overlapping, complementary, or even contradictory. Consequently, a lot of research efforts have focused on data integration and data aggregation from various sources, and especially for data from the Web. A specific problem that arises in this direction is the detection of *near duplicate* information coming from different sources or from the same source in different points in time. This is a crucial task when searching for information, so that resources, such as Web pages, documents, images, and videos, that have been identified as near duplicates can be grouped together, merged, or removed, in order to avoid repetition and redundancy in the results.

As a typical example, consider a news aggregation service which monitors and aggregates articles from a large number of news agencies. Near duplicates naturally occur in this scenario, since many of these agencies are expected to have articles reporting on the same news stories, which involve the same people,

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 136–150, 2010.

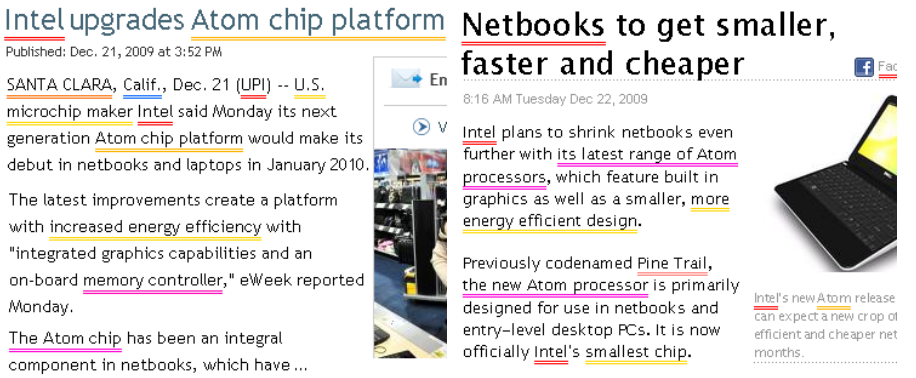



**Fig. 1.** Two near duplicate news articles. The underlined text shows the identified entities that are described in RDF data of each news article.

events, and locations. Moreover, news agencies often update their articles or republish articles that were published somewhere else, possibly with slight changes. For instance, national news agencies often republish articles which were originally published by a commercial newspaper, and vice versa. In most cases, this republishing also introduces small changes in the news articles, for instance a comment that this article is a republishing, correction of spelling mistakes, an additional image, or some new information. The goal of the news aggregation service is to present to the users a unified view of the articles of all news agencies. To achieve this, it needs to detect the near duplicate news articles and to handle them accordingly, for example by filtering them out or grouping them together.

Detecting near duplicate resources requires computing their similarity and selecting those that have a similarity higher than a specified threshold (typically defined by the application based on its goals). Hence, there are two main issues to be addressed: (a) how to compute the similarity between a pair of resources, and (b) given that near duplicate detection is a task that often needs to be performed online, how to efficiently identify resources that are similar enough to qualify as near duplicates, without performing all the pairwise comparisons.

Regarding the first issue (i.e., similarity of two resources), comparing two resources based only on their content may not be sufficient. For instance, two Web pages or two news articles in the aforementioned example written by different authors with different writing styles, may not have a very high similarity when compared using a bag of words representation, while they may still refer to the same entities and qualify as near duplicates (see Figure 1). However, in the Semantic Web, resources are annotated with metadata in the form of RDF statements. Such annotations can be made manually or (semi-)automatically using tools for natural language processing and information extraction, such as the Calais Web Service [18] or metadata extractors [16], which identify and extract from unstructured text entities, facts, relationships, and events, and provides them in the RDF format. This structured and semantically rich information can be exploited to more accurately identify near duplicate resources. Existing approaches that deal with

the problem of efficiency in similarity search, e.g., [2,10,15], do not operate on structured data (see Section 5).

Our goal is to perform semantic-aware and efficient detection of near duplicate resources by combining indexing schemes for similarity search with the RDF representations of the resources. More specifically, our main contributions are as follows:

1. We introduce *RDFsim*, an efficient algorithm for detecting near duplicate RDF resources. In contrast to existing text-based techniques, our approach is able to more effectively identify near duplicate resources, using their RDF representations, and by considering not only the literals but also the structure of the RDF statements.
2. We provide a probabilistic analysis for the correctness of the algorithm, showing also how *RDFsim* is configured to satisfy the given quality requirements.
3. We describe an online system that we have implemented in order to test and illustrate our method for near duplicate detection on a large and continuously updated collection of news articles.
4. We experimentally evaluate the efficiency and effectiveness of our approach, using a real-world data set composed of RDF data extracted from recent news articles from various news agencies.

The rest of the paper is organized as follows. Section 2 introduces and explains the representation of resources and the indexing structure of *RDFsim*. Section 3 explains the process of querying for near duplicate resources, and discusses configuration of the *RDFsim* parameters. Section 4 presents an online system that applies our approach to detect near duplicate news articles, and reports the results of our experimental evaluation. Finally, Section 5 presents and discusses related work, and Section 6 provides conclusions and future work.

## 2 Representing and Indexing Resources

### 2.1 Overview

A resource in the Semantic Web is described by a set of RDF triples of the form $(subject, predicate, object)$, where *subject* is a URI identifying a resource, *predicate* is a URI representing a property of the resource, and *object* represents the value of this property, which can be either a literal or a URI identifying another resource. These triples form a graph, where the nodes correspond to subjects and objects, and the edges correspond to predicates. When a node is not identified by a URI (i.e., blank nodes), we use the node id information that is provided. Hence, each resource is represented by an RDF graph $R$, constructed from the RDF triples which describe this resource.

Let $\mathcal{R}$ be the set of all available resources, and $sim : \mathcal{R} \times \mathcal{R} \rightarrow [0,1]$ a function computing the similarity between two resources, based on their RDF graphs. We define near duplicate resources as follows.

**Definition 1.** *Given two resource descriptions $R_1$ and $R_2$, a similarity function sim, and a similarity threshold minSim, then these two resources are near duplicates if $sim(R_1, R_2) \geq minSim$.* ■

Given a potentially large set of resources $\mathcal{R}$, the problem we focus on is to efficiently identify all pairs of near duplicate resources in $\mathcal{R}$. A straightforward solution to this problem is to first perform a pairwise comparison between all the resources, and then to select those pairs having similarity above the given threshold. However, this is not scalable with respect to the number of resources, and hence not suitable for performing this task under time restrictions (e.g., online processing), or when the set of resources $\mathcal{R}$ is dynamic.

To address this problem efficiently, we need to avoid the pairwise comparisons of resources. For this purpose, we propose a method that relies on Locality Sensitive Hashing (LSH) [10]. First, each resource is converted into the internal representation used by *RDFsim*, which is then indexed in an index structure based on LSH. This index structure allows us to efficiently detect the near duplicates of a given resource, with probabilistic guarantees. The rest of this section deals with the representation and indexing of resources, while the process of finding the near duplicates of a given resource is described in Section 3.

### 2.2 Resource Representation

As explained in Section 1, our method emphasizes on semantic-aware detection of near duplicate resources, i.e., it operates on the RDF representation of the resources. As this information is often not available a priori, a pre-processing step may be required to extract semantic information for the resources. There are several tools that can be used for this purpose, such as the Calais Web Service [18] (see Section 4.2 for more details). Subsequently, ontology mapping methods can be applied to handle the cases where different vocabularies are used by different sources. In addition, some metadata may be deliberately filtered out by the application, as they may not be relevant to the task of near duplicate detection. For example, in the case of the news aggregation scenario, an article identifier assigned to the article by the particular agency publishing it should not be taken into consideration when searching for near duplicate articles.

Once the RDF graph describing the resource has been constructed, it needs to be transformed to a representation that is suitable for indexing in an index based on LSH, while preserving the semantic information for the resource. For this purpose, *RDFsim* applies a transformation of the RDF graph of each resource $R_x$ as follows: each RDF triple is represented as a concatenation of the predicate and the object. In the case that the object is a literal, then the predicate is concatenated with the literal. In the case that the object is itself the subject of another RDF triple, e.g., $R_y$, then the predicate is concatenated with the representation $rep(R_y)$ of $R_y$, which is generated recursively. During this recursive generation, cycles are detected and broken. This process is illustrated by the following example.

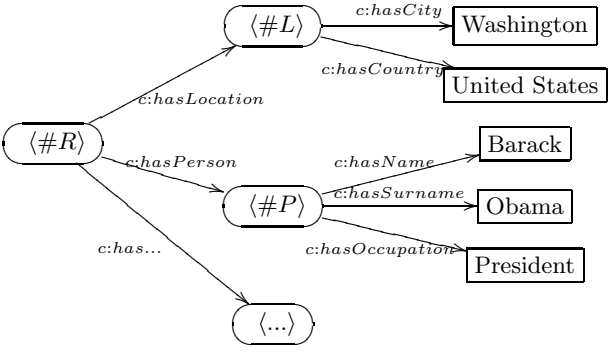



**Fig. 2.** Representation of resources takes into consideration the semantic structure

*Example 1.* Consider the RDF graph shown in Figure 2. The representation of the nodes $L$ and $P$ are the following:

$$rep(L) = \{\text{“c:hasCity, Washington”, “c:hasCountry, United States”}\}$$

$$rep(P) = \{\text{“c:hasName, Barack”, “c:hasSurname, Obama”,}$$
$$\text{“c:hasOccupation, President”}\}$$

Then, the representation of the resource $R$ is generated recursively using the representations of the resources under $R$ (e.g., $L$ and $P$) as follows:

$$rep(R) = \{\text{“c:hasLocation, L”, “c:hasPerson, P”, } \ldots\} \cup rep(L) \cup rep(P)$$

Notice that some resources may have large and complex RDF graphs (e.g., large documents), leading to very lengthy representations. However, this does not constitute a problem since these representations do not need to be maintained in main memory. Instead, the representation of each resource is only computed and used once, as an intermediate step for the purpose of hashing it in the index structure.

Along with the resource representation, our algorithm also needs a similarity method (see Definition 1) that is used for computing the similarity between two RDF representations. For the purpose of this work we apply one of the standard similarity measures, Jaccard coefficient. However, *RDFsim* and the underlying LSH index can incorporate other measures, and there have already been analytic results which enable LSH on different distance measures [6], for example for the cosine similarity.

### 2.3 Indexing Structure

The index used by *RDFsim* is based on the Locality Sensitive Hashing (LSH) approach of [10]. The main idea behind LSH is to hash points from a high dimensional space using a hash function $h$ such that, with high probability,

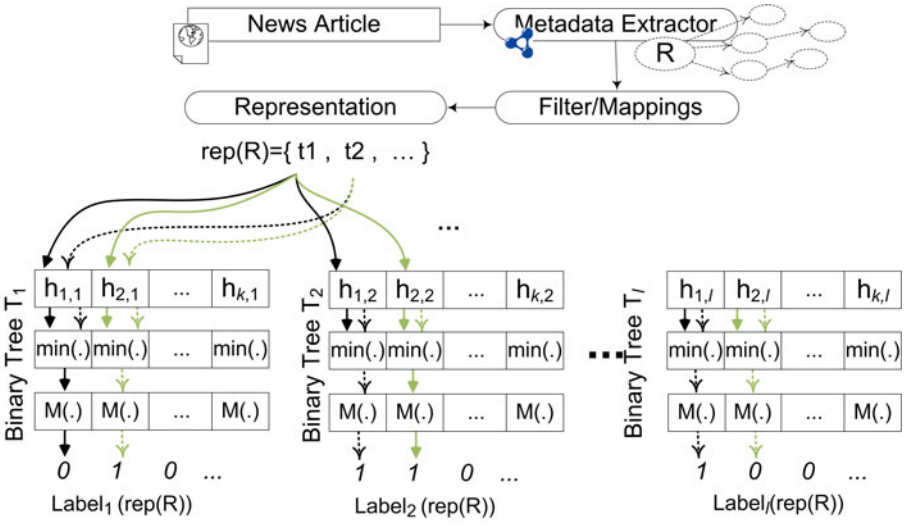



**Fig. 3.** An illustration of the process followed for generating the labels of RDF resources, which are used for inserting these resources into the indexing structure

nearby points have similar hash values, while dissimilar points have significantly different hash values, i.e., for a distance function $D(\cdot, \cdot)$, distance thresholds $(r_1, r_2)$, and probability thresholds $(pr_1, pr_2)$:

- if $D(p, q) \leq r_1$, then $Pr[h(p) = h(q)] \geq pr_1$
- if $D(p, q) > r_2$, then $Pr[h(p) = h(q)] < pr_2$

More specifically, we use an indexing structure $\mathcal{I}$ that consists of $l$ binary trees, denoted with $\mathcal{T}_1$,$\mathcal{T}_2$, …, $\mathcal{T}_l$. To each tree, we bind $k$ hash functions, randomly selected from a family of locality sensitive hash functions $\mathcal{H}$. We denote the hash functions bound to tree $\mathcal{T}_i$ as $h_{1,i}$, $h_{2,i}$, …, $h_{k,i}$.

Figure 3 shows the process we follow for indexing resources. When a new resource $R_x$ arrives, first its representation $rep(R_x)$ is computed as described above. Recall that $rep(R_x)$ consists of a set of terms (i.e., the elements of the set $rep(R_x)$). We compute $l$ labels of length $k$. Each label corresponds to a binary tree. *RDFsim* computes the label of $rep(R_x)$ for each tree $\mathcal{T}_j$ as follows:

- It hashes all the terms in $rep(R_x)$ using each hash function $h_{i,j}(\cdot)$ that is attached to the binary tree $\mathcal{T}_j$.
- It detects the minimum hash value produced by $h_{i,j}(\cdot)$ over all terms in $rep(R_x)$, denoted as $min(h_{i,j}(\cdot))$.
- It maps $min(h_{i,j}(\cdot))$ to a bit 0 or 1 with consistent mapping $\mathcal{M} \mapsto [0, 1]$. This resulting bit is used as the $i$'th bit of the label of $rep(R_x)$.

The same map $\mathcal{M}$ is used for all the binary trees. Any mapping function can be used, for example *mod* 2, as long as it returns 0 and 1 with equal probability.

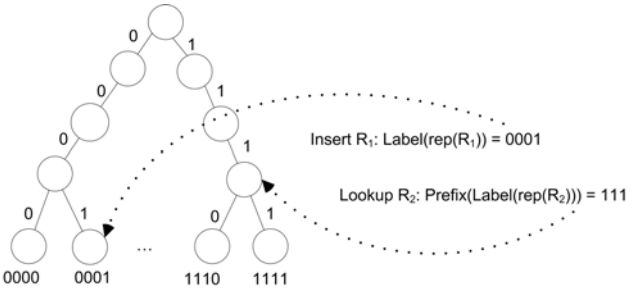



**Fig. 4.** Inserting and searching for resources in a tree of *RDFsim*

After computing the $l$ labels of a resource, the algorithm inserts the resource in the inverted index. Let $Label_i(rep(R_x))$ denote the binary label computed from $R_x$ for the binary tree $\mathcal{T}_i$. Then, $R_x$ is inserted in the tree using $Label_i(rep(R_x))$ as its path. For example, if $Label_i(rep(R_x)) = 0001$, then $R_x$ is inserted at the node with the specific path in tree $\mathcal{T}_i$ (see Figure 4).

## 3 Querying for Near Duplicate Resources

Executing a query for near duplicate resources is similar to the process described above for indexing a resource. Let $R_q$ denote the resource for which we want to search for near duplicates, and $minSim$ the minimum similarity between the query $R_q$ and another resource $R_p \in \mathcal{R}$ for considering the two resources as near duplicates. Our method provides a trade-off between performance and recall, expressed by the minimum probability $minProb$ that each near duplicate of $R_q$ is found.

First, we create the labels for the query $Label_1(\text{rep}(R_q))$, $Label_2(\text{rep}(R_q))$,..., $Label_l(\text{rep}(R_q))$, which correspond to each of the $l$ trees $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_l$.

Assume now that we are interested only for *exact* matches of $R_q$, i.e., exact duplicates. Then, the query would be executed by performing a lookup of each label in the corresponding tree, selecting the resources indexed in the identified nodes, and examining whether each of these resource is an exact duplicate of $R_q$. Notice that due to the hashing and mapping functions employed during the indexing process, several resources may be indexed under the same node, hence the last step in the aforementioned process is required to filter out false positives.

Since in our case we are interested in finding the near duplicates of $R_q$, we need to relax the selection criterion in order to retrieve resources that are not exact matches but highly similar to $R_q$. Recall that due to the property of Locality Sensitive Hashing, similar resources are indexed at nearby nodes in the tree with high probability. Hence, the selection criterion can be relaxed by performing a lookup not for the entire label but only for a prefix of it, of length $k'$. The question that arises is how to determine the appropriate value for $k'$. Setting a high value for $k'$ leads to a stricter selection, and hence some near duplicates

may be missed. On the other hand, a low value for $k'$ retrieves a large result set, from which false positives need to be identified and filtered out, thus reducing the performance of query execution. For example, in the extreme case where $k' = 1$, half of the resources from each tree are retrieved, leading to a very large result set. Consequently, $k'$ should be set to the maximum value that still allows for near duplicate resources to be detected with probability equal or higher than the requested *minProb*. Once $k'$ has been determined, we retrieve from each tree the resources with the same prefix to the respective label of $R_q$, which results in the set of candidate near duplicates for $R_q$, denoted by $\mathcal{ND}_{cand}(R_q)$. Then, for each resource in $\mathcal{ND}_{cand}(R_q)$, we compute its similarity to $R_q$, filtering out those resources having similarity lower than *minSim*. In the following, we provide an analysis on how to determine the right value for $k'$.

The appropriate value $k'$ of the prefix length to be used for the lookup during query execution is determined by the values of *minProb* and *minSim*. We assume that the index comprises $l$ binary trees, and labels of total length $k$ ($k' \leq k$). The computation is based on the following theorem.

**Theorem 1.** *Let $sim(P, Q)$ denote the Jaccard similarity of two resources $P$, $Q$, based on their respective representations $rep(P)$ and $rep(Q)$. The corresponding labels $Label_i(rep(P))$ and $Label_i(rep(Q))$, $i = 1 \ldots l$, of the two resources are equal with probability $Pr[Label_i(rep(P)) = Label_i(rep(Q))] = \left(\frac{1+sim(P,Q)}{2}\right)^k$. Furthermore, the probability that the two resources have at least one common label is $1 - \left(1 - \left(\frac{1+sim(P,Q)}{2}\right)^k\right)^l$.*

*Proof.* As explained in Section 2.3, each bit in the label is computed by (a) hashing all terms of the representation using a hash function from a family of LSH functions $\mathcal{H}$, (b) getting the minimum hash value over all terms, and (c) mapping it to binary. Let $min(h_{i,j}(rep(P)))$ denote the minimum value of the hash function $h_{i,j}$ over all the terms of $rep(P)$, and $\mathcal{M}(min(h_{i,j}(rep(P))))$ the result of the mapping function. The labels $Label_j(rep(P))$ and $Label_j(rep(Q))$ of the two resources $P$ and $Q$ will have the same corresponding bit $i$ if either of the following holds:

(a) $min(h_{i,j}(rep(P))) = min(h_{i,j}(rep(Q)))$ or
(b) $min(h_{i,j}(rep(P))) \neq min(h_{i,j}(rep(Q)))$ and $\mathcal{M}(min(h_{i,j}(rep(P)))) = \mathcal{M}(min(h_{i,j}(rep(Q))))$.

The probability of (a) is directly related to the similarity of the two representations [5], and precisely,

$$Pr[min(h_{i,j}(rep(P))) = min(h_{i,j}(rep(Q)))] = sim(rep(P), rep(Q)) \quad (1)$$

The probability of (b) equals to

$$(1 - Pr[min(h_{i,j}(rep(P))) = min(h_{i,j}(rep(Q)))])/2$$

Since the two cases are mutually exclusive, the probability that either (a) or (b) is true is the sum of the two probabilities, and equals to $\frac{1+sim(P,Q)}{2}$.

For two resources to have the same label $i$, then all bits $1, 2, \ldots, k$ of the two labels must be equal. The probabilities are independent, therefore:

$$Pr[Label_i(rep(P)) = Label_i(rep(Q))] = \left(\frac{1 + sim(P,Q)}{2}\right)^k \tag{2}$$

Then, the probability that the two resources have at least one common label is:

$$\begin{aligned} Pr[\exists i : Label_i(rep(P)) = Label_i(rep(Q))] &= 1 - Pr[\neg\exists i : Label_i(rep(P)) = Label_i(rep(Q))] \\ &= 1 - \left(1 - \left(\frac{1 + sim(P,Q)}{2}\right)^k\right)^l \end{aligned} \tag{3}$$

■

Following directly from Equation 3, we can compute the value of $k'$ as:

$$k' = \left\lfloor -\frac{\log\left(1 - (1 - minProb)^{1/l}\right)}{\log(2) - \log(1 + minSim)} \right\rfloor \tag{4}$$

The number of trees $l$ comprising the index and the length $k$ of each label are set during the initialization of *RDFsim*. Higher values of $l$ allow *RDFsim* to also use longer prefixes of length $k'$ for querying, which results to fewer false positives, and consequently to lower cost for retrieving the candidate near duplicate resources and comparing them to the query. However, as $l$ increases, there is an extra cost imposed for maintaining the additional trees. For tuning these parameters $l$ and $k$, one needs to have some knowledge regarding the queries and the distribution of the resources to be indexed. If this information is not available, one can choose values that are large enough to support a wide range of queries, while still having a good performance. For our experiments, we experimented with different combinations of $l$ and $k$, and we observed that an index with $l = 20$ and $k = 50$ enabled *RDFsim* to answer queries efficiently, for probabilistic guarantees as high as 98% and minimum similarity as low as 0.8. By further increasing $l$ and $k$ one can enable stricter probabilistic guarantees and lower similarity thresholds, albeit with a higher cost for maintaining the index.

## 4 Prototype and Evaluation

In this section, we describe a prototype implementation that uses *RDFsim* to identify near duplicate news articles. We then report the results of our experimental evaluation using the news articles collected by our prototype application.

### 4.1 Prototype Implementation

To test our approach on a real-world scenario, we consider a news aggregation service, which aims at providing a unified view over the articles published on

the Web by various news agencies, identifying and grouping together all near duplicate articles. In particular, we have implemented a prototype in Java 1.6 that uses *RDFsim* to index the RDF representations extracted from incoming news articles and to detect near duplicates, as described in Sections 2 and 3. The application is accessible online, at the following URL: `http://out.l3s.uni-hannover.de:8898/rdfsim/`

The application operates on a large collection of RDF data extracted from real-world news articles. In particular, we crawl news articles from the Google News Web site, which links to articles from various news agencies, such as BBC, Reuters, and CNN. For each newly added news article, we use the *OpenCalais* Web service [18] to extract the RDF statements describing the information available in it[1]. OpenCalais analyzes the text of the news articles and identifies entities described in this text, such as people, locations, organizations, and events, providing an RDF representation of the information in the article.

For the implementation of the binary trees required for indexing the RDF representations of the articles, there are two alternatives that can be used: (a) a main memory binary tree implementation, or (b) an implementation on secondary storage, e.g. a relational database. An efficient main memory implementation of binary trees has been presented in [2] for solving the approximate $k$-nearest neighbor problem. The binary trees are represented as PATRICIA tries [17], which reduces the amount of required memory by replacing long paths in the tree with a single node representing these paths. This compression technique makes the number of tree nodes linear to the number of resources stored in it. In our case, since there are $l$ trees, the total memory requirements will be $O(n \times l)$, where $n$ is the number of indexed resources. However, although accessing the main memory is much faster compared to secondary storage, this approach is limited by the capacity of main memory, and hence it is not suitable for a large number of RDF resources.

Hence, in our implementation, we have used a relational database, in particular MySQL 5, to efficiently store and retrieve all the resources with a given label. The resources are stored in a relational table $\mathcal{I}$ as tuples of the form $(resource\_id, tree\_id, hash\_value)$. *RDFsim* needs to find all labels that share the same prefix of length $k'$ with the query, where $k' \leq k$. This can be efficiently executed in a relational database using SQL operators, e.g., the $LIKE$ operator in MySQL. Hence, all the resources with prefix $v$ from the tree $t$ can be retrieved using the following expression:

$$\pi_{resource\_id}(\sigma_{tree\_id=t \text{ and } hash\_value \ LIKE \ 'v\%'}(\mathcal{I}))$$

The size of the database is $O(n \times l)$, where $n$ is the number of resources, and the complexity of querying is $O(\log(n))$ per tree, i.e.,$O(l \times \log(n))$ in total.

Upon receiving a keyword query, the application identifies the news articles containing these keywords. Then, for each of the found news articles, it retrieves its near duplicates. Based on the near duplicates, it groups the news articles and

[1] The RDF schema for the Web service output is available at: `http://www.opencalais.com/documentation/calais-web-service-api`

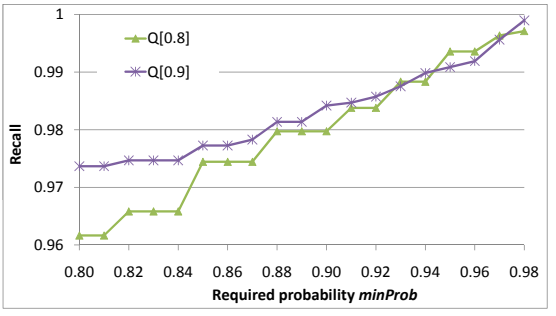



**Fig. 5.** Probabilistic guarantees vs. recall

it returns these groups as the answer to the query. In addition, for each group, we also generate a data cloud that summarizes the entities found in these news articles, taking into consideration the frequency of appearance of these entities in the articles.

### 4.2 Experimental Evaluation

The purpose of the experiments was to evaluate *RDFsim* with respect to quality and efficiency, for executing queries for near duplicate resources. Efficiency was measured as the average time required to execute each query, and quality was measured with recall, i.e., the number of near duplicates detected, divided by the number of total near duplicates in the repository. Note that precision is always 1, since RDFsim includes a filtering step that filters out false positives, as described in Section 3. All the experiments were executed on a server using 1 Gb RAM and one Intel Xeon 2.8GHz processor.

As testbed, we have used the prototype described in Section 4.1. The data set consisted of 94.829 news articles, with a total of 2.711.217 entities, described as RDF statements, and it was stored in a MySQL 5 database, residing at the same machine. The data set is available for download at the following URL: `http://out.l3s.uni-hannover.de:8898/rdfsim/data.html`

We indexed all the news articles using 20 binary trees ($l = 20$), and labels of length 50 ($k = 50$). The ground truth for the experiments was constructed by applying an exhaustive search to detect all pairs of articles that have pairwise similarity above a threshold $minSim$. With $\mathcal{Q}[minSim]$, we denote the set of resources that have at least one near duplicate for the threshold $minSim$. For each article in $\mathcal{Q}[minSim]$, we detected the near duplicate articles. All queries were repeated for different required probabilistic guarantees, expressed as the minimum probability $minProb$ that each near duplicate article with the query will be returned, with $minProb \in [0.8, 0.98]$.

Figure 5 plots the average recall for the queries, for $minSim = 0.8$ and $minSim = 0.9$. As expected, recall increases with the required probability $minProb$. This is due to the fact that when $minProb$ increases, *RDFsim* chooses

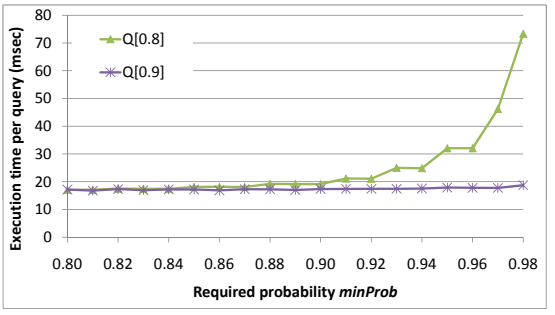



**Fig. 6.** Probabilistic guarantees vs. average query execution time

a smaller length $k'$ for the prefixes of the query labels (see Section 3), and thereby the query retrieves a larger number of candidates. However, it is not necessary to set $minProb$ to very high values in order to get high recall; for our dataset, a value of $minProb = 0.9$ already results in recall over 0.98, which satisfies the practical requirements for most applications.

We also note that the recall is always higher than the value of $minProb$, which verifies that the probabilistic guarantees of the algorithm, described in Section 3, are always satisfied. In fact, the difference between the actual recall and the expected recall (the recall guaranteed by $minProb$) is notable, especially for low $minProb$ values. This happens because $minProb$ controls the probability that each near duplicate will be retrieved, under the assumption that all near duplicates have similarity $minSim$ with the query. However, in practice most of the near duplicates have similarity higher than $minSim$. Therefore, the individual probability that these near duplicates are retrieved ends up to be higher than $minProb$, and the overall quality of the results is better than the one expected according to the value of $minProb$.

With respect to efficiency, Figure 6 shows the average execution time per query, for varying $minProb$ values. The measured time includes the total time required to answer the query, i.e., generating the labels for the query, detecting and retrieving the candidate near duplicates, and comparing all retrieved near duplicates with the query to filter out the false positives. We see that for all configurations, the average execution time is small, always below 100 msec per query. Note that, if exhaustive comparisons are used instead for detecting the near duplicate resources, the time required is around 1 minute per query.

We also see that the average execution time for the queries in $\mathcal{Q}[0.9]$ is always less than the corresponding time for the queries in $\mathcal{Q}[0.8]$. This is due to the effect of the similarity threshold $minSim$ on $k'$: for a higher $minSim$ value, $RDFsim$ can choose a higher value for $k'$, thereby avoiding many false positives and reducing the execution cost significantly. For example, for $minProb = 0.8$, $RDFsim$ sets $k'$ to 49 for $minSim = 0.9$, whereas the corresponding $k'$ value for $minSim = 0.8$ is only 24. Table 1 shows the different combinations of the values of these parameters.

**Table 1.** Values of $k'$ for different combinations of similarity and probability

| minSim/minProb | 0.80 | 0.82 | 0.84 | 0.86 | 0.88 | 0.90 | 0.92 | 0.94 | 0.96 | 0.98 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.8 | 24 | 23 | 23 | 22 | 21 | 21 | 20 | 19 | 18 | 16 |
| 0.9 | 49 | 48 | 47 | 46 | 44 | 43 | 41 | 39 | 37 | 33 |

As expected, the execution time increases as the requested probability $minProb$ increases. This is also due to the lower $k'$ value chosen by $RDFsim$ for answering queries with higher $minProb$ values. This effect is more noticeable for $minSim = 0.8$, since the lower $minSim$ value causes an additional reduction to $k'$, and increases the false positives significantly. For $minSim = 0.9$, the effect of increasing the probabilistic guarantees $minProb$ is not so noticeable because the value of $k'$ remains high, i.e., $k' \geq 33$, and therefore $RDFsim$ does not retrieve many false positives. However, even for queries with very high requirements, e.g., $minProb = 0.98$ and $minSim = 0.8$, the execution time is less than 80 msec per query. Summarizing, the experimental results verify the probabilistic guarantees offered by $RDFsim$ and confirm the effectiveness of the algorithm for detecting near duplicate resources in large RDF repositories in real-time and for configurable requirements.

## 5 Related Work

The problem of data matching and deduplication is a well studied problem appearing with several variants and in several applications [12]. Traditionally, approaches that deal with textual data employ a bag-of-words model and rely on string similarity measures to compare resources [7]. Our work follows a different approach, which instead aims at leveraging the semantic information that can be extracted from the the available resources, so that identifying near duplicate resources can then be performed at the semantic level.

Hence, the approaches that are mostly relevant to our work are the ones that operate not on unstructured text but on complex objects that also contain relationships (e.g., RDF statements, graphs). Such approaches are often employed in Personal Information Management Systems. For example, the Reference Reconciliation [9] algorithm processes the data and identifies near duplicates before propagating and exchanging information in a complex information space. A modified version of this algorithm [1] has also been used for detecting conflict of interests in paper reviewing processes. Probabilistic Entity Linkage [11] constructs a bayesian network from the possible duplicates, and it then uses probabilistic inference for computing their similarity. Other approaches introduced clustering using relationships [3,4], and graph analysis based on the included relationship [13,14]. In contrast to these approaches, our work focuses on the efficient processing of the data for identifying near duplicates, by avoiding the pairwise comparisons between resources.

Locality Sensitive Hashing has also been used for building indexes for similarity search, based on different variants, such as p-stable distributions [8], random

projection [6], and minwise independent permutations [5]. In this work, we follow the latter, which is appropriate for the employed similarity measure, i.e., the Jaccard coefficient, as shown in [10]. Complete indices that incorporate LSH for nearest neighbor and near duplicate queries have been presented in LSH Index [10] and LSH Forest [2]. The LSH Index maintains an in-memory similarity index, which enables queries for $k$-nearest neighbors and near duplicates. Although very efficient, the LSH Index does not allow the user to choose a probability and similarity per query; instead, these are pre-determined from the index configuration. On the other hand, LSH Forest [2] uses index labels of varying length, similar to *RDFsim*. Compared to LSH Forest, *RDFsim* allows the indexing of RDF data, and derives different probabilistic guarantees, which apply to near duplicate detection rather than $k$-nearest neighbor search, which is the main focus of LSH Forest. In addition, *RDFsim* is also built on a relational database, making it easier to be implemented and integrated in existing systems.

## 6 Conclusions and Future Work

We have presented a novel approach that efficiently detects near duplicate resources on the Semantic Web. Our approach utilizes the RDF representations of resources to detect near duplicates taking into consideration the semantics and structure in the resource descriptions. It also employs an index using LSH in order to efficiently identify near duplicates, avoiding the need for a large number of pairwise similarity computations. We provided a probabilistic analysis that allows to configure the algorithm according to specific quality requirements of users or applications. In addition, we have implemented a system that illustrates the benefits of the approach on a real-world scenario regarding the online aggregation of news articles, and we have presented the results of our experimental evaluation.

Directions for future work include exploiting this efficient, online near duplicate detection method, to improve tasks such as diversification or summarization of search results.

## Acknowledgments

This work is partially supported by the FP7 EU Projects OKKAM (contract no. 215032) and Living Knowledge (contract no. 231126).

## References

1. Aleman-Meza, B., Nagarajan, M., Ramakrishnan, C., Ding, L., Kolari, P., Sheth, A.P., Arpinar, I.B., Joshi, A., Finin, T.: Semantic analytics on social networks: experiences in addressing the problem of conflict of interest detection. In: WWW, pp. 407–416 (2006)
2. Bawa, M., Condie, T., Ganesan, P.: LSH forest: self-tuning indexes for similarity search. In: WWW, pp. 651–660 (2005)

3. Bhattacharya, I., Getoor, L.: Deduplication and group detection using links. In: Workshop on Link Analysis and Group Detection, ACM SIGKDD (2004)
4. Bhattacharya, I., Getoor, L.: Iterative record linkage for cleaning and integration. In: DMKD, pp. 11–18 (2004)
5. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations (extended abstract). In: STOC (1998)
6. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: STOC, pp. 327–336 (2002)
7. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string distance metrics for name-matching tasks. In: Workshop on Inf. Integration on the Web (2003)
8. Datar, M., Indyk, P.: Locality-sensitive hashing scheme based on p-stable distributions. In: SCG 2004: Proceedings of the twentieth annual symposium on Computational geometry, pp. 253–262. ACM Press, New York (2004)
9. Dong, X., Halevy, A.Y., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD, pp. 85–96 (2005)
10. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB, pp. 432–442 (1999)
11. Ioannou, E., Niederé, C., Nejdl, W.: Probabilistic entity linkage for heterogeneous information spaces. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 556–570. Springer, Heidelberg (2008)
12. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of tampa. American Statistical Association (1989)
13. Kalashnikov, D.V., Mehrotra, S.: Domain-independent data cleaning via analysis of entity-relationship graph. ACM Trans. Database Syst., 716–767 (2006)
14. Kalashnikov, D.V., Mehrotra, S., Chen, Z.: Exploiting relationships for domain-independent data cleaning. In: SDM (2005)
15. Manku, G.S., Jain, A., Sarma, A.D.: Detecting near-duplicates for web crawling. In: WWW, pp. 141–150 (2007)
16. Minack, E., Paiu, R., Costache, S., Demartini, G., Gaugaz, J., Ioannou, E., Chirita, P.-A., Nejdl, W.: Leveraging personal metadata for desktop search - the Beagle++ system. In: Journal of Web Semantics (2010)
17. Morrison, D.R.: PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. J. ACM (1968)
18. Open Calais, `http://www.opencalais.com/`

# Guarding a Walled Garden — Semantic Privacy Preferences for the Social Web

Philipp Kärger and Wolf Siberski

L3S Research Center & University of Hannover, Germany

**Abstract.** With increasing usage of Social Networks, giving users the possibility to establish access restrictions on their data and resources becomes more and more important. However, privacy preferences in nowaday's Social Network applications are rather limited and do not allow to define policies with fine-grained concept definitions. Moreover, due to the walled garden structure of the Social Web, current privacy settings for one platform cannot refer to information about people on other platforms. In addition, although most of the Social Network's privacy settings share the same nature, users are forced to define and maintain their privacy settings separately for each platform. In this paper, we present a semantic model for privacy preferences on Social Web applications that overcomes those problems. Our model extends the current privacy model for Social Platforms by semantic concept definitions. By means of these concepts, users are enabled to exactly define what portion of their profile or which resources they want to protect and which user category is allowed to see those parts. Such category definitions are not limited to one single platform but can refer to information from other platforms as well. We show how this model can be implemented as extension of the OpenSocial standard, to enable advanced privacy settings which can be exchanged among OpenSocial platforms.

## 1 Introduction

The Social Web gained momentum in the last years. This is shown not only by the high number of users currently registered and communicating on Social Network applications but also by the growing number of different platforms and applications available. Since more and more data is shared and more and more personal information is exposed on the Social Web, the need for advanced and fine-grained privacy preferences emerges [1]. But when looking at the privacy features of current Social Web applications, one is presented with a restricted and simple mapping between predefined categories of things to protect and categories of people who are allowed to access those information (see Figure 1 for examples). A standard policy, for example, is that specific data or requests, e.g., a user's profile information, is only allowed for people the user has a contact or friendship relation with. But as soon as more complex restrictions are needed to be put on the requester, a mapping between simple categories does not suffice. Examples are "being member of a group", "sharing the same interest", "working for the same company",

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 151–165, 2010.

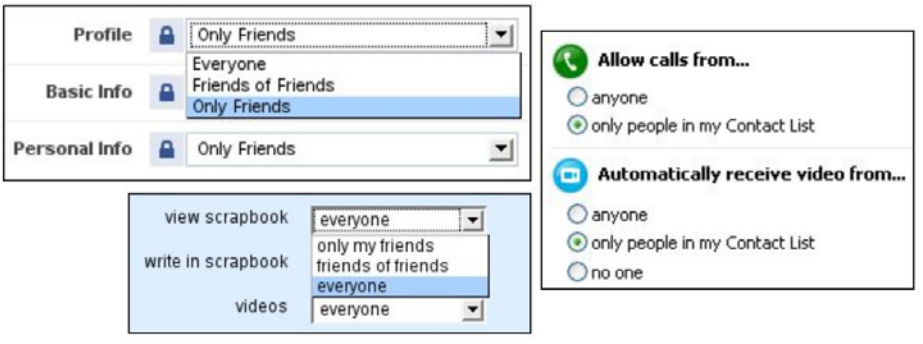



**Fig. 1.** Example mappings from object categories to subject categories in the privacy settings of Facebook (upper left), Orkut, and Skype (right)

or "being over 18". On social platforms, just like in real life, people base privacy decisions on social information, such as "is the requester my friend", "did I ever talk to her", "is she working in the same project", etc. The problem is not only that this social information is not available for privacy preferences, unfortunately, the existing Social Web platforms hide all this information behind fences: the Social Web is partitioned into various platforms and thus social data is not linked but encapsulated in proprietary data silos. This issue is often referred to as the "Walled Garden" of the Social Web [2]. The main problem arising from this separation is that information about people and their relationships is trapped inside the platforms and not available outside the platform it was stated in. Thus, social data and contexts are not available for privacy preferences [3].

But not only the social data is isolated, the privacy preferences themselves are trapped as well. Assume a user who may have stated on one platform that only friends of friends can see her profile and only friends can send her messages. This person cannot reuse her privacy policy on another platform, although the second platform may as well have friendship relations, a profile and the possibility to send messages. Consequently, the first thing a user has to do when creating a new profile on a new platform is to recreate her privacy settings since there is no way to exchange those settings and apply one platform's privacy settings on another social application.

In this paper, we analyse the format and features of privacy settings in current Social Network applications. Based on this analysis, we provide a formal model for privacy preferences on the Social Web that bases on rules and that straightforwardly extends the pairwise mapping in current privacy preferences. Our approach is solving the three aforementioned shortcomings of Social Web privacy settings: (1) It lets users freely define complex categories of persons like "people who are either friends or colleagues". It also allows for complex categories of actions or objects that are to be protected such as "sending me group invitations" or "seeing my pictures that are tagged with `eswc` and taken in June 2010". (2) It allows users to define categories of people by referring to arbitrary social data either stored on other platforms or available on the Semantic Web. (3) Privacy preferences defined

in this model can be exchanged among platforms since they refer to well-defined semantic categories gathered from all over the Social Web. We show how privacy preferences are enforced based on those category definitions and describe an extension of the OpenSocial standard that implements this model and allows any platform that supports OpenSocial to make use of our model. This implementation also features an RDF serialization of privacy preferences and allows them to be exchanged between platforms.

The remainder of this paper is structured as follows. In the next section we motivate our approach with a scenario and extracted requirements. In Section 3 we review today's privacy preferences on the Social Web. Based on these observations we describe our model in Section 4 and its implementation based on OpenSocial in Section 5. Related work is described in Section 6 and Section 7 concludes the paper.

## 2 Motivation and Requirements

To illustrate the goal of our approach we start with a motivating scenario. It serves to extract the requirements and it will be used throughout the paper to explain our approach.

Alice is a member of several Social Network applications and platforms, she has an account on Facebook as well as on Orkut. To keep in contact with her friends, she is using Skype for chat and IP telephony. On LinkedIn she is managing her business contacts. As a privacy setting, Alice wants only her friends to access her profile which contains all personal information such as name, age, organizations, address, interest, etc.. With her age, she is even more strict: only her family members are allowed to see it.

Beyond personal data, Alice generally wants to share any uploaded picture only with her friends. As "friend" she considers her contacts in Skype and her friends in Orkut and Facebook. Her contacts on LinkedIn are not included since they are rather business contacts. On Facebook, Alice recently uploaded some pictures she took at ESWC and tagged them with `eswc`. With these pictures she is not as restrictive: she wants to share them with anybody she calls a Semantic Web fellow, that is, anybody who is in the group *ESWC*, who has stated as interest *Semantic Web*, or who is listed as Friend in her FOAF profile.

On all the four Social Networking applications, one can send messages. Alice is quite restrictive here because she is facing a message overload since her network grew. That is why she allows only her Skype contacts to send messages to her on any of the platforms. An exception is messages on LinkedIn, since Alice plans to change her job, she wants all LinkedIn contacts (i.e., business contacts) to be allowed to send messages.

**Extracted Policies.** To sum up the described scenario we can extract the following policies that make up Alice's privacy preferences:

P1 Disclose my profile information only to my contacts in Facebook, Skype, Orkut and LinkedIn.
P2 Disclose my age on any platform only to my family.

P3 Disclose pictures only to friends.
P4 Disclose ESWC-pictures to Semantic Web fellows only.
P5 Accept messages sent from Skype contacts only.
P6 Accept LinkedIn-messages sent from business contacts.

On top of that, Alice used a personal vocabulary to define her privacy preferences: what she considers an ESWC picture or a business contact may differ from other users. Later in the paper we will refer to those concepts as category definitions. So we can extract the following definitions of categories in her privacy preferences:

D1 *Profile information* is everything that is name, organizations, address, interest, or age
D2 *Family* is everybody who is in my *family*-group on Facebook.
D3 An *ESWC picture* is everything that is both, a picture and tagged with `eswc`.
D4 A *Semantic Web fellow* is everybody who is in the Facebook group *ESWC*, who has stated as interest *Semantic Web*, or who is listed in my FOAF profile.
D5 A *business contact* is everybody who is a contact on LinkedIn.

In Figure 3, later in the paper, we provide a graphical representation of Alice's preferences and category definitions.

**Requirements.** The given scenario requires several extensions to current approaches to privacy on the Social Web. (1) Users may be allowed to freely define new categories of people (like "family members") or of objects (like "ESWC pictures"); thus reflecting their particular social environment. (2) Privacy preferences are expressed crossing the borders of social platforms. Properties of a requester may be gathered from different platforms (like "contacts on LinkedIn") or data sources (like "friends in my FOAF profile") in order to allow a certain action. (3) Policies referring to generic concepts should hold on all platforms, regardless on which platform they were defined. For example, the rule that only contacts are allowed to see Alice's profile information should apply on all platforms where Alice is participant.

## 3 Today's Social Web Privacy Preferences

Most of the Social Network applications[1] share similar concepts: there is always a profile containing name, an image, contact information, etc. In most of the cases, there is a way to communicate (text messages, wall posts, etc.) and there are connections among people and new connections can be set up. Consequently, the privacy preferences also share the same nature among platforms: they are typically a set of mappings between objects or actions other users can access – called *object categories* in the following – and groups of people that are allowed

[1] With this term we refer to any kind of application that is based on a Social Network, ranging from Web platforms like Facebook to social communication tools like Skype.

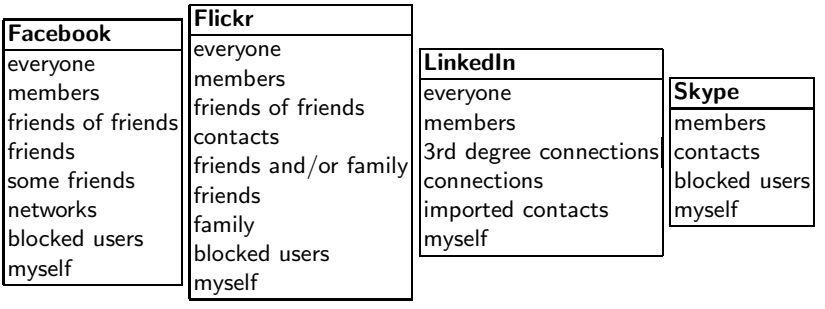



**Fig. 2.** Subject categories available in the privacy settings of current Social Network applications. They only roughly capture real-life relationships among people.

to access these objects – called *subject categories* (see Figure 1 for example mappings). Examples for *object* categories are "send a chat message", "view a picture", "see address" etc. On the other hand, *subject* categories include "contacts", "friends of friends", etc. Figure 3 lists the subject categories that are available for privacy settings in Facebook, Skype, LinkedIn and Flickr.

**Limited options of categories.** Objects and actions whose access can be restricted in today's privacy preferences fall into five high level categories:

1. accessing certain parts of a user's profile such as contact data or date of birth
2. accessing specific content such as pictures or videos uploaded to the platform or the user's wall or message board
3. communication actions such as sending chat messages, "poking", inviting to games or sending gifts
4. actions related to connections among persons, e.g., creating a friendship link
5. other actions such as tagging people in pictures

All these categories of objects can be restricted to be allowed only by a specific group of subjects. However, combinations of object categories as they are used in the scenario's definitions D1 and D3, are not possible. Subject categories are even more restricted: looking at Figure 3, it is easy to see that subject categories as they are currently offered in privacy preferences do not reflect the complex considerations one undertakes while making privacy decisions in real life.

**No cross-platform definition of categories possible.** Privacy preferences can only be defined based on information that is available on the very same platform [4]. Although other platforms may share the same concept of friendship, one cannot define and use categories like "people that are my friends on any platform". Even publicly available social information (that may be available on the Semantic Web) cannot be considered.

**No portability among platforms.** The object categories and the subject categories are similar in most of the platforms (see Figure 3). Still, there is no method to apply privacy setting defined on one platform to another one. For each platform, an identical privacy setting has to be defined manually.

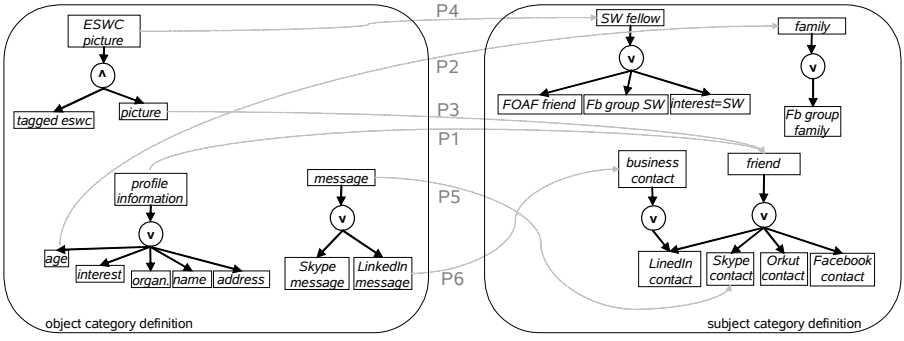



**Fig. 3.** A graphical representation of the example privacy preferences from Section 2. Category definitions are represented as AND/OR graphs. The policies (P1-P6) mapping object categories (left) to subject categories (right) are represented as dotted lines.

**Authorization pairs.** Interestingly, the classical authorization or access control triple is not used in Social Web applications. This classical scheme is known from access control in databases or file systems and requires a privacy statement to consist of three parts: an object that is going to be protected, an action whose performance on the object is to be restricted (for example, *access*, *write* and *execute*) and a subject (typically a role) that is allowed to perform the action on the object (cf. Clark-Wilson model in [5, Chapter 2.8]). On Social Web platforms, however, the concept of object and action is merged, resulting in a subject-object pair. An example is given in Figure 1 where writing and viewing a scrapbook in Orkut's settings form two separate object categories actually composed of an action and the actual object. The reason for refraining from the classical triple scheme in the context of Social Web platforms is obvious: first, it is too complicated to be maintained by the average user since the complexity is increased by one dimension. Second, either the variety of different actions that can be performed on the same object is low (profile data can only be viewed, messages can only be sent, etc.) or an action and an object can be merged where necessary (e.g., the scrapbook example in Fig. 1). These two arguments let us keep the authorization pairs pattern for our approach as well.

## 4 A Unified Model for Privacy Preferences

The requirements and shortcomings described in the previous sections are used in the following to develop a unified and interoperable preference model that fits the needs for the Social Web while still ensuring privacy policies to be enforced.

### 4.1 Defining New Subject and Object Categories

As stated in the requirements, to express access restrictions on more complex categories of objects, users shall be allowed to define new categories based on the

ones offered by a social application. New categories shall be defined as disjunctions of other categories in order to group together concepts that deserve similar privacy preferences. Conjunctions are required wherever existing categories are not fine-grained enough. Categories defined that way shall also be reused to define other personalized categories.

Recalling the example from the motivation (Definition D3), *eswc_picture* is the conjunction of everything that is a *picture* and that is *tagged_with_eswc*. To support both, disjunction and conjunction as well as the reuse of defined categories for more definitions, we represent definitions as Datalog rules as follows.

**Definition 1 (Category, category definition, context).** *An object category $c_o$ rsp. subject category $c_s$ is a unary predicate whose argument is an object rsp. a subject. An* object/subject category definition *$P$ on a set of categories $C$ is a set of rules of the form $H \leftarrow B_1, \ldots, B_n.$ (conjunctive rule) or $H \leftarrow B_1; \ldots; B_n.$ (disjunctive rule)*[2] *where $H, B_i \in C$ are object/subject categories and there is no pair of rules "$H_1(X) \leftarrow body_1.$" and "$H_2(Y) \leftarrow body_2.$" (with $body_i$ being either a conjunction or disjunction of predicates) with $H_1 = H_2$. Properties of objects/subjects are represented as sets of ground facts of object/subject categories stating the category an object/subject belongs to. We call this set of facts the* context *of an object/subject, denoted by $Con$.*

The two boxes in Figure 3 shows the graphical representations of definition rules in form of an AND/OR graph. An example rule defining an object category is

$$eswc\_picture(X) \leftarrow picture(X), tagged\_with\_eswc(X).$$

New categories of subjects can be defined in a similar way. For example, *Semantic Web fellow* from Definition D4:

$$sw\_fellow(X) \leftarrow inSWGroup(X); swAsInterest(X).$$

The context of a file $f$ that is a picture (i.e., belongs to the category *picture*) and is tagged with `eswc` has the context $\{picture(f)., tagged\_with\_eswc(f).\}$. Consequently, given an object $o$ with its context $Con(o)$ and a category definition $P$, $o$ belongs to a category $c$ if $P \cup Con(o) \models c(o)$, that is, $c(o)$ is in the semantic consequence of the logic program $P \cup Con(o)$[3]. Contexts of subjects are typically available as RDF, e.g., subject information in FOAF, or gathered from proprietary sources. The context of objects is determined on the platform where the request is happening.

The restriction that a category definition should not contain two rules with the same category in the head is justified by the fact that a category used in privacy preferences is only defined once, either as conjunction or disjunction of other categories. We chose this simplification here because Social Web users are not expected to understand a nesting of conjunction and disjunction. However, by introducing auxiliary predicates, nesting could be simulated easily.

[2] We use ; to denote disjunction in a rule's body. A rule $H \leftarrow B_1; \ldots; B_n.$ is a shortcut for the list of $n$ rules of the form $H \leftarrow B_i.$ $(1 \leq i \leq n)$.

[3] In the remainder of the paper we may omit $Con(o)$ and use the shortcut $P \models c(o)$ where unambiguously applicable.

$O := \{everybody(\_), contact(\_), blocked\_user(\_), myself(\_)\}$
$S := \{seeContactNumber(\_), sendChat(\_), call(\_), sendVideo(\_)\}$
Default mapping $\mathcal{M}$ when installing Skype:
$\mathcal{M}(sendChat(\_)) := everybody(\_)),$
$\mathcal{M}(call(\_)) := everybody(\_)),$
$\mathcal{M}(sendVideo(\_)) := contact(\_),$
$\mathcal{M}(seeNumerOfContacts(\_)) := everybody(\_)$

**Fig. 4.** The object and subject categories ($O$ and $S$) available in Skype and Skype's default privacy preferences $\langle O, \emptyset, S, \emptyset, \mathcal{M}\rangle$ when being installed. Skype has empty category definitions as every current Social Network application.

Restricting categories to be unary predicates is a conceptual simplification. For example, $tagged\_with\_eswc(f)$ could well be understood as syntactic sugar for $tagged\_with(f, "eswc")$. Again, we assume users to use unary category definitions in their privacy preferences rather than categories with two or more variables. In fact, in our implementation (see Section 5), we realize a predicate determining tags of an object by SPARQL queries allowing for more than one variable where the tag explicitely stated in the category name is shifted as an argument of the predicate that accepts general tags.

As a consequence of those simplifications plus the fact that no negation is included, the rules used in the category definitions are very simple and the evaluation of a goal (e.g., if an object or subject belongs to a specific category) can be evaluated with PTIME complexity [6]. Such simple programs can be represented as AND/OR graphs (see Figure 3) where the nodes that have no outgoing edges are the basic categories offered by the platform and are contained in the context of an object or can be retrieved for a subject (e.g., if someone is a friend in a FOAF profile). The restriction that no category is defined twice is reflected in the graph by the fact that no concept node has two outgoing edges and the outgoing edge always leads to either an AND or an OR node.

Further, it is worth noting that this category definition with rules coincides with the simple Description Logic featuring only conjunction and disjunction of concepts [7]. We stick to the rule representation here, since syntactic restrictions are expressible in a more straightforward way. Further, rules better reflect our implementation based on rule-based Semantic Web policies [8].

### 4.2 Defining Privacy Preference Mappings

Until now, users are enabled to define new categories for their privacy settings. Following the binary mapping scheme (see Figure 1) we now define how to use these categories to build up a policy. We define privacy preferences as a mapping between object and subject categories—with the difference that those categories are user-defined following the category specifications from Definition 1.

**Definition 2 (Privacy preference).** *A privacy preference $\mathcal{P}$ is a quintuple $\langle O, P_O, S, P_S, \mathcal{M}\rangle$, where $O$ is a set of object categories, $P_O$ is an object category*

*definition on $O$, $S$ is a set of subject categories, $P_S$ is a subject category definition on $S$, and $\mathcal{M} : \mathcal{O} \mapsto \mathcal{S}$ is a mapping from object categories to subject categories.*

*Example 1.* Privacy preferences of current social platforms are always of the form $\langle O, \emptyset, S, \emptyset, \mathcal{M} \rangle$ because – as pointed out in Section 3 – category definitions are not allowed. As a more detailed example, Figure 4 shows the default privacy preferences implemented in Skype clients. Further, a graphical representation of the privacy preferences from the scenario in Section 2 is given in Figure 3.

Such a privacy preference is applied in the realm of a Social Web application where the sets of objects and subjects are defined as well as an ownership relation determining who is allowed to enforce policies on which object. Further, each application offers a set of subject as well as object categories which can be used to define personalized categories (see Def. 1).

**Definition 3 (Social Web application).** *A Social Web application is a hextuple $\langle Obj, Subj, O, S, Cat, Owns \rangle$ whereas $Obj$ is the set of objects in the application, $Subj$ the set of subjects, $O$ and $S$ a set of object and subject categories. $Cat$ is a function assigning a context to each subject and object. $Owns : Obj \mapsto Subj$ is a function defining the ownership of objects, i.e., the subject that is supposed to define privacy preferences for a given object.*

### 4.3 Enforcing Privacy Preferences

Based on our model for privacy settings, we now define what a request is and how to determine if a request meets a privacy preference or not. Generally, a request to access a specific object is allowed if the requester matches the privacy restriction attached to the category the object belongs to. Since objects may belong to several object categories it is important to determine the correct, most descriptive object categories for a given object. For example, a file $f$ may belong to the category *picture*, to the category *tagged_with_eswc* and thus—according to Definition D3 from the scenario—as well to the category *ESWC_picture*. In this case, the category that describes best what $f$ belongs to is *ESWC_picture*. We refer to those categories as *descriptive categories*. It is intuitive to apply the policy that is defined for *ESWC_picture* instead of the one defined for *picture*. Since Alice defined pictures being visible only to friends but ESWC pictures being visible to Semantic Web fellows (which is a far more general than friends), this intuition is actually what she intended: a requester accessing a picture that is tagged with `eswc` has to meet different conditions than a requester accessing a picture not having this tag. We formally define descriptive categories as follows:

**Definition 4 (Descriptive category).** *Let $P$ be an Object Category Definition and $o$ an object, then an object category $c$ is a descriptive category of $o$ if*

1. *$P \models c(o)$ and*
2. *there is no $c' \neq c$ with $P \models c'(o)$ and $\forall X : P \models c(X) \rightarrow P \models c'(X)$*

*We define $Des(o)$ to be the set of all descriptive categories of an object $o$.*

A request is a pair $\langle o, s_v \rangle$ where a subject $s_v$ (the viewer or requester in this case) is requesting access to an object $o$.

**Definition 5 (Granting access).** *Given a Social Web application and the privacy preference $\mathcal{P} = \langle O, P_O, S, P_S, \mathcal{M} \rangle$, $s_v$ is allowed to access $o$ iff there is a subject category $c_s$ and an object category $c_o \in Des(o)$ such that $c_s = \mathcal{M}(c_o)$ and $P_S \models c_s(s)$. That is, the subject must belong to at least one subject category that is mapped to one of the object's descriptive categories.*

So far, we have defined which object category is to be considered for deciding which subject category the requester has to belong to. But looking at the scenario in Section 2 and Figure 3, it may happen that some object categories do not have a subject category mapped. For example, the object $o =$"sending Alice a Skype message" will always lead to denial of access because $Des(o)$ has the only element $Skype\_message$ and $\mathcal{M}(Skype\_message)$ is empty. However, Figure 3 reveals the intuition that the subject category mapped to the super concept of "Skype message" shall be applied, in this case, it is the category that is mapped to "message". Thus, informally spoken, the super categories of a descriptive category shall apply, if for no descriptive category of an object a subject category is mapped. For this we need to define the set of super categories $Sup$ according to disjunctive rules in the object category definition.

**Definition 6 (Disjunctive super category).** *Given an Object Category Definition $P$ and an object category $c_o$, the set of disjunctive super categories $Sup$ of $c_o$ is defined as $Sup(c_o) := \{c'_o \mid \exists r \in P : r = c'_o \leftarrow B_1; \ldots; c_o; \ldots; B_n.\}$*

Given this concept we can relax Definition 5 to accept incomplete mappings and inherit the subject categories mapped to super categories of an object's descriptive categories.

**Definition 7 (Granting access (incomplete mappings)).** *Given a Social Web application and the privacy preference $\mathcal{P} = \langle O, P_O, S, P_S, \mathcal{M} \rangle$, $s_v$ is allowed to access $o$ iff there is a subject category $c_s$ and an object category $c_o \in Des(o)$ such that $\exists x_1, \ldots, x_n : x_i \in Sup(x_{(i+1)}) \wedge x_n = c_o \wedge \forall x_i (2 \leq i \leq n) \mathcal{M}(x_i) \neq \emptyset \wedge (c_s = \mathcal{M}(x_n)) \wedge (P_S \models c_s(s))$.*

The chain defined by the $x_1, \ldots, x_n$ is a sequence of object categories defined in $P_O$ where the user defined the category $x_i$ by a disjunction containing $x_{i+1}$. If for $x_1$ a subject category is defined and for the remaining $x_2$ to $x_n$ no mapping has been provided, the mapping for $x_1$ will be applied. Note that for the case where $\mathcal{M}(c_o)$ is not empty this definition coincides with Definition 5 and $i = 1$.

## 5 Implementation

In order to validate the applicability of our privacy preference model, we first built a preference reasoner based on the policy engine Protune [9] that handles general category definitions and considers OpenSocial data as well as general Social Web data for the reasoning process. Second, we extended the OpenSocial-based Social Platform Shindig in a way, that privacy preferences defined

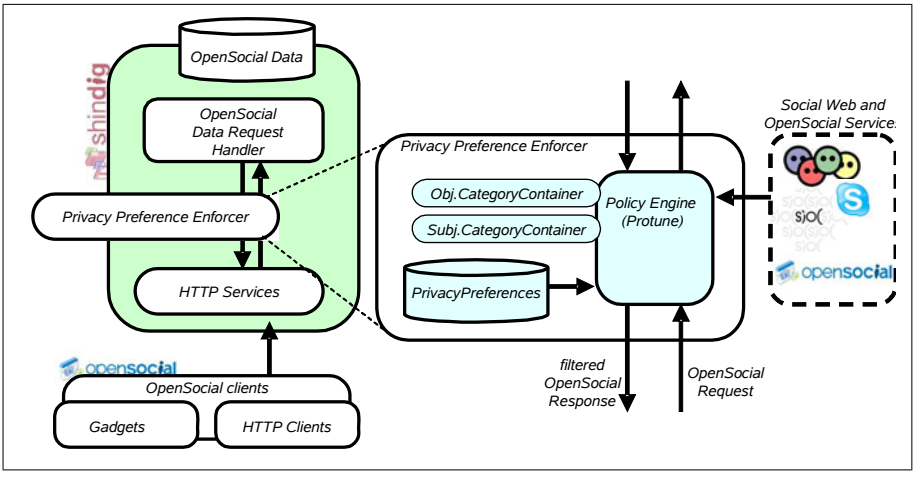



**Fig. 5.** The OpenSocial container Shindig (left) extended by a general privacy preference enforcement (center). The policy engine integrates external social and Semantic Web data into the reasoning process (right).

according to our model are enforced in any Social Web platform based on Shindig (see an architecture overview in Figure 5). We made our implementation available on line at `www.L3S.de/~kaerger/SocialWebPrivacy`. In the following, we shortly describe the components our implementation is based on and then detail the implementation itself.

**Protune.** Protune [9] is a policy framework featuring a logic programming-inspired policy language and a policy engine that supports credential handling, trust negotiation, and automated explanation generation of evaluation outcomes. The Protune engine is able to integrate external data sources into the reasoning process such that ground facts do not have to be present explicitly but can be retrieved on demand from external sources during the reasoning process. In the present work we use this feature to incorporate social data from the (Semantic) Web into the reasoning process [4].

**OpenSocial.** Facing the bulk of Social Web platforms that went on line in the last years each with proprietary technology, OpenSocial [10] is an interface definition describing functions that are common in most SocialWeb platforms. If a platform supports OpenSocial, gadgets or remote procedure calls which were initially implemented for a different OpenSocial-enabled platform can easily be imported. OpenSocial offers four types of requests: asking for information about people (i.e., profile information), about activity notifications (e.g., an image was uploaded, a group was joined), about application data (data that is stored for specific applications inside a social platform, such as applications for sending gifts), and about sending messages.

**Apache Shindig.** Apache Shindig[4] is an OpenSocial container for hosting OpenSocial web applications. It is an open source implementation for OpenSocial clients (e.g., JavaScripts accessing OpenSocial services) and OpenSocial servers (social platforms offering OpenSocial services). In the present paper, we extended the server-side Java implementation of Shindig to support the filtering of requests from OpenSocial clients.

### 5.1 A Category-Based Policy Engine

Given a privacy preference as defined in Definition 2, the evaluation of a request is performed in two steps: first, the object categories for the object are determined and second, the subject categories, that are mapped to the object categories are checked for the requester:

1. For the given object and its context (the set of basic categories the object belongs to) the set of descriptive categories (see Definition 4) is derived from the object category definition in the privacy preference.
2. It is then checked if the requester belongs to the subject categories that are mapped to these descriptive categories. For each of those categories, a query to the policy engine is posed.

The available object categories and the specification how to find out if given an object's identifier the object belongs to the category, is defined in the *Object Category Container* (see Figure 5). This container can easily be extended if, for example, a certain environment requires specific object categories, e.g., images that are larger than 2 MB, etc. The available subject categories are stored in the *Subject Category Container* which can also be extended easily in case new subject categories are required. For example, in order to retrieve project memberships of people, a new platform may be integrated that stores projects and the people working in them. Currently, the following subject categories are supported: a person is listed in my FOAF profile, is following me on Twitter, is my friend on Flickr, is my friend on some OpenSocial platform, shows a specific value in the OpenSocial.Person.FIELD[5] on some OpenSocial platform, is my co-author on DBLP. For details about how to gather the social data for the reasoning process we refer the reader to our previous work in [4]. In the following section we describe how this privacy enforcement is integrated into Shindig.

### 5.2 An OpenSocial Container with General Privacy Preferences

We extended the OpenSocial container Apache Shindig to provide advanced privacy control over data that is exposed by Shindig's OpenSocial interface. If an HTTP request (either JSON RPC or REST, both OpenSocial implementations are available in Shindig) arrives, it is first checked which object is requested and

[4] See `http://incubator.apache.org/shindig/`

[5] See `code.google.com/apis/opensocial/docs/0.7/reference/opensocial.Person.Field.html`

the according object categories are collected. OpenSocial requests typically ask for several objects at once. For example, the request for a person's complete profile contains all the requests for viewing the person's name, hobbies, address, etc. Consequently, such an OpenSocial request is internally transformed into a set of requests for each single object and these requests are passed to the policy engine. As a result, if one of the requests fails, the OpenSocial request is not rejected as a whole but the objects that are not allowed to be accessed by the requester are removed from the response.

### 5.3 Results

In the following we summarize the features of our approach and explain how our implementation solves the requirements identified in Section 2.

*Category definitions.* New categories can be defined based on the basic categories that are implemented in the category containers. Object categories currently implemented refer to OpenSocial concepts only, thus each profile information field, sending messages, activity notifications, and application data can be protected. As described in Section 5.1, subject categories can be defined arbitrarily.

*Crossing borders of social platforms.* Subject categories can be defined based on social data inside Shindig, in other OpenSocial platforms, on other Social Web platforms and on Social Semantic Web data.

*Platform independence.* In its current implementation, any Shindig-based platform can use the presented privacy model, thus, privacy preferences can be exchanged at least among those platforms.[6] Apart from its actual implementation described in this section, our model is platform independent with the following conditions. The basic subject categories are platform independent and can be applied out-of-the-box. Basic object categories that are part of an object's context instead are platform dependent - however, it is a small effort to adopt the current implementation such that object category decisions can be made on another platform as well. Such decisions are, for example, if a request is for accessing a person's age, for sending a message, or for seeing something tagged with `eswc`. However, as soon as user-defined categories are concerned, be it subject or object categories, they are completely platform independent.

On top of that, our implementation features an RDF export of privacy preferences (basically a serialization of the format defined in Def. 2) that allows the exchange of privacy preferences between Shindig-based OpenSocial platforms. A possible scenario is that a user stores her preferences on a central personal location to be accessed by the platforms and applications she is working with.

## 6 Related Work

In this section we relate our approach to other work either using Semantic Web techniques for privacy control or extending Social Web standards towards privacy control crossing platform borders.

[6] More details are available at `www.L3S.de/~kaerger/SocialWebPrivacy`

Lockr [11] is an access control scheme for sharing content. It exploits a local address book storing social relationships to be applied for access control policies on several content sharing platforms on the Web. Lockr also motivates the use of one privacy setting across platforms but focuses more on the authentication of subjects (via so-called attestations) than on definition of complex privacy preferences. Our approach adds the idea of defining which categories of objects to share with whom by relying on arbitrary social data whereas Lockr requires users to manually maintain a social network on their local machine and to define a privacy setting for each resource separately.

An approach similar to Lockr is presented in [12]. There, content is shared by emailing secret links referring to the content to a set of subjects. Again, the focus lies more on the authentication of users. Our approach could apply the presented techniques for credential exchange and attribute assertion to establish a subject's context.

The work presented in [13] describes a formal model for privacy preservation on Social Network systems. The focus of this work is to model access control on Facebook-style platforms including the step-wise establishment of friendship relations, etc. Our work builds on top of that since we consider an extension of privacy preferences and their evaluation.

Privacy and OpenSocial is subject of the research presented in [14]. The proposed solution is meant to help users in justifying their privacy settings with the help of a privacy score: the higher the score, the better, the more secure, the more restrictive the privacy settings. An extension to the OpenSocial interface is suggested that is able to deal with privacy scores. This work shares our approach's motivation and complements it, since its goal is to evaluate privacy settings instead of providing better means to express them.

In [15], a Description Logic based access control model for Web 2.0 is described where access control policies are defined as triples of subjects, objects, and permissions. This approach focuses on the use of lightweight ontologies to structure subjects, objects, and permissions. In contrast to our approach, object and subject categories are not defined by rules, but organized in a tree-like hierarchy (in contrast to AND/OR graphs) thus featuring only disjunction of concepts. Further, [15] does not provide a formal definition for the evaluation of access requests.

## 7 Conclusions

In this paper, we introduce a privacy preference scheme for Social Network applications that is flexible enough to express user-defined categories of objects and subjects. This enables users to reflect their personal social environment in the privacy preferences. Since these categories may leave the realm of one single social platform, privacy preference can be expressed based on arbitrary social data. Furthermore, since our scheme is based on generic categories that are common to most Social Network applications, it can be ported from one platform to another thus avoiding redundant definitions of privacy preferences.

Our implementation shows that this scheme, realized as extension to a standard OpenSocial platform, can be used to provide a simple privacy setting format that works for any OpenSocial compliant platform. Further, since it is based on standards like RDF and OpenSocial, privacy preferences can be exchanged easily among Social Network applications.

## References


1. Rosenblum, D.: What anyone can know: The privacy risks of social networking sites. IEEE Security & Privacy 5(3) (May-June 2007)
2. Breslin, J., Decker, S.: The future of social networks on the internet: The need for semantics. IEEE Internet Computing 11(6), 86–90 (2007)
3. Grandison, T., Maximilien, E.M.: Towards privacy propagation in the social web. In: Workshop on Web 2.0 Security and Privacy at the 2008 IEEE Symposium on Security and Privacy, Oakland, California, USA, May 18-21 (2008)
4. Kärger, P., Kigel, E., Olmedilla, D.: Reactivity and social data: Keys to drive decisions in social network applications. In: Second ISWC Workshop on Social Data on the Web, SDoW 2009 (2009)
5. Ferraiolo, D.F., Kuhn, R., Chandramouli, R.: Role-Based Access Control. Artech House (2003), ISBN: 1580533701
6. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge University Press, Cambridge (2003)
7. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: Description Logic Handbook. Cambridge University Press, Cambridge (2003)
8. Bonatti, P.A., Duma, C., Fuchs, N., Nejdl, W., Olmedilla, D., Peer, J., Shahmehri, N.: Semantic web policies - a discussion of requirements and research issues. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 712–724. Springer, Heidelberg (2006)
9. Bonatti, P.A., Olmedilla, D.: Driving and monitoring provisional trust negotiation with metapolicies. In: 6th IEEE Policies for Distributed Systems and Networks (POLICY 2005), Stockholm, Sweden, June 2005, pp. 14–23. IEEE Computer Society, Los Alamitos (2005)
10. OpenSocial Foundation: OpenSocial API v0.9 (August 2009), `http://code.google.com/apis/opensocial/`
11. Tootoonchian, A., Saroiu, S., Ganjali, Y., Wolman, A.: Lockr: better privacy for social networks. In: CoNEXT 2009: Proceedings of the 5th international conference on Emerging networking experiments and technologies, December 2009, pp. 169–180. ACM, New York (2009)
12. Sun, S.T., Hawkey, K., Beznosov, K.: Secure web 2.0 content sharing beyond walled gardens. In: Proceedings of the 25th Annual Computer Security Applications Conference (ACSAC) (December 2009)
13. Fong, P., Anwar, M., Zhao, Z.: A privacy preservation model for facebook-style social network systems, pp. 303–320 (2009)
14. Liu, K., et al.: Towards privacy-aware opensocial applications. Google Talk (May 2009), `http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/slides/pr_google_v5_3.pdf`
15. Fausto Giunchiglia, R.Z., Crispo, B.: Ontology Driven Community Access Control. In: Proceedings of the First International Workshop on Trust and Privacy on the Social and Semantic Web (SPOT 2009), Heraklion, Greece (2009)

# Using Social Media for Ontology Enrichment

Paola Monachesi and Thomas Markus

Utrecht University, Utrecht, The Netherlands

**Abstract.** In order to support informal learning, we complement the formal knowledge represented by ontologies developed by domain experts with the informal knowledge emerging from social tagging. To this end, we have developed an ontology enrichment pipeline that can automatically enrich a domain ontology using: data extracted by a crawler from social media applications, similarity measures, the DBpedia knowledge base, a disambiguation algorithm and several heuristics. The main goal is to provide dynamic and personalized domain ontologies that include the knowledge of the community of users.

## 1 Introduction

Social media applications are accessed by millions of users that actively participate in the creation of textual and visual content, providing tags to describe the resources they have contributed. Social media begin to acquire relevance also in educational contexts with learners relying on them for learning purposes. For example, the Massachusetts Institute of Technology has a channel for posting videos on Youtube that has 57,000 subscribers and the channel page has been viewed for more than 1 million times. On YouTube, there are videos of lectures given in top universities that have more than 50,000 views.

There is thus the need to support the learners in accessing and exploiting this material in the most appropriate way. One possibility is to employ the tags provided by users to the resources in order to provide search results and recommendations about the most relevant material for the given learning task. However, [13] reports that learners don't find tag clouds particularly useful when searching for learning material since they only show relations among topics but they don't provide information about how the topics are related. Ontologies prove to be a more valuable support than tag clouds in the knowledge discovery process. They provide a clear structure which is based on relations among concepts that can be also very useful in discovering new topics and associations. More specifically, domain ontologies can guide and support the learner in the learning path, facilitate (multilingual) retrieval and reuse of content as well as mediate access to various sources of knowledge, as concluded in [14]. However, this formalization might not always correspond to the representation of the domain knowledge available to the learner which might be more easily expressed by the tagging emerging from communities of peers via available social media applications.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 166–180, 2010.

In the context of the *Language Technology for LifeLong Learning* project,[1] we propose an ontology enrichment methodology that complements the formal knowledge represented by domain ontologies with the informal knowledge emerging from tagging. More specifically, in our approach, we include the expert view on the domain by maintaining the ontology structure but we complement it with the 'wisdom of the crowd' emerging from tagging. We provide thus more dynamic ontologies that take into account the evolving vocabulary of the Community of Practice. Similarity measures have been evaluated and are employed to identify tags which can be related to the concepts of an existing domain ontology. A knowledge base such as DBpedia [1] is used in order to map the tags into the ontology in combination with a disambiguation mechanism. However, tags are also related to users providing information about their interests, their knowledge and their level of expertise within a domain. The MOAT ontology is employed to create a link between users and the meaning of various tags, allowing thus the identification of the vocabulary of Communities of Practice.

The paper is organized as follows. Section 2 discusses the state of the art. Section 3 introduces the ontology enrichment process and its various components. Section 4 discusses the role that similarity measures can play in enhancing ontologies with tags while section 5 focuses on reference knowledge bases such as DBpedia to map the relevant tags into an existing ontology. Section 6 presents the approach to tag disambiguation adopted and the methodology employed to create ontologies related to Communities of Practice. In section 7, we evaluate the resulting ontology. Finally, in section 8, we discuss some future work and perspectives.

## 2 State of the Art

There are two main approaches to structure the tags extracted from social media applications in order to organize them and to understand their meaning. The former approach relies on the information that can be retrieved from the social media applications such as users, tags and tagged resources. In particular, [17] and [8] developed algorithms to derive a hierarchy of tags, based on the data of a tripartite tagging network. Three measures for relatedness are compared in [4]: one measure is based on co-occurrence, one is based on the cosine similarity, and then there is the FolkRank algorithm. The measures are applied to find a set of closely related tags. Subsequently, the authors propose a mechanism of semantic grounding: the found tags are mapped to WordNet, in order to inspect the semantic distance between the related tags. Cosine similarity appears to yield more synonyms, where the other two measures rather yield different concepts, among which are superconcepts, which make them appropriate for retrieving taxonomic relationships. FolkRank, in addition, is capable of detecting multi-word lexemes from distinct tags. In [20], the tag-resource-user relations are represented as multidimensional vectors and they use a probabilistic model to find categories of knowledge.

[1] http://www.ltfll-project.org/

The latter approach exploits external semantic resources to structure sets of tags. An example is provided by [19] that tries to make explicit the semantic structure of tagging for semantic web applications. They describe an approach using tag preprocessing (morphologic similarity, exclusion of isolated tags), statistical tag clustering based on co-occurrence, and relation identification by looking up terms in online ontologies. A similar approach is described in [6] which, however, focuses on how an actual ontology can be generated on the basis of a folksonomy. They propose that in addition to providing the tags, the community can also directly help to identify or judge/approve relations in the ontology.

The work presented in this paper relies on the techniques previously described and integrates them in a comprehensive and automatic approach to ontology enrichment in which similarity measures are combined with external semantic resources. In addition, while the approaches mentioned above are an attempt to develop (light) ontologies from sets of tags, our proposal differs from them because it relies on existing domain ontologies. More specifically, we embed the tags extracted from social media application into the structure of an existing ontology. It is thus possible to exploit the growing number of ontologies available as result of the Semantic Web initiative and enhance them with the extended vocabulary of Community of Practices arising from social data. This is a more relevant use of existing ontologies than that suggested in [19], in which ontologies are exploited to discover possible relations among tags. Existing ontologies are normally limited in size and in domain, making it quite difficult to find enough relations. We believe that a large background knowledge base such as DBpedia [1] is much more appropriate for the relation discovery task.

## 3 Ontology Enrichment with Social Data

Domain specific ontologies are relevant for learners, since they offer structured information on a certain topic which is lacking in large background semantic resources, such as DBpedia.

Domain ontologies, however, might be too static since it is quite demanding to update them on a regular basis. They usually represent the conceptualization of a domain by experts. However, the conceptualization and the vocabulary employed by the expert might differ substantially from that available to the learner. We have developed a methodology that allows for the enrichment of an existing ontology on the basis of the vocabulary of the Community of Practice that the learner is part of. More specifically, the resulting ontology integrates socially relevant concepts within the structure of an expert view domain ontology. This makes the enriched domain ontology more accessible for use by a variety of learners.

In [13], experimental evidence is provided to show that an ontology enriched with social tags constitutes a useful approach in the context of a learning task. Both beginners and advanced learners agreed that it can be a valuable tool in knowledge discovery tasks since it can provide structure to the heterogeneous list of documents which constitutes the output of search engines.

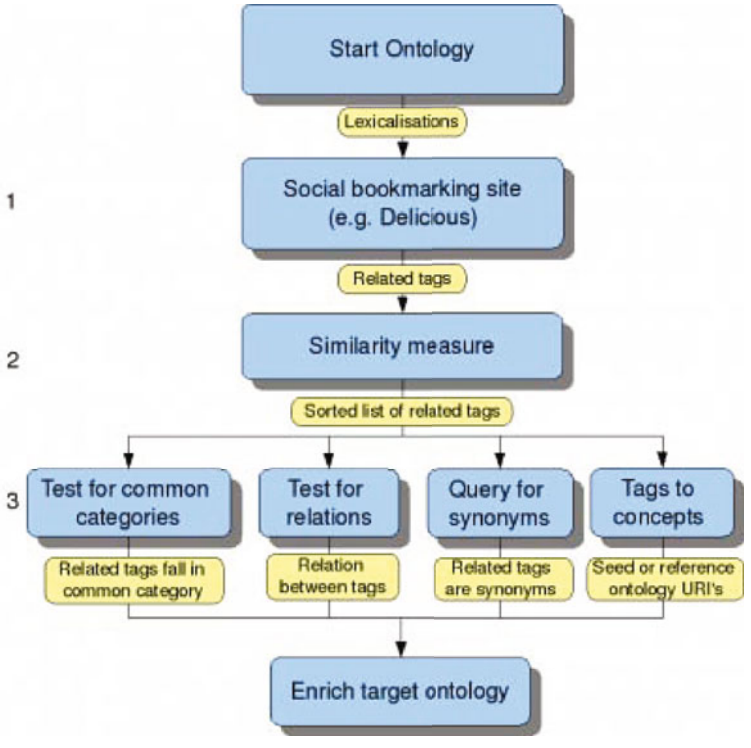



**Fig. 1.** The ontology enrichment pipeline

We have developed an ontology enrichment pipeline that can automatically enrich a domain ontology using data extracted by a crawler from social media applications, similarity measures, the DBpedia knowledge base, a disambiguation algorithm and several heuristics, as illustrated in figure 1.

The figure shows the steps involved in the enrichment process. More specifically, we have taken as starting point the LT4eL domain ontology on computing that was developed in the Language Technology for eLearning project.[2] It contains 1002 domain concepts, 169 concepts from OntoWordNet and 105 concepts from DOLCE Ultralite. The connection between tags and concepts is established by means of language-specic lexicons, where each lexicon specifies one or more lexicalizations for each concept [10].

As first step in the process, we extract data by means of a crawler that uses APIs provided by the social networking applications to get information about users, resources and tags. The crawler extracts links to resources from social media applications such as Delicious, YouTube and Slideshare together with the tags used to classify the resources and information about the social connections developed inside these web sites. The data extracted by the crawler can be interpreted as a folksonomy, which is a hypergraph describing the information

[2] http://www.lt4el.eu

about users, resources and tags, as specified in [11]. In the second step of the enrichment process, similarity measures are employed to identify tags that are related to the lexicalization of concepts already existing in the LT4eL domain ontology (cf. section 4). In the last step, we attempt to identify relations among the existing LT4eL concepts and new concepts derived from the tags, by relying on a background ontology such as DBpedia. Several heuristics are employed to discover taxonomic relations, synonyms and new relations explicitly coded in DBpedia (cf. section 5). Ambiguities are resolved through an appropriate disambiguation algorithm (cf. section 6).

## 4 Similarity Measures

Various similarity measures have been investigated in order to assess their possible contribution to automatic ontology enrichment. More specifically, our goal was to test which measures would allow for identification of more specific terms, alternative lexicalizations of pre-existing concepts and whether it would be possible to identify the relevant domain in case of ambiguity.

The similarity measures were applied to a large Delicious dataset which was previously aggregated with the crawler described in the previous section. It contains 598379 resources, 154476 users and 221796 tags on a wide range of subjects, but with an emphasis on computer related terminology.

In the rest of this section, we describe the various algorithms that were implemented as web services in order to assess their possible application in the ontology enrichment process.

*Co-occurrence.* In order to implement co-occurence, a tag-tag co-occurrence graph is calculated. This is a weighted, undirected graph. Two tags are connected if there is at least one post containing both tags. The weight of an edge is given by the number of posts that contain both t1 and t2. Given a tag t, all tags t2 such that weight (t,t2) is maximal gives a set of most related tags. This measure provides valuable input to extract taxonomic relationships between tags, as indicated by [4]. The co-occurrence measurement is also used by [18]. However, instead of using the measurement as described above, they point out that this measure should be normalized. There are two normalization methods:

- Assymmetric
- Symmetric: According to the Jaccard coefficient

The notion of co-ocurrence has been further developed into resource co-ocurrence and user co-ocurrence. In user co-ocurrence, the individual users are taken into account when calculating the co-ocurrence scores. A tag only co-occurs with another tag if that specific user actually added the two tags. This is the type of coocurrence that is defined by [4]. This is different from resource co-ocurrence where tags are said to co-occur when added to the same resource (by different users).

*Cosine similarity.* Given two vectors, cosine similarity is used to compute the similarity between two tags. The vectors can be computed in different ways, which leads to the distinction of different approaches.

- Tag Context Similarity: For each tag, a vector is created with as length the number of tags. The weights are the co-occurrence values of two tags. [4] points out that this method is suitable for finding synonyms and [19] used this measure to cluster tags.
- Resource Context Similarity: For each tag, a vector is constructed with as length the number of resources. The number of times the tag is used to annotate a resource determines the weight (tf). In [4], it is showed that this method is also suitable for finding synonyms. In [7], tf * idf is used in addition to the original method (tf). In all cases, they found tf * idf to be superior. They tried three different methods using this measure to cluster tags: hierarchical clustering, maximal complete link clustering and k-means clustering. The latter one yielded poor results, whereas hierarchical clustering had the best performance.
- User Context Similarity: For each tag, a vector is constructed with as length the number of users. Number of times the tag is used by a specific user, determines the weight.
- Document-Term similarity: This method is used in [3] that has applied this technique on Technorati Data. Similarity is calculated here from textual similarity of documents they annotate. This method is therefore not useful when tags are applied to images, videos etc. They induced a hierarchy of tags using similarity of the articles that were tagged.

### 4.1 Evaluation of Similarity Measures

We have evaluated the various similarity measures in the context of the ontology enrichment process and we have taken into account how many users and resources are necessary to obtain appropriate results. We have created a standard set of evaluation tags for which we have verified that our aggregated dataset contains enough information. This set contains 12 terms within the computing domain with different levels of abstractness. Since some measures can return thousands of results which would take too long to be evaluated manually, the analysis focused on the first 20 items. This means that for each of the measures 12*20=240 results were analyzed. The 12 standard test terms were: java, docbook, xml, xhtml, css, tex, standards, linux, design, blog, tools, software.

All the coocurrence measures were applied to this standard list and their results were analyzed. Two domain experts evaluated whether the output from the similarity measures consistently matched one or more of the possible output criteria. The different similarity measures were rated on a 5-point scale for each of the following criteria:

- Does the measure return concepts which are similar to the input term (e.g. java and jre)?

| Similarity method | Similar concepts | Synonyms | Tail useable | Close in hierarchy |
|---|---|---|---|---|
| Resource Coocurrence (Jaccard) | 5 | 1 | 1 | 4 |
| Resource Coocurrence (Assymetric) | 3 | 1 | 1 | 1 |
| User Coocurrence (Jaccard) | 5 | 1 | 1 | 5 |
| User Coocurrence (Assymmetric) | 3 | 1 | 1 | 1 |
| Resource Cosine Similarity | 5 | 3 | 1 | 4 |
| User Cosine Similarity | 3 | 1 | 1 | 3 |

**Fig. 2.** Results of the evaluation of similarity measures

- Does it reliably list synonyms at the top of the result list (e.g. cpu and processor, javascript and ecmascript) ?
- Is it possible to find a pattern (spelling error, unrelated term) (html and cookies) within the results found in the tail (items with the lowest score)?
- Are the related tags close to each other in the ontological hierarchy, taking the existing LT4eL domain ontology as a point of reference (e.g. xhtml and xml)?

The table in fig. 2 gives an overview of how the similarity measures perform with respect to each of the criteria mentioned above.

It shows that the different normalization methods for co-ocurrence greatly influence the results returned. A detailed analysis of the results, indicates that the data could be very useful in a manual enrichment of the ontology. However, the results are less useful if the goal is an automatic ontology enrichment process, as in our case. For example, the first hits for asymmetric co-ocurrence are very generic which are of little value because the relation to the input term is too trivial.

User co-ocurrence was found to be roughly equivalent to resource co-ocurrence for larger number of resources and users [13]. The results suggest that a small number of users doesn't need to be a problem with respect to the representativeness of the result as long as enough resources are tagged. If the user co-ocurrence similarity measure is employed, a sample of about 10-15 users and about 200 resources seems to be sufficient for a precision of about 0.75 when compared to the results from resource co-ocurrence. This result was determined by implementing custom tools which could automatically query our web services, gather and average the results for various numbers of users and of resources. A more extensive description can be found in [13].

The table in fig. 2 shows that none of the similarity measures was able to reliably discover synonyms. We concluded that this was due to the fact that our test set didn't contain terms which have widely used synonyms. Another set of 5 terms was created that did have clear synonyms (e.g. CPU/processor). These additional terms were then used to re-evaluate the cosine-based measures to see whether they would reliably return synonyms, because the literature strongly suggests that they are suited for this task. In some cases, these measures can indeed be used to identify synonyms. However, their position in the result list is unreliable. For example, if we consider the differences in lowercase/uppercase, the same tags appear in different forms in the repository (e.g. 'java', 'Java' etc.). We would expect that given a tag 'java', the other form 'Java' also appears

(high) in the list. In exactly 50% of the cases, we find a tag in the related tags list (somewhere in the top-20), which only differs in uppercase/lowercase. The position in the top-20 ranges from 3rd to 20th.

In order to improve the ranking of familiar or unfamiliar concepts, we also did experiments which take the term frequency and inverse document frequency into account (i.e. tf versus tf * idf). After an evaluation based on our 12 test terms, we concluded that the results of both methods were almost similar. We did not find any advantage using tf * idf.

## 5 Reference Ontologies

Even though the application of the various similarity measures didn't allow for an automatic interpretation of the data in the computing domain, we have used it as first step in the ontology enrichment process. More specifically, given a seed tag in the LT4eL computing ontologies, similarity measures can be employed to find additional related tags that can be used to enhance the ontology.

The main goal is to include information that is relevant to a learner and his peers in the existing domain ontology structure. Tagging systems provide us with a domain vocabulary which is validated as common knowledge by the community that has produced it. The information implicitly contained in tag collections can be employed to assess how relevant a term is in a given domain. Similarity measures allow us to select possible lexicalizations of concepts which are related to the existing ones in the ontology, and which we consider to be 'socially relevant' with respect to the input lexicalisation. More specifically, we have chosen to adopt the resource coocurrence measure in our system for efficiency reasons and wide use in the literature.

However, if we want to map the related terms identified by similarity measures to the concepts present in the ontology, we still face the problem of identifying the appropriate relations. To this end, several heuristics are employed. They heavily rely on the use of a large knowledge base such as DBpedia.

For example, we employ DBpedia to assess whether a related tag can be considered a new concept or a lexicalization of an existing one. By making use of the SKOS vocabulary [12], we can differentiate between a preferred lexicalization (the head term) and additional lexicalizations (i.e. popular and alternative terms for the same concept). The *rdf:type* assertion between a DBpedia resource and a resource from some other ontology can be used to infer that the DBpedia concept is actually a sub-concept of the object of that statement and should be added as such to the seed ontology.

DBpedia also contains a category structure and a list of all the DBpedia concepts and other categories present in such a category hierarchy. We can automatically calculate the closest shared categories for two concepts and return them.

To summarize with an example: given the pre-existing domain ontology concept 'XHTML', the similarity measure system generates the tag 'xslt' which is attested in DBpedia as a resource (i.e. a concept) and it shares the category

'XML' with the 'XHTML' concept. Given that the category 'XML' is already a concept present in the domain ontology the new concept 'XSLT' can be added as a subclass of it.

The resulting ontology integrates the socially relevant concepts within the structure of an expert view domain ontology. Methods that derive ontology-like structures from tag systems such as those described in section 2 cannot provide high quality of results. This is due to the unavailability of explicit structural information in folksonomies. On the contrary, this structural information has been made explicit in ontologies and our approach relies on it.

## 6 Tag Disambiguation

The ontology enrichment approach discussed in the previous sections can be employed to enrich an ontology with unambiguous terms such as 'HTML' while this is not the case for 'Java' (both a programming language and an island in Indonesia amongst other things).

The ontology enrichment pipeline is employed to enhance an ontology in the computing domain which is relatively unambiguous. However, even in this domain several exceptions are attested. For example, out that of the 7231 tags, resulting as output of similarity measures, only 1271 are unambiguous while 5960 are ambiguous. In the latter case, disambiguation is crucial in order to properly map tags to concept.

An interesting approach to disambiguation is proposed in [16]. They use Tagpedia, which is a system based on a Wikipedia corpus in order to disambiguate tags. More specifically, disambiguation is carried out by relying on the distance in the text between two tags. An alternative approach is described in [5], in which the agreement between two concepts is not calculated directly, due to efficiency reasons. He considers instead the common Wikipedia categories related to the ambiguous terms as a way to disambiguate.

In our approach, we make use of the Wikipedia disambiguation page to obtain possible interpretations of a term. We rely also on the structural information (i.e. pagelinks) that is available in Wikipedia in order to construct a network of interrelated meanings. In Wikipedia, pagelinks specify relations among various articles that, for our purpose, we consider concepts. The basic assumption behind our disambiguation approach is that there is a correspondence between the various tags that people associate with resources in social media applications and the pagelink structure in Wikipedia.

We have thus deviated from the work in [5] which is dependent on Wikipedia category information for disambiguation. This implies that we are in a position to disambiguate concepts which lack proper categorisation. We believe that this should be more suitable for our task because it can deal with incompatible sets of meanings.

To exemplify our approach to disambiguation, consider a user that has tagged a resource with the tags: *python* and *ruby*. Both *ruby* and *python* are ambiguous terms, where *ruby* could refer to things such as an expensive jewel or a

programming language and *python* could mean a specific species of snake or a programming language. By considering the different concepts (i.e. Wikipedia articles) associated with the terms *python* and *ruby* and pagelinks contained in the articles a choice can be made between the various interpretations. As in previous work, we employ at least two terms in order to disambiguate.

The disambiguation algorithm is illustrated in Figure 3. More specifically, we have developed a new approximation algorithm inspired by social network analysis which is able to disambiguate the tags that are shared by a single tagging instance. We start by retrieving the set of meanings which are associated with each input term (Figure 3, step 1) through the disambiguation page in Wikipedia. A term is considered ambiguous if we retrieve more than one meaning from Wikipedia.

For example, all the possible interpretations for the input terms *C*, *D*, *Ruby*, *Python* and *Perl* are retrieved. The term *Ruby* has interpretations such as: programming language, gem, Ruby MRI, Ruby Wax and hardware design language. *Python* has other interpretations such as: programming language and snake. *C* and *D* are also highly ambiguous. Only the term *Perl* is unambiguous and its only interpretation is a programming language.

After determining the possible concepts for the input term, the pagelinks between these concepts are retrieved.

This graph of interconnected meanings is then processed by a graph layout algorithm (Fruchterman-Reingold) which clusters concepts with strong ties together (Figure 3, step 2).

After the graph layout algorithm has assigned each concept a fixed location in the graph, a Self Organising Map [9] based clustering component is applied. It divides the graph into separate clusters of interconnected concepts (Figure 3, step 3). This step is necessary: considering only unconnected clusters of concepts is not sufficient (as illustrated in the graphical example).

In our example, we obtain a cluster which includes ruby programming language, python programming language and ruby hardware design language, Ruby MRI and Perl. They are in the same cluster because pagelinks occur relatively often between them. On the other hand, ruby gem and python snake will appear far apart from the other concepts, because they lack pagelinks to those concepts.

The centrality of each concept inside its own cluster is calculated next (Figure 3, step 3, nodes with underlined text). Centrality is a measure that indicates which node (concept) is most central or 'important' in its cluster [2]. The centrality values are used to reduce the number of concepts in the cluster. Concepts with the highest centrality value for the associated term are retained. All the others are removed from the cluster.

This means that the concepts Ruby hardware design language and ruby MRI, both originated from the term Ruby, will be removed from the cluster. Only Perl, ruby programming language and python programming language will remain in the cluster after filtering by centrality.

Subsequently the clusters are sorted by the number of concepts still present in them. The idea is that each cluster now contains the maximum amount of

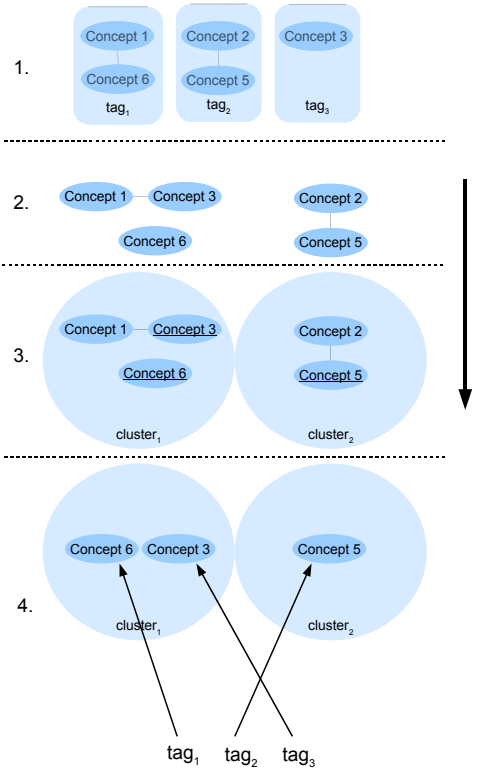



**Fig. 3.** Disambiguation process

central and coherent meanings for the largest number of terms. Term concept assignments (Figure 3, step 4) will start with the cluster which contains the largest number of concepts (ruby programming language, python programming language, Perl). The concepts from this cluster are paired with their respective input terms and the terms are thus disambiguated.

The next cluster will be chosen by the the number of terms not yet disambiguated. The rationale for not selecting the next-largest cluster is that it could contain conflicting concepts. Disregarding concepts belonging to already disambiguated terms will retain truly complementary clusters instead of conflicting ones. In the example considered, this criterion leaves C and D to be available for disambiguation.

In order to link tags extracted from social media application to their related concepts the following process is applied: For every resource, the tags added by each individual user are considered. Such a collection of tags is called a *Tagging*-instance in the SCOT vocabulary. The tags associated with the Tagging-instance are processed by the disambiguation algorithm and result in a list of term-concept pairs. These term-concept pairs are then stored using the MOAT ontology [15]. It is thus possible to differentiate between *global meanings* as the

list of all meanings that could be related to a tag in a folksonomy space and *personal meanings* related to a specific user or Community of Practice (CoP). As a result not only the meaning of a given tag is available, but also who assigned this meaning.

```
<tag:RestrictedTagging>
  <tag:taggedResource
     rdf:resource="http://www.python.org"/>
    %use a different page in which python is not present
  <foaf:maker
     rdf:resource="http://userdirectory.example.com/Mary"/>
  <tag:associatedTag
     rdf:resource="http://delicious.com/tag/python"/>
  <moat:tagMeaning
     rdf:resource="http://dbpedia.org/resource/Python_(progr_language)"/>
</tag:RestrictedTagging>
```

**Listing 1.** MOAT example

This possibility is especially relevant in our eLearning application since we can identify the meaning that is common to a group of users sharing a specific interest (i.e. a Community of Practice). The MOAT ontology allows us to model the differences in meaning that emerge in the disambiguation process. The applications in our eLearning domain are obvious. The information can be employed to provide more appropriate search results for the learning material and in addition it becomes possible to identify communities with a superficially similar vocabulary which are actually distinct.

## 7 Evaluation

In order to evaluate our ontology enrichment methodology, we have compared three different ontologies:

1. the LT4eL computing ontology with the related English lexicon (1200 classes);
2. a manually enriched ontology which takes the LT4eL one as basis (1336 classes and 1672 lexical entries). This is our gold standard.
3. the automatically enriched ontology, which takes the original LT4eL ontology as basis. (2016 classes and 2325 lexical entries)

A first analysis of the lexical differences between (1) and (2) shows a difference of 80 lexicalisations. The aim of our evaluation was to assess whether the automatic enrichment process would add lexicalisations (and related concepts) that overlap with manually added lexicalizations given a similar sub-domain.

The automatically enriched ontology has been generated by considering each coocurring tag in our Delicious data set as eligible for enrichment. Even though we considered every coocurring tag as eligible for use in ontology enrichment, the lexical overlap between the manually enriched ontology and the automatic one is minimal. More specifically, 69 terms which have been added manually

to the LT4eL ontology are multi-word units and are not attested in Delicious. They are representative of the expert view of the domain given their level of specificity and include terms such as: NMTOKEN attribute, XML element type declaration, XML attribute list declaration. The remaining 21 terms are attested in Delicious but only 13 of them are generated by the similarity measures and are attested in DBPedia.

Regardless of the minimal lexical overlap between the manually and the automatic enriched ontology, it is not the case that the terms added automatically are not appropriate and are misplaced in the ontology, as the following evaluation (that filters upper ontology concepts) reveals:

- Total number of unique statements: 1265
- Accurate enough for ontology enrichment: 1010
- Too inaccurate: 255

This brings the amount of usable additions to about 80%. We have analyzed the added relations further and discovered that:

- Relations with the very general *ltfll:related* relation: (598). The 'related' label only indicates that two terms are related but it doesn't say in which way.
  - Correct: 497 (83%)
  - Incorrect: 101
- Clear ontological relations (*rdfs:subclassof* or either DBpedia specific ones): 667
  - Correct:513 (77%)
  - Incorrect: 154

There is minimal overlap between the ontology produced by means of a manual enrichment process carried out by an expert and our automatic enrichment process. The latter includes the vocabulary of the community of users, while the former includes very specialized tags provided by an expert. It is exactly this complementarity that we wanted to achieve by embedding tags into an existing ontology and that we want to exploit in eLearning applications.

In addition to this quantitative evaluation, we have run an experiment focused on support provided by an ontology enhanced with tags in comparison with tag clouds extracted from Delicious (lacking ontological structure) in the context of a learning task. The underlying assumption is that conceptualization can guide learners in finding the relevant information to carry out a learning task (a quiz in our case). The hypothesis was that learners might differ with respect to the way they look for information depending on whether they are beginners or more advanced learners. While beginners might profit from the informal way in which knowledge is expressed through tagging, more advanced learners might profit from the way knowledge is structured in an ontology. In general, both advanced learners and beginners profited from the clear ontology structure present in the graph. However, beginners relied mainly on documents to find the relevant information both in the case of the enhanced ontology and in the case of the cluster of related tags. The advanced learners made more use of the enriched ontology and used less documents. We refer to [13] for additional details on the results of the experiment and their interpretation.

## 8 Conclusions

One of the goals of the *Language technology for LifeLong learning* project is to develop services that facilitate learners and tutors in accessing formal and informal knowledge sources in the context of a learning task. To this end, a Common Semantic Framework has been developed in which domain ontologies constitute the core element. There are obvious shortcomings in the use of ontologies in this context: they are too static since they model the knowledge of the domain at a given point in time, they might be incomplete or might not correspond to the representation of the domain knowledge available to the learner.

We have proposed an ontology enrichment pipeline to overcome some of these problems by exploiting social data which are crawled from existing social media applications. In our approach, we include the expert view on the domain by maintaining the ontology structure but we complement it with the 'wisdom of the crowd' in order to provide more dynamic ontologies that take into account the evolving vocabulary of the community.

The domain ontology enriched with social tags constitute the basis of the semantic search implemented to retrieve formal and informal resources. However, we have also exploited the possibility of a socially driven search by employing tags and social networks in the recommendation of learning material. Our ultimate goal is to integrate the strong features of both types of searches, that is the structured information contained in ontologies with the social information coming from the tags and social networks. In addition, the creation of ontologies related to a Community of Practice allows for the recommendation of experts that can support learners in their learning tasks.

## References

1. Auer, S., Bizer, C., Lehmann, J., Kobilarov, G., Cyganiak, R., Ives, Z.: 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)
2. Brandes, U.: A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology 25(2), 163–177 (2001)
3. Brooks, C.H., Montanez, N.: Improved annotation of the blogosphere via autotagging and hierarchical clustering. In: WWW 2006: Proceedings of the 15th international conference on World Wide Web, pp. 625–632. ACM, New York (2006)
4. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic grounding of tag relatedness in social bookmarking systems. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 615–631. Springer, Heidelberg (2008)
5. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: Proceedings of EMNLP-CoNLL, pp. 708–716 (2007)
6. van Damme, M., Hepp, K., Siorpaes, K.: Folksontology: An integrated approach for turning folksonomies into ontologies. In: Proceedings of the ESWC Workshop Bridging the Gap between Semantic Web and Web 2.0, Innsbruck, Austria. Springer, Heidelberg (2007)

7. Gemmell, J., Shepitsen, A., Mobasher, B., Burke, R.: Personalization in Folksonomies Based on Tag Clustering. In: Intelligent Techniques for Web Personalization & Recommender Systems (2008)
8. Heymann, P., Garcia-Molina, H.: Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Stanford InfoLab Technical Report 2006-10 (2006)
9. Kohonen, T.: The self-organizing map. Neurocomputing 21(1-3), 1–6 (1998)
10. Lemnitzer, L., Simov, K., Osenova, P., Mossel, E., Monachesi, P.: Using a domain-ontology and semantic search in an eLearning environment. In: Proceedings of The Third International Joint Conferences on Computer, Information and Systems Sciences, and Engineering (CISSE 2007). Springer, Heidelberg (2007)
11. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Journal of Web Semantics 5(1), 5–15 (2007)
12. Miles, A., Matthews, B., Wilson, M., Brickley, D.: SKOS Core: Simple knowledge organisation for the web. In: Proceedings of the International Conference on Dublin Core and Metadata Applications, pp. 12–15 (2005)
13. Monachesi, P., Markus, T., Mossel, E.: Ontology Enrichment with Social Tags for eLearning. In: Cress, U., Dimitrova, V., Specht, M. (eds.) EC-TEL 2009. LNCS, vol. 5794, pp. 385–390. Springer, Heidelberg (2009)
14. Monachesi, P., Simov, K., Mossel, E., Osenova, P.: What ontologies can do for eLearning. In: Proceedings of International Conference on Interactive Mobile and Computer Aided Learning, IMCL 2008 (2008)
15. 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)
16. Ronzano, F., Marchetti, A., Tesconi, M., Minutoli, S.: Tagpedia: a semantic reference to describe and search for web resources. In: Proc. of The Workshop Social Web and Knowledge Management of the World Wide Web Conference, vol. 8, pp. 19–25 (2008)
17. Schmitz, P.: Inducing Ontology from Flickr Tags. In: Proceedings of the Collaborative Web Tagging Workshop at the 15th WWW Conference (WWW 2006), Edinburgh, Scotland (2006)
18. SigurbjŽrnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: Proc. 17th Intl. Conf. on World Wide Web, pp. 327–336 (2008)
19. Specia, L., Motta, E.: Integrating Folksonomies with the Semantic Web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 624–639. Springer, Heidelberg (2007)
20. Wu, X., Zhang, L., Yu, Y.: Exploring Social Annotations for the Semantic Web. In: Proc. of WWW 2006 (2006)

# Representing Distributed Groups with $_{dg}$FOAF

Felix Schwagereit, Ansgar Scherp, and Steffen Staab

WeST Institute, University of Koblenz-Landau, Germany
{schwagereit,scherp,staab}@uni-koblenz.de

**Abstract.** Managing one's memberships in different online communities increasingly becomes a cumbersome task. This is due to the increasing number of communities in which users participate and in which they share information with different groups of people like colleagues, sports clubs, groups with specific interests, family, friends, and others. These groups use different platforms to perform their tasks such as collaborative creation of documents, sharing of documents and media, conducting polls, and others. Thus, the groups are scattered and distributed over multiple community platforms that each require a distinct user account and management of the group. In this paper, we present $_{dg}$FOAF, an approach for distributed group management based on the well known Friend-of-a-Friend (FOAF) vocabulary. Our $_{dg}$FOAF approach is independent of the concrete community platforms we find today and needs no central server. It allows for defining communities across multiple systems and alleviates the community administration task. Applications of $_{dg}$FOAF range from access restriction to trust support based on community membership.

## 1 Introduction

An increasing number of Web 2.0 platforms providing social networking and community building functionality are available. Users participate in multiple of such platforms and share information with different groups of people of their social network such as the professional network, sports club, colleagues, special interest groups, friends, family, and others. This is due to network effects, i.e., users need to join platforms other users are already members in. In addition, the platforms provide different functionality and have different scope such as building a professional network, sharing media, or conducting polls. In order to conduct the tasks within a specific group of the social network, multiple community platforms need to be used. Thus, the users require accounts in several community platforms and the groups the users are members in are scattered over different of such platforms.

Communities are considered in this work as set of people that share a common purpose [7]. This is the case for a long-lasting professional organization like the W3C, which has the goal of creating web standards, as well as for an ad-hoc created group of private persons organizing a party. The members of a community may be recognized by their interests, behavior, or contacts. Thus they form an implicit community since their membership is not explicitly defined.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 181–195, 2010.

If membership in a community is managed intentionally, an explicitly defined community is formed. This paper focuses on communities that explicitly specify their members. In the following, such communities are called *groups*. A person may be involved in many different groups such as professional networks, sports clubs, and groups with specific interests. If these groups are distributed over different systems, the management of group memberships becomes a tedious task. Therefore, a decentralized approach for managing group memberships is an appropriate way to tackle these problems.

The $_{dg}$FOAF approach assumes the following setting: People can describe themselves with the Friend-of-a-Friend (FOAF) vocabulary [3]. We assume that personal descriptions as well as other statements are located at Linked Open Data (LOD) stores [4] owned by the particular person. So once the location of the self description of a particular person is known and confirmed (e.g., using signed graphs [10]) it can be assumed that all the data available at this location is stated by this person. Based on this self description, several approaches like OpenID [8] or FOAF+SSL [9] exist to authenticate persons. Different Web-based services can use these authentication mechanisms for providing Web 2.0 functionality as indicated in [13]. The contribution of this paper is $_{dg}$FOAF, a policy based mechanism for defining distributed groups and for determining group membership. $_{dg}$FOAF provides functionality sufficient to support typical Web 2.0 scenarios of group administration as well as other Internet-based collaboration tools.

## 2 Motivating Scenario

In a typical group management scenario, several people need to organize their actions to pursue a common goal, which is in our case the organization of an event. The scenario is motivated from the "Consumer Social Group" use case of the WeKnowIt[1] project: The 1999 class of the Westpark High School graduated 10 years ago. Alice thinks it is time for a class reunion. She asks her former classmate Bob to help her organizing such an event. (i) To prepare the class reunion, Alice and Bob set up a group named WestparkHigh99, which is going to be populated by all known members of their old class. (ii) Together Alice and Bob form the organizing committee for the class reunion, the group WestparkHigh99Committee. (iii) During their ongoing preparations Alice and Bob persuade Carol to join the organizing committee and further classmates are found. (iv) Carol adds Ronald and Bob adds Simon to the group. Over time, the group grows as more and more classmates are discovered. Later Alice realizes that Ronald is not the person they had assumed. (v) Since Carol is temporarily not available and cannot correct her mistake, Alice removes Ronald from the group. (vi) Alice, Bob, and Carol set up online services like a photo sharing system to collect old photos and a scheduler to find a date for their reunion. These services exist on different specialized platforms like Flickr (`http://flickr.com`)

[1] `http:www.weknowit.eu`

and Doodle (http://doodle.com). Access to these services is granted only to members of the WestparkHigh99 group.

## 3 Problem Areas of the Scenario

We investigate how the scenario can be implemented with currently available means. We focus on the required infrastructure and the actions that are carried out by the involved people. These actions are creation of groups and modification of group membership. Also the group membership of people is determined in order to decide whether they are allowed to access specific services.

### 3.1 Centralized Solution

In a centralized solution, Alice would start to search for online services she needs for organizing the reunion. Probably she wants to use the services "Photo Sharing" and "Meeting Scheduler" along with other services. Then she would create a group WestparkHigh99 at every chosen service with herself as administrator. Later she adds Bob and Carol as administrators for her group at each service and the administrators add the former classmates to each of the services. So each service can determine through its own group structures whether a specific person is allowed access and using it.

The actions are depicted in Figure 1 a) and numbered in the order of execution: (1) create group WestparkHigh99, (2) add Bob as admin for WestparkHigh99, (3) add Carol as admin for WestparkHigh99, (4) add Ronald as member of WestparkHigh99, (5) add Simon as member of WestparkHigh99, (6) remove Ronald from the group WestparkHigh99, (7) determine membership in WestparkHigh99. As shown in Figure 1 a), all actions need to be carried out for each service platform separately. Also the users have no common control over

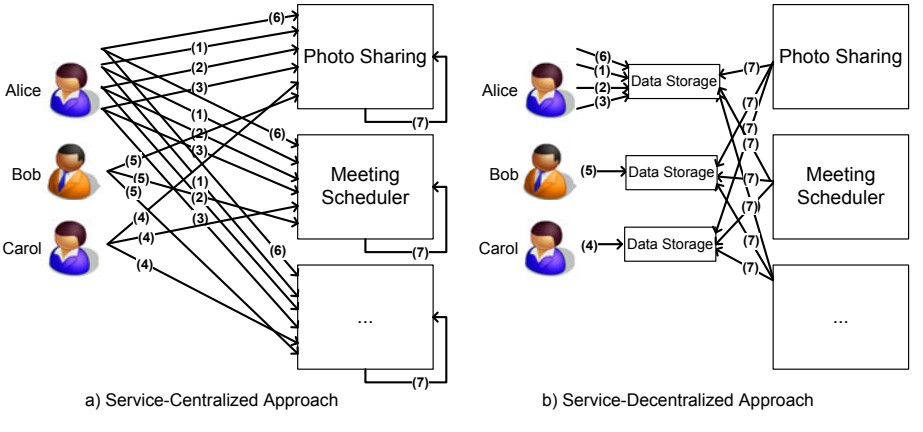



**Fig. 1.** Service-Centralized Approach vs. Service-Decentralized Approach

changes of the end user policies at the service platforms. Once they created all group structures they are locked in that platform, because otherwise they would have to recreate all the structures again on another platform. Thus, the centralized solution where all actions are centered around individual service platforms has several disadvantages:

- High effort to maintain different identical group structures
- Error-prone because of redundancy
- Lock-in effect

### 3.2 Decentralized Solution

To overcome the disadvantages of the centralized solution, we separate the group management from the services that are used by the group to carry out their tasks. In such a decentralized solution, the actions for the group management need to be carried out only once as depicted in Figure 1 b). The numbers match the actions described in the previous section.

In the decentralized solution a data storage is introduced, which is open for online read access but owned by one person. Every person can conduct group management tasks by modifying her own data storage. This requires fewer actions from the people involved. On the other side, the platforms are required to conduct more actions, since the have to "construct" the current status of the group from different data storages. New questions arise, if group management is decentralized in the way described above:

- Can multiple administrators act on behalf of one, single group? Although it is easy for every administrator to define her own group in her own data storage, this is not sufficient for multiple administrators, who cannot modify each others' data.
- Where is the group's policy located? The policy describes which actions can be conducted by whom. But since different administrators act on the same group they all need to express that they do so without any central entity.
- How can an administrator undo the action of another administrator? Since no person can modify the actions stated in data storages owned by other people, deletion is not applicable.
- How can the service platforms determine the membership of one person? This task is not trivial since data needs to be combined that comes from different data storages and might therefore be contradicting.

In a naive alternative approach, all group management tasks could be conducted on one single data storage. This would solve the problem of multiple data storages, but the problem of managing this one data storage as well as the problem of multiple platforms used by different groups would remain. So this naive approach would not completely address our problem statement. The challenges resulting from these questions are tackled with the integrated $_{dg}$FOAF approach.

## 4 Requirements

In this section, we collect requirements from the scenario while considering the open questions mentioned in the previous section. These requirements will form a baseline for the architecture and functionality of $_{dg}$FOAF. For each requirement, we also explicitly refer to the scenario in Section 2 by referring to the (<number>) of the relevant part.

1. *Group Administrators:* For a group, one administrator or a set of administrators can be defined. The administrators are allowed to modify memberships of people. In the scenario, Alice and Bob have initially created the group WestparkHigh99 with themselves as administrators (i), (ii).
2. *Granting of Membership:* Administrators can grant group membership to persons or members of other groups. In the scenario, Bob is an administrator. So he adds Simon to WestparkHigh99 (iv).
3. *Banning of Members:* Administrators can ban persons from groups. In the scenario, Alice bans the wrong Ronald from the group (v).
4. *Management of Group Administration:* New administrators can be appointed or existing administrators can resign. Since Carol becomes member of the WestparkHigh99Committee, the administrators of the WestparkHigh99 group have changed (iii).
5. *No Super Administrators:* For administration of a group, super-administrators, i.e., a person who administrates administrators, are not mandatory. In the scenario, the group WestparkHigh99Committee is not administrated by persons other than members of the group itself (iii).
6. *Determination of Group Memberships:* A mechanism needs to be provided that determines the group membership of a person based on the group definition and the actions of the group administrators. In the scenario, the services set up for WestparkHigh99 determine the group membership of every person who wants to use them (vi).
7. *Referencing of Groups:* Especially in a distributed environment, a group needs an unambiguous identifier that can be referenced by services based on group membership. The group and group memberships should be stated in such a way that, if one definition is deleted, the remaining parts of the group continue to exist. In the scenario, Bob and Carol can still add group members, although Alice's part is not available (vi).

## 5 The $_{dg}$FOAF Approach

We introduce the foundational design of $_{dg}$FOAF by distinguishing stable and changing group characteristics, which lead to the concepts of group policy and membership definition. This design is a decentralized solution described in Section 3.2.

Groups are immaterial entities. They only come into existence by making statements about them or in other words by describing them. When all statements about a group are combined, they reveal all characteristics of a group.

Certainly the most interesting characteristics of a group for practical use are its name and its members. We distinguish between two general types of group characteristics: the characteristics that are stable from creation to deletion of the group and the characteristics that can change.

In $_{dg}$FOAF all stable characteristics of a group are called group policy, which has to be stated explicitly. So, by definition, a group policy cannot change for one specific group, although it is possible to recreate a group that has a slightly different group policy. The group policy contains at least one name. Although such a name cannot be guaranteed to be unique, it can be chosen such that the group cannot be confused with another similar group. Other characteristics, which are not to be changed through the life of a group, are the rules specifying how membership can be achieved. In $_{dg}$FOAF these rules are conditions that a person has to satisfy in order to affect group membership of herself or others.

Membership of persons in a group is the only characteristic of groups covered in $_{dg}$FOAF that can change through the life of a group. By stating membership definitions a person can add or remove members from the group. In contrast to other linked data, the statements about $_{dg}$FOAF groups can be evaluated by taking into account whether the person who made the statements meets the explicit conditions in the group's policy. Therefore different and possibly conflicting membership definitions can be checked, so that for each combination of membership statements only one single interpretation can be given. Therefore the group policy acts as the basis for deciding whether a membership definition by a specific person is going to affect the group characteristics.

## 6 Concepts and Schema of $_{dg}$FOAF

Based on the requirements discussed in Section 4, we introduce all concepts of $_{dg}$FOAF that are needed to describe groups according to the design approach of Section 5. Further, we introduce a Resource Description Framework (RDF)[2] schema for serialization of these concepts and illustrate its usage with an example from the scenario. This schema is depicted in Figure 2. It contains all concepts that are needed to express $_{dg}$FOAF groups. Besides the namespace `foaf` for elements already defined in the standard FOAF vocabulary, the namespace `dgfoaf` is introduced to define elements added for $_{dg}$FOAF. From the scenario given in Section 2, we have extracted a fragment of Alice's $_{dg}$FOAF profile. It is presented in Figure 3. In the following, we introduce the concepts of $_{dg}$FOAF based on this profile.

*Persons* are atomic entities in $_{dg}$FOAF, who act as owners of data storages and therefore of the groups stored there. In the scenario, Alice is defined as a person, who is owner of the data storage at the location `http://alice.com/me.ttl`. In Alice's $_{dg}$FOAF profile shown in Figure 3 this is represented at lines 2-4.

*Groups* are entities with two explicit and stable properties. The first property is the group owner, the person who owns the group's location, i.e., the group's URI. The second stable property of a group is its group policy. So

[2] `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`

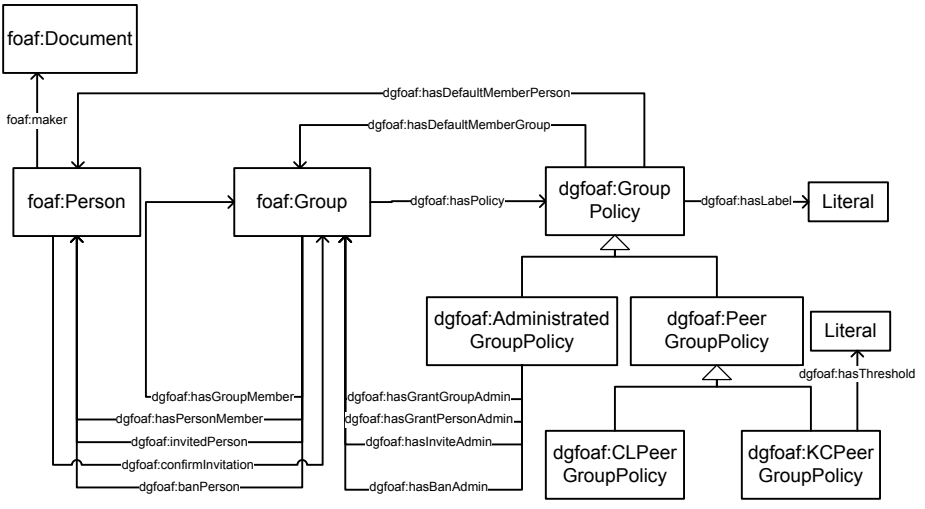



**Fig. 2.** $_{dg}$FOAF Schema

groups are structures, consisting of one person (the group owner) and one policy. In the scenario, Alice defines two groups WestparkHigh99 and WestparkHigh99Committee. These two groups can be found in Alice's $_{dg}$FOAF profile at lines 6 and 14. The corresponding policies for the two groups are referenced at lines 7 and 15.

*Group Policies* have various properties. One of these properties is their label, which is an arbitrary string. Different labels allow giving groups with identical policies different properties in order to distinguish between them. In Alice's $_{dg}$FOAF profile, two labels for two groups are defined at lines 10 and 19. Group policies can also define specific persons (defaultMemberPerson) or the members of specific groups (defaultMemberGroup) to be members of the group, without the need for membership grants. The two default member properties are available for every policy type.

Two types of group policies can be distinguished: policies for administrated groups and policies for peer groups. For *policies for administrated groups*, restrictions can be made about what administrative actions are allowed to be made by which persons. These persons need to be member of other groups, called admin groups. Admin groups can exist for granting membership to persons (grantPersonAdmin), inviting persons (inviteAdmin), granting membership to groups (grantGroupAdmin), and banning persons from the group (banAdmin). In the scenario, Alice's group WestparkHigh99 has an administrative group policy defined at lines 9-12. Lines 11 and 12 define the group WestparkHigh99Committee as admin group for granting membership and banning persons. *Policies for peer groups* can be divided into two types: While a Clique-PeerGroupPolicy (`dgfoaf:CLPeerGroupPolicy`) defines a group in which members need to be connected to every other member, the K-Core-PeerGroupPolicy (`dgfoaf:KCPeerGroupPolicy`) defines a group in which all members need to

```
1  @prefix : <http://alice.com/me.ttl#> .
2  :me a foaf:Person ;
3      foaf:name "Alice" .
4  <http://alice.com/me.ttl> foaf:maker :me .
5
6  :WestparkHigh99 a foaf:Group ;
7      dgfoaf:hasPolicy :WestparkHigh99Policy ;
8      dgfoaf:banPerson <http://ronald.com/me.ttl#me> .
9  :WestparkHigh99Policy a dgfoaf:AdministratedGroupPolicy ;
10     dgfoaf:hasLabel ``WestParkHigh99Group" ;
11     dgfoaf:hasGrantPersonAdmin :WestparkHigh99Committee ;
12     dgfoaf:hasBanAdmin :WestparkHigh99Committee .
13
14 :WestparkHigh99Committee a foaf:Group ;
15     dgfoaf:hasPolicy :WestparkHigh99ComPolicy ;
16     dgfoaf:hasPersonMember <http://bob.com/me.ttl#me>,
17         <http://carol.com/me.ttl#me> .
18 :WestparkHigh99ComPolicy a dgfoaf:CLPeerGroupPolicy ;
19     dgfoaf:hasLabel ``WestParkHigh99CommitteeGroup" .
```

**Fig. 3.** Alice's $_{dg}$FOAF Profile in Turtle Syntax

be connected to at least $K$ other members. In the scenario, Alice defines the group WestparkHigh99Committee as having a Clique-PeerGroupPolicy in line 18. Therefore every other member has to grant membership to Alice in order that Alice becomes member of WestparkHigh99Committee and vice versa.

*Membership Definitions* are assertions, made by the group owner, about the membership of other persons in her group. If the owner is allowed to assert them they result in a group membership change. Since membership definitions can change over time they are not part of the group policy. For groups with a policy for administrated groups, the following membership definitions are available: grant membership to persons (hasPersonMember), invite persons (invitedPerson), grant membership to members of other groups (hasGroupMember) and ban persons from the group (banPerson). Note that an explicit distinction between persons and groups being members of a group has to be made to avoid affects resulting from a type change of the referenced person to group. For example, if Alice grants membership to Bob, he should not be able to grant membership to other persons by simply changing the type of the entity representing him as `foaf:Person` to `foaf:Group` and adding persons to this misused group. In Alice's $_{dg}$FOAF profile she bans Ronald from being a member of WestparkHigh99 in line 8. For groups with a policy for peer groups, only the membership definition hasPersonMember is available for granting membership to other persons. In the scenario, Alice has granted such membership to Bob and Carol for group WestparkHigh99Committee in lines 16 and 17.

The elements of $_{dg}$FOAF described above are summarized in Table 1. Basic concepts are *Persons*, *Groups* consisting of one person and one policy, and *Labels*

**Table 1.** Policies and Membership Definitions for $_{dg}$FOAF Groups

**Basics:**

$$
\begin{aligned}
&Persons := the\ set\ of\ all\ Persons\\
&Labels := the\ set\ of\ all\ Strings\\
&Groups := Persons \times Policies
\end{aligned}
$$

**Policies:**

$$
\begin{aligned}
&GeneralGroupPolicyProperties\\
&\quad := \{defaultMemberPerson\} \times Persons \cup\\
&\quad\quad \{defaultMemberGroup\} \times Groups \cup\\
&\quad\quad Labels\\
&AdministratedGroupPolicyProperties\\
&\quad := \{hasGrantPersonAdmin, hasGrantGroupAdmin,\\
&\quad\quad hasInviteAdmin, hasBanAdmin\} \times Groups\\
&AdministratedGroupPolicies\\
&\quad := \{pr_i\}_{i\in\{1,\ldots,k\}} |\ pr \in AdministratedGroupPoliyProperties \cup\\
&\quad\quad GeneralGroupPolicyProperties, k \in \mathbb{N}\\
&CliquePeerGroupPolicies\\
&\quad := \{pr_i\}_{i\in\{1,\ldots,k\}}\ |\ pr \in GeneralGroupPolicyProperties, k \in \mathbb{N}\\
&KCoreGroupPolicies\\
&\quad := \{pr_i\}_{i\in\{1,\ldots,k\}}\ \cup \{\{threshold\} \times t\}\\
&\quad\quad |\ pr \in GeneralGroupPolicyProperties, k \in \mathbb{N}, t \in \mathbb{N}\\
&Policies := AdministratedGroupPolicies \cup CliquePeerGroupPolicies \cup\\
&\quad\quad KCorePeerGroupPolicies
\end{aligned}
$$

**Membership Definitions:**

$$
\begin{aligned}
&PersonMembershipDefinitions\\
&\quad := \{personMembershipDefinition, banPersonDefinition,\\
&\quad\quad invitePersonDefinition, invitationConfirmationDefintion\}\\
&\quad\quad \times Persons \times Groups \times Persons\\
&GroupMembershipDefinitions\\
&\quad := \{groupMembershipDefinition\} \times Persons \times Groups \times Groups\\
&MembershipDefinitions\\
&\quad := PersonMembershipDefinitions \cup GroupMembershipDefinitions
\end{aligned}
$$

consisting of arbitrary strings. In order to define *Policies* it is required to define their properties. *GeneralGroupPolicyPropterties* can be used in policies of every type. *AdministratedGroupPolicyProperties* are used in *AdministratedGroupPolicies*. The *CliquePeerGroupPolicies* can only have general properties, while the *KCorePeerGroupPolicies* additionally have one *threshold*. All elements of these three policy types finally represent the *Policies* in $_{dg}$FOAF. *Membership Definitions* in $_{dg}$FOAF can be distinguished in *PersonMembershipDefinitions* and *GroupMembershipDefinitions*. *PersonMembershipDefinitions* are a tuple of one policy type (person membership, ban, invitation, confirmation), the person who issued them, i.e., defined it in his FOAF file, the group that is to be addressed, and the person who is the object of this definition. *GroupMembershipDefinitions* are a tuple of the type *groupMembershipDefinitionType*, the person who issued them, i.e., defined it in his FOAF file, the group A that is to be addressed, and group B whose members is granted membership in group A.

# 7 Determination of Group Memberships in $_{dg}$FOAF

The semantics of $_{dg}$FOAF is expressed in the methods that are used to determine membership of a given person based on all group policies and the membership definitions. So the semantics can be entirely reduced to the description of the membership function. The membership function has two parts. Part one is the merging of identical group policies. In the second part, the membership of a person in a specific group is determined.

## 7.1 Merging of Group Policies

In a centralized group management, the definition of a group and its members is stored in a single location. In a distributed group management, this single location cannot be assumed, especially when more than one administrator is responsible for managing the group. Choosing a single location point would then result in an administrator who is owning that location and the other administrator has no or only limited access to it. To overcome these drawbacks, in $_{dg}$FOAF there is no single location where the group definition is stored and no distinction between different "classes" of administrators. In order to manage a group by multiple administrators, they have to agree on the same group policy. Thus, every administrator has to be able to maintain a copy of the group policy in his $_{dg}$FOAF profile. If such a group has exactly the same group policy as the group maintained by another administrator, both groups are considered to be one single group. That results in merging all memberships in the two groups to membership in one group. Therefore in $_{dg}$FOAF there is implemented the function $isIdentical : Group \times Group \mapsto \{true, false\}$ which returns true if both group policies are identical and false otherwise. In order to be identical, the two policies are required to have exactly the same properties. Furthermore, the properties of the two policies are required to have identical values. We distinguish three possible types of values: literals, *Person*s, and *Group*s. Values of literals and persons are required to be identical. To compare values of type *Group* the function *isIdentical* is used recursively. Please note that based on this definition, two groups can be considered identical because of their identical policies although they have different owners.

Based on the function *isIdentical*, another function is defined that assigns to each group a so called general identifier. Two groups that have identical policies are therefore assigned the same general identifier: $generalIdentifier : Group \mapsto GeneralIdentifier$. The general identifier is used in the membership function to abstract from different groups with identical policies, so that they can be processed as one single group. In the following, all definitions of groups and membership are considered to be made about groups referenced by their general identifier.

## 7.2 Membership Function

The determination of group membership of a person in a group is the core component of $_{dg}$FOAF. The methods for determining membership differ between

the two general types of groups, namely administrated groups and peer groups. Nevertheless some properties are identical for both types of groups. In order to preserve the provenance of every membership definition, the definitions are labeled with the person who issued them, i.e., in whose $_{dg}$FOAF profile they are defined. Membership in a group can be defined by a default policy. Thus, it cannot be altered without affecting the merging with other groups, c.f. Section 7.1. If default membership is granted to a person, this person is a member in every configuration of the group. Such persons cannot be banned or otherwise removed. Two types of default membership can be distinguished: *defaultMemberPerson* and *defaultMemberGroup*. A *defaultMemberPerson* can only be a reference to a person node. This person is in every case determined to be a member of the particular group. The *defaultMemberGroup* references another group B. All members of group B are then automatically members of the particular group.

*Administrated Groups* can have several administrators. Each administrator is allowed issuing specific membership definitions. Only the allowed membership definitions can affect the membership of persons in the group. The group policy defines who is allowed to issue membership definitions. This is done by explicitly declaring a group as an admin group. Members of such a group get the respective administration role. Consequently, the membership of a person in an admin group needs to be verified in order to verify if this person has got the corresponding administration right. We distinguish administration roles for the following types of membership definitions:

- Granting membership to specific persons
- Inviting specific persons to the group
- Granting membership to all members of specific groups
- Banning persons as members of the group

Corresponding to these different roles, four different types of functions are generated that assign a truth value to a tuple of person and group, e.g., $canGrantPersonMembership : Person \times Group \mapsto \{true, false\}$ that is true for all persons who have the role of granting membership to specific persons. Based on these functions, all membership definitions can be interpreted in the following way: A membership definition issued by a person that is allowed to do so is valid, otherwise it will be ignored. The valid membership definitions of a group are the basis for determining membership of a specific person in that group. Three ways of gaining membership in an administrated group exist:

- A valid membership definition exists, which grants membership to a specific person. In addition, there exists no valid membership definition, which bans the person from that group.
- A valid membership definition exists, which invites a person to the group. This person has issued a confirmation of invitation and there exists no valid membership definition, which bans the person from that group.
- A valid membership definition exists, which grants membership to members of another specified group, e.g., group B. Then the members of group B are also members in the group, if there exists no valid membership definition, which bans the person from that group.

*Peer Groups* are not administrated by members of other specific groups but directly by the members of the peer group itself. Membership in a peer group is based on membership definitions that have been issued by persons. These membership definitions are interpreted as a social network. So methods for constructing cohesive subgroups can be applied. Suitable methods for peer groups are based on the degree of each person, i.e., the number of connections of each person to other persons. There also exist methods that are based on other measures like path length. But since these measures are difficult to control from the local view point of each person, they are less suitable than the degree property used in the following.

As described in Section 6, we distinguish two types of peer groups according to their policy: K-Core-PeerGroupPolicy and Clique-PeerGroupPolicy. The K-Core Policy requires every member to have a mutual connection to at least $K$ other members. The threshold $K$ is a property of any K-Core Policy. So a social network analysis algorithm for detecting k-cores [12] can be applied. The Clique Policy requires that every member has a mutual connection to every other member. The result of the applied algorithms is a set of persons who are in the particular peer group.

## 8 $_{dg}$FOAF for an Access Control Application

In this section, we show how $_{dg}$FOAF can be embedded in an architecture that integrates authentication and authorization for restricting access to web resources. The authentication is done via a FOAF-based authentication mechanism like FOAF+SSL [9] or with OpenID [8]. The authorization step is based on a query of group membership in a $_{dg}$FOAF group. We illustrate this architecture at the example of a picture sharing service in the scenario of Section 2. The architecture is depicted in Figure 4. In this scenario, Simon wants to access resources of the picture sharing service, which are restricted to members of the group WestparkHigh99. In order to allow Simon to access the resources, authentication and authorization needs to be performed by the picture sharing service.

*Authentication* Before the system can decide whether an access request to its resources can be allowed, it needs to clarify the identity of the person. For authentication, a FOAF compatible mechanism like FOAF+SSL or OpenID is used (task A). So the person who wants to be authenticated has to prove that he is the owner of a specific FOAF file. In case of FOAF+SSL, this is done by knowledge of a secret key, which is used during HTTPS authentication. With OpenID, this is done by proving his identity to an external service. If the authorization was successful, Simon has proven that the person who wants access is the person described in Simon's FOAF profile.

*Authorization* Knowing his FOAF-based identity, the picture sharing service needs to decide whether Simon is allowed to gain access. In the scenario, access is only granted to members of the group WestparkHigh99. Therefore the system needs to determine Simon's membership in this group. Therefore in

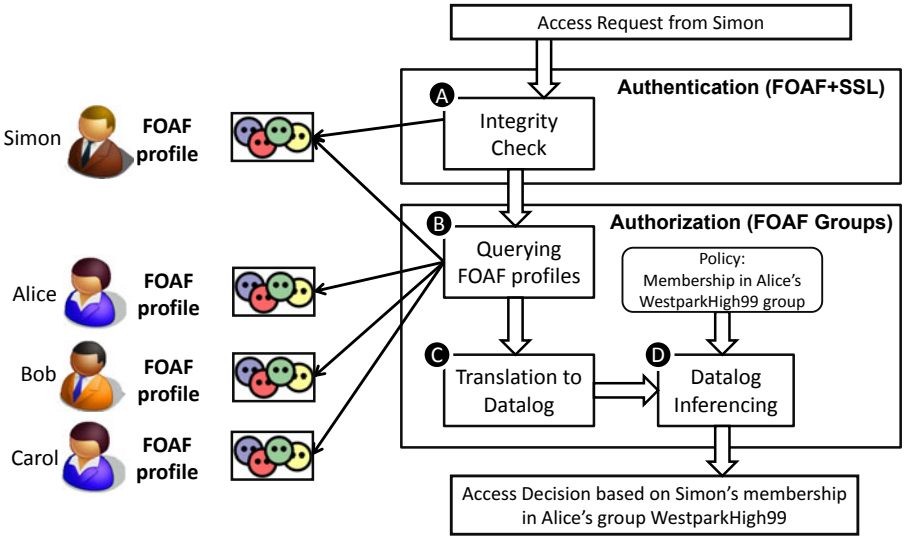



**Fig. 4.** Elements of Membership Determination

task B, all FOAF profiles that contain information about members of the group WestparkHigh99 are queried for their group definitions. In our scenario, we assume that Alice, Bob, Carol, and Simon have created their personal $_{dg}$FOAF profiles with an appropriate editor. These FOAF profiles are queried for membership determination. The task of discovering the needed FOAF profiles in order to determine a particular membership can be non-trivial and complex for other scenarios. Therefore techniques similar to proof-carrying authentication [1] may be applied, which are outside the scope of this paper. Subsequently the collected $_{dg}$FOAF group definitions are translated to a declarative language in task C. We use Datalog as it allows for a declarative representation of $_{dg}$FOAF semantics. The generated Datalog statements constitute a knowledge base used in the next task. Finally in task D, a Datalog inference engine queries the knowledge base for the membership of Simon in the group WestparkHigh99. According to the result of the query Simon is granted access to the photos uploaded to the picture sharing service.

We have implemented the described authorization step as a prototype in Java and defined the processes as Datalog rules. The relevant parts of $_{dg}$FOAF profiles are retrieved via defined SPARQL queries using Sesame[3] and then translated to Datalog facts. We use the DLV[4] interpreter as Datalog engine. For easy use of $_{dg}$FOAF within existing platforms the service of determining membership of a person could also be provided by a trusted third party. Therefore, the community platform would not need to integrate the membership determination itself.

[3] http://www.openrdf.org/
[4] http://www.dbai.tuwien.ac.at/proj/dlv/

## 9 Related Work

FOAF [3] is a well known vocabulary for describing attributes of persons and relationships between persons. A person can create one (or more) FOAF profiles to provide information about herself. Popular properties are `foaf:name`, `foaf:homepage`, and `foaf:mbox`. The `foaf:knows` property is often used to state a relationship to other persons. Groups can be defined in FOAF as instances of the type `foaf:Group`. With `foaf:membershipClass` and `foaf:member`, persons can be assigned to a group. Although it is possible to infer not explicitly given group members by a class definition of members, no formal specification is given for implications of group membership in FOAF. In the WIQA framework [2], provenance of an arbitrary statement is considered as one parameter for accessing trust in this statement. While $_{dg}$FOAF provides a specification for expressing group structures, the WIQA framework is more generic. The OpenSocial application programming interface (API)[5] is an effort led by Google to standardize an interface for profiles that are stored at social network platforms. Applications based on this API can be executed on specific social network platforms using the available data. They do not allow for exchange and modification of group specifications.

Our $_{dg}$FOAF approach provides functionality similar to what could be retrieved out of a combination of existing more general approaches for policy-based trust management frameworks (SD3 [5], RT [6], KAOS [11]). But this specific combination along with its restriction to specific policy alternatives for describing groups is the strength of $_{dg}$FOAF. SD3 [5] allows expression of policies in Datalog and uses provenance of facts that are retrieved from another location. Therefore the $_{dg}$FOAF approach would be expressible in SD3. The difference is that $_{dg}$FOAF relies on an RDF representation of data and is restricted to specific policy types, which is crucial for interoperability in the Web. In RT [6], delegation of group administration is allowed similar to $_{dg}$FOAF. However, the RT frameworks lacks for describing groups that have more than one administrator. KAOS [11] uses RDF representations for its policies as $_{dg}$FOAF does. However KAOS assumes a specified policy enforcement point, while in $_{dg}$FOAF this task can be fulfilled by any actor.

## 10 Conclusion

In this paper, we have presented $_{dg}$FOAF as an approach for a distributed group management based on FOAF. Based on a real world scenario, we have derived the requirements for such a distributed management of groups and introduced the concepts and schema of $_{dg}$FOAF. In addition, we have developed mechanisms for merging policies of distributed groups and determining group membership. We have demonstrated the use and implementation of our approach in the context of an application for access control. With $_{dg}$FOAF, we achieved an important step towards distributed management of groups that is a) independent of concrete

[5] http://www.opensocial.org/

platforms and b) supports the usage of the groups over these platforms. In our future work, we plan to further enhance the $_{dg}$FOAF approach and to provide a discovery strategy for finding all relevant $_{dg}$FOAF profiles belonging to one group.

*Acknowledgement.* This research has been co-funded by the EU in FP7 in the WeKnowIt project (215453).

## References

1. Appel, A.W., Felten, E.W.: Proof-carrying authentication. In: Proceedings of ACM CCS 1999, New York, NY, USA (1999)
2. Bizer, C., Cyganiak, R.: Quality-driven information filtering using the WIQA policy framework. Web Semantics: Science, Services and Agents on the World Wide Web 7(1), 1–10 (2009)
3. Brickley, D., Miller, L.: FOAF vocabulary specification 0.97 (2010), http://xmlns.com/foaf/spec/20100101.html
4. Hausenblas, M., Halb, W., Raimond, Y., Heath, T.: What is the Size of the Semantic Web? In: Proceedings of I-Semantics, pp. 9–16 (2008)
5. Jim, T.: SD3: A Trust Management System with Certified Evaluation. In: IEEE Symposium on Security and Privacy, Los Alamitos, CA, USA (2001)
6. Li, N., Mitchell, J.C.: RT: a Role-based Trust-management framework. In: DARPA Information Survivability Conference and Exposition (2003)
7. Preece, J.: Online Communities: Designing Usability, Supporting Sociability. John Wiley, Chichester (2000)
8. Recordon, D., Fitzpatrick, B.: Openid authentication 1.1 (May 2006), http://openid.net/specs/openid-authentication-1_1.html
9. Story, H., Harbulot, B., Jacobi, I., Jones, M.: FOAF+TLS: RESTful Authentication for the Social Web. In: SPOT 2009 Workshop at ESWC, Heraklion (2009)
10. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F.: Signing individual fragments of an RDF graph. In: WWW. ACM, New York (2005)
11. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction and Enforcement. In: POLICY 2003 Workshop, Washington, DC, USA (2003)
12. Wasserman, S., Faust, K.: Social network analysis. Cambridge University Press, Cambridge (1994)
13. Yeung, C.A., Liccardi, I., Lu, K., Seneviratne, O., Berners-Lee, T.: Decentralization: The Future of Online Social Networking. In: W3C Workshop on the Future of Social Networking Position Papers, Barcelona, January 15-16 (2009)

# Semantics, Sensors, and the Social Web: The Live Social Semantics Experiments

Martin Szomszor[1], Ciro Cattuto[2], Wouter Van den Broeck[2], Alain Barrat[2,3], and Harith Alani[4]

[1] City eHealth Research Centre, City University London, UK
[2] Complex Networks and Systems Group
Institute for Scientific Interchange (ISI) Foundation, Turin, Italy
[3] Centre de Physique Théorique (CNRS UMR 6207), Marseille, France
[4] Knowledge Media Institute, The Open University, UK

**Abstract.** The Live Social Semantics is an innovative application that encourages and guides social networking between researchers at conferences and similar events. The application integrates data from the Semantic Web, online social networks, and a face-to-face contact sensing platform. It helps researchers to find like-minded and influential researchers, to identify and meet people in their community of practice, and to capture and later retrace their real-world networking activities. The application was successfully deployed at two international conferences, attracting more than 300 users in total. This paper describes the Live Social Semantics application, with a focus on how data from Web 2.0 sources can be used to automatically generate Profiles of Interest. We evaluate and discuss the results of its two deployments, assessing the accuracy of profiles generated, the willingness to link to external social networking sites, and the feedback given through user questionnaires.

## 1 Introduction

Most conference attendees would agree that networking is a crucial component of their conference activities. It is vital for researchers to network and foster collaboration, with many ties originating in casual meetings and spontaneous conversations. Networking at conferences can be very daunting, especially to junior researchers and those who cross discipline boundaries. Furthermore, researchers often lose track of whom they met and where at such events. Such problems become even more evident in medium to large size conferences, where it is easy for individual researchers to get lost in the crowd. Unfortunately, there are few applications to help researchers to initiate, capture, and preserve their online as well as offline social interactions during conferences.

To this end, we have developed Live Social Semantics (LSS), a novel application that brings together data from the Semantic Web, Web 2.0, and a face-to-face (F2F) contacting sensing platform. LSS encourages and strengthens collaboration and communication between researchers by supporting their social networking activities during conferences, helping them find and locate like-minded individuals, people in their community of practice, and to permanently log, and retrace, their F2F contacts. LSS integrates (a) academic related data from the Semantic Web, (b) the rich social data

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 196–210, 2010.

from major social networking sites, and (c) a physical-presence awareness infrastructure based on active radio-frequency identification (RFID). This application was deployed at two major international conferences, ESWC 2009 in Crete and HyperText 2009 in Turin, where it was used by 300 researchers with promising results.

The next section describes a variety of related works, followed by a full description of the LSS architecture in Section 3. Section 4 presents our method for building and representing Profiles of Interest. Results and evaluation of LSS deployments are covered in Sections 5 and 6 respectively. Discussion and future work is given in Section 7, followed by conclusions in Section 8.

## 2 Related Work

The interplay of networking and social contact at a conference gathering was initially investigated in the context of opportunistic networking for mobile devices [12] by using wearable Bluetooth-enabled devices. Subsequent work focused on sensing organisational aspects [6] by using Bluetooth-enabled mobile phones, and on characterising some statistical properties of human mobility and contact [21,16]. These early experiments could not assess face-to-face (F2F) human contact in a large-scale setting since they mostly relied on Bluetooth communication. Wu and colleagues used what they call "sociometric badges" to investigate the impact of F2F interactions on productivity [22]. These badges used radio frequency to detect physical proximity, infrared to detect F2F body alignments, and voice sensors to detect conversations.

RFID is an increasingly popular technology for location tracking. IBM used RFIDs to track attendees of a conference in Las Vegas in 2007. The devices were used to track session and meal attendance [21]. The information they collected was limited to the name, title and institution of attendees. No social or semantic data was collected or used. Fire Eagle[1] by Yahoo! is a service that detects the geographical location of users (e.g. based on WIFI access points), and allows them to share it with their online friends.

Recently, the SocioPatterns project[2] investigated patterns of human contact at large-scale social gatherings by deploying a distributed RFID platform that is scalable and attains reliable detection of F2F interactions as a proxy of social contact [3]. The LSS application presented here leveraged that platform to mine real-time social contacts.

To the best of our knowledge, our LSS application is the first where real-world F2F contacts are mashed up in real time with semantic data from online tagging systems. The free nature of tagging generates various vocabulary problems: tags can be too personalised; made of compound words; mix plural and singular terms; they can be improper words; they can be synonymous, etc. [14,9,10]. This total lack of control obstructs analysis [13]. In our work, we follow the approach of *cleaning* existing tags using a number of term filtering processes, similar in spirit to those used in [11].

LSS constructs semantic models of interests for individuals by merging and processing their tagging activities from multiple online social networking systems (SNS). This process involves dealing with several problems, such as filtering of tags, disambiguating them, associating tags with semantics, and identifying interests. Tag ambiguity is a well

[1] `http://fireeagle.yahoo.net/`
[2] `http://www.sociopatterns.org`

recognised problem in folksonomies. Clustering has been investigated as a disambiguation approach, where tags are grouped together based on their co-occurrence, to facilitate distinction between their different meanings [4,23,17]. While such techniques have demonstrated that the underlying folksonomy structure does contain information that can enable automatic disambiguation, they are too computationally expensive and lack any semantic grounding. Angeletou and colleagues [2] used WordNet to identify ambiguous tags, and compared the WordNet senses for the tag to those of the co-occurring tags, to identify the most similar sense. In our approach, we used DBpedia[3] for disambiguating tags, automatically associating them with Semantic Web URIs. Some manually-driven approaches have been proposed for assigning URIs to tags (e.g. [17,15]). Similarly to [20], we explore a strategy for the automatic selection of URIs using DBpedia concepts [7].

## 3 Live Social Semantics Application

### 3.1 General Architecture

The system architecture of LSS is shown in Figure 1. The diagram is vertically partitioned into two spaces: the online world (i.e. data about individuals held on the web), and the physical space (i.e. RFID-based contact data). Data in the online world is sourced from the following:

- Social networking sites: Tagging and social relation data is collected from Delicious, Flickr, Facebook, and lastFM using the Extractor Daemon. This data is then used to reflect the online contact network of individuals. The tagging data is processed by the Profile Builder (centre, top of diagram) to infer their interests. The Tagora Sense Repository is responsible for associating tags to URIs from DBpedia.
- Semantic Web Linked Data: Information on publications, projects, and the Community of Practice (COP) of researchers is retrieved from RKBExplorer[4] [8] and semanticweb.org. This data is used to reflect the contact network of individuals based on their paper co-authorships and project co-memberships.

Physical space data is collected from F2F contacts between individuals which are measured using RFID readings (Section 3.2). Such data is fully integrated with the online world data in a triple store (centre right of Figure 1), where all the data is stored. This enriches the visualisation and processing of real-world social contacts with the online social contacts of those individuals. A focused *Contact* ontology[5] was used to represent real-world social interactions between individuals, recording the total contact time on a daily basis (sections 3.2 and 3.3).

### 3.2 Real-Time Social Contacts

Real-world interactions of conference attendees are mined using RFID hardware and software infrastructure developed by the SocioPatterns project [3]. Willing participants

[3] `http://dbpedia.org/`
[4] `http://www.rkbexplorer.com`
[5] `http://tagora.ecs.soton.ac.uk/schemas/LiveSocialSemantics`

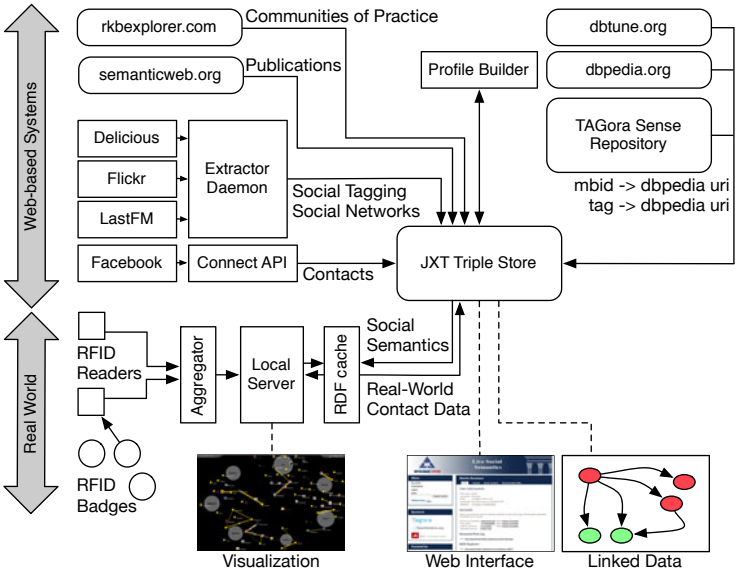



**Fig. 1.** Live Social Semantics Architecture

were issued with RFID badges to monitor their F2F contacts with others. The RFID badges engage in multi-channel bi-directional radio communication, and by exchanging low-power signals which are shielded by the human body, they can reliably assess the continued F2F proximity of two individuals. A F2F contact is recorded if users face each other for around 20 seconds or more, within a distance of around one meter.

We generate a weighted graph to represent the cumulative F2F contacts between the participants. This information is periodically uploaded to the triple store via RDF/HTTP and integrated with the other data layers.

We use the real-world and online social relations to compute simple recommendations. For example, if two attendees are in F2F contact at a given time, the server searches for, and displays, any mutual contacts from the online world data, for example people who are not present at the given time, but are nevertheless connected to the two users in one of the online social networks used by LSS (Section 3.3). Details of using RFID in LSS can be found in [5,1].

### 3.3 Visualisation

LSS has two visualisations, taken from the SocioPatterns project (detailed in [1]):

– Spatial View: This view provides an overview of the real-time contact graph. It represents the location of RFID-badge wearing participants within range of the RFID readers, as well as their ongoing social contacts. Each participant is represented by a labelled yellow disc or, when available, by their Facebook profile picture. The contacts are represented by yellow edges, whose thickness and opacity reflects the weight of the contact. The edges are decorated, where applicable, with small

Facebook, Flickr, Delicious, lastFM or COP icons, marking the occurrence of that relationship in the respective network.
– User-focus view: This view displays the social neighbourhood of a particular user. It shows all participants with whom this user has ongoing contact with (yellow edges for live contacts) or had significant (cumulative) contact with (grey edges for historical contacts). This view also attempts to *close relevant triangles*, by showing mutual contacts as explained earlier.

## 4 Semantic Profiles of Interest

Tags usually reflect the interests of their authors. Such interests could range from topics, places, events, people, hobbies, etc. We have developed a tool that processes the public tagging activities of users and automatically generates a list of DBpedia URIs to represent the interests of the taggers [18]. To generate Profiles of Interest (POI) from social tagging, we follow these steps (described in the following sections):

1. **Collect tagging information:** A user's complete tagging history is extracted from the target site using public APIs or screen scraping, and converted to RDF. We utilise our Tagging Ontology to represent all tagging events, recording the resource tagged, the tag ordering, and date of annotation.
2. **Associate Tags with Potential Concepts:** Using the TAGora Sense Repository, we associate each tag to a set of potential DBpedia URIs that represent the tag senses.
3. **Perform Tag Disambiguation:** For tags with more than one candidate sense, we perform some basic disambiguation to discover the intended meaning.
4. **Calculate Interest Weights:** For each DBpedia URI identified as a potential interest, we calculate a weight based on tag frequency and a time decay factor.
5. **Create Profile of Interest:** Generate a ranked list of interests.
6. **User Verification:** Allow users to verify and edit their POI as they see fit.

### 4.1 Collecting Tagging Data

The first step in building a POI is to collect social tagging information from various folksonomies. In previous work we used the Google Social API[6] to find and correlate several social networking accounts of given users [19]. In LSS however, users must explicitly enter their social networking accounts on the LSS website. Therefore users are given full control in deciding which of their accounts will be used and shared.

The data collection process is responsible for harvesting information from a range of social networking sites. In the case of Flickr, Facebook, and Last.fm, APIs are provided that allow us to download a complete history of user tagging activity. However, for Delicious, the API is limited and so we used custom screen-scraping scripts. All tagging information is stored by LSS in RDF. We use the TAGora tagging ontology[7] (Figure 2) which is specifically designed to represent tag assignments (posts) and tag use (frequency, time, relation to Global Tag, etc). In future work we plan to merge this ontology

[6] `http://code.google.com/apis/socialgraph/`
[7] `http://tagora.ecs.soton.ac.uk/schemas/tagging`

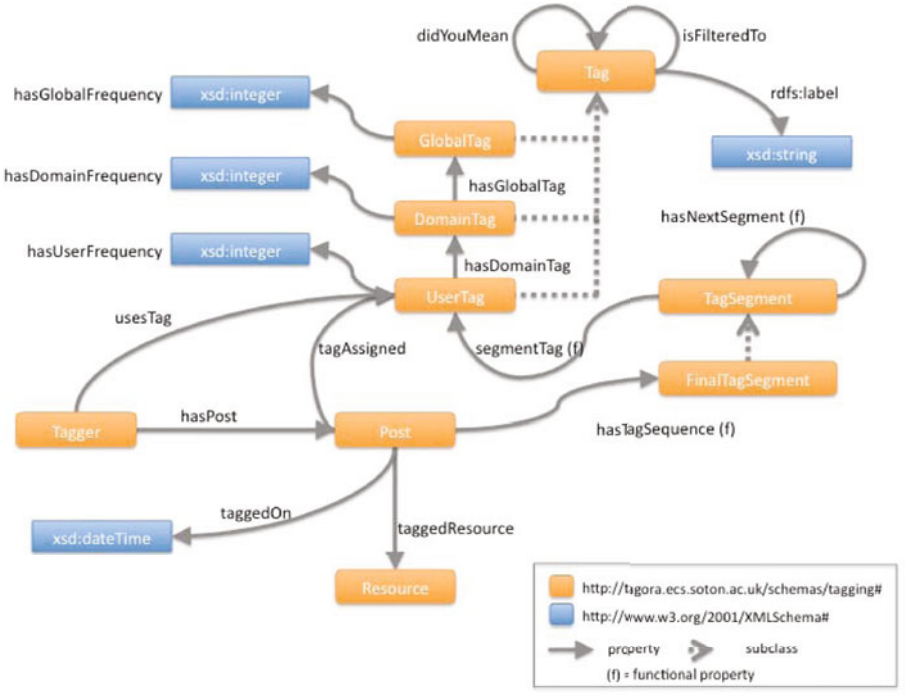



**Fig. 2.** TAGora Tagging Ontology

with SIOC,[8] by extending the latter to include our detailed tagging representations that are necessary for tag disambiguation.

### 4.2 Associate Tags with Potential Concepts

Tags can be misspelled, synonymous and come in a morphologic variety. As a result, important correlations between resources and users are sometimes lost simply because of the syntactic mismatches of the tags they used. To this end, we developed the TAGora Sense Repository[9] (TSR), a Linked Data enabled service endpoint that provides tag filtering services and extensive metadata about tags and their possible senses. When the TSR is queried with a particular tag string, by formatting a URI that contains the tag (in a REST style, e.g. `http://tagora.ecs.soton.ac.uk/tag/apple/rdf`), the tag is filtered, matched against a set of potential DBpedia and W3C Wordnet URIs that represent possible meanings for the tag. For the purposes of the LSS application, we also provide a SPARQL endpoint. In the following Sections, we briefly describe the functionality of the TSR in terms of the index we built and the search API provided.

**Creating The Resource Index.** The first stage in building the TSR was to process the XML dump of all Wikipedia pages to index all titles, mine redirection and disambiguation links, and extract term frequencies for each of the pages. For the current version

[8] `http://sioc-project.org/sioc`
[9] `http://tagora.ecs.soton.ac.uk/tsr/`

we use a dump available from http://download.wikimedia.org, created on 08/10/2008. For each Wikipedia page in the dump, we extract and index the page title, a lower case version of the title, and a concatenated version of the title (i.e. the title Second_life becomes secondlife). This multiple title indexing enables us to match more easily tags that are made up of compound terms. We also extract redirection links, disambiguation links, as well as the terms contained in the page and their frequencies. During this indexing process, we also store a list (and count) of all incoming links to each page. Since the dump is large, we only store terms with a frequency greater than the mean frequency of all terms in that page. This data is stored in a triple store using our own extended DBpedia ontology since we are providing more detailed metadata about the entries than DBpedia.org, such as the term frequencies. Each Wikipedia page in the TSR is also linked to DBpedia via the owl:sameAs property.

**Searching For Senses.** When the TSR is queried with a tag, the first step is to find a list of candidate DBpedia resources that represent possible senses of the tag. We begin by normalizing the tag string (i.e. removing non-alphanumeric characters as described in [12]). The triple store is then queried for all entries with the same lowercase title or concatenated title as the tag. During this process, we are likely to encounter redirection links and/or disambiguation links, both of which are followed. When a set of candidate senses has been created, we calculate the total number of incoming links for each resource (including the sum of incoming links for any pages that redirect to it). Finally, a weight is associated with each possible sense, calculated by dividing the number of incoming links associated with that sense by the total number of incoming links for all senses associated with the tag. This basic page rank inspired measure means senses that have very specific meanings receive much lower weights than those associated with general concepts.

For each user tag in LSS, we use a property in the Tagging ontology that links it to the Global Tag in the TSR. Figure 3 shows how a FOAF profile for an LSS user (denoted with the URI tagora:eswc2009/foaf/4) is linked to a representation of their Delicious activity (with the URI tagora:delicious/martinszomszor). The property tagging:usesTag links their FOAF URI to each of their Delicious Tag URIs (e.g. ontologymapping) that is, in turn, linked to the Global Tag URI in the TSR (tagora:tag/ontologymapping in this case). The TSR provides a link from the Global Tag to the possible DBpedia senses (via the disam:hasPossibleSense property). These links can be used to infer that an individual is potentially interested in a particular concept, in this example the tag ontologymapping is mapped to the DBpedia entry for Semantic_Integration.

### 4.3 Tag Disambiguation

The disambiguation process aims to analyse the context in which a tag has been used to identify the most likely sense among all possible senses for that tag. Tags are considered ambiguous if they are associated with multiple senses (i.e. more than 1 DBpedia resource). For such a tag, its context is captured and represented as a term vector. By context we mean the other tags that were used to annotate a given resource, hence each use of the tag can have different contexts. We construct another vector from term

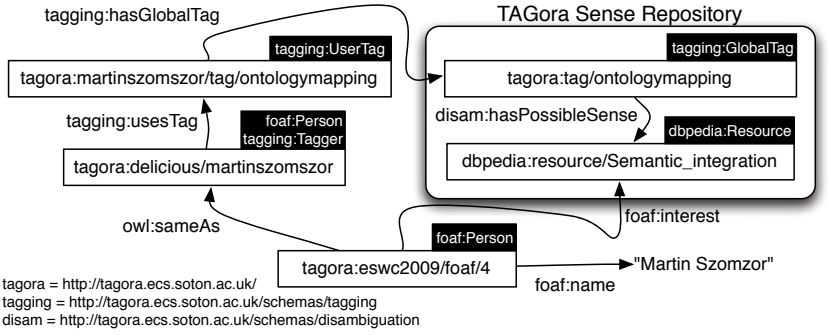



**Fig. 3.** Linking users to interests inferred from their tagging activities on social networking sites

frequencies associated with the possible DBpedia senses. We then measure cosine similarity between these vectors, and if one of the similarity scores is above a threshold (0.3 in this case), we conclude that this is the correct sense for that tag. If more than one (or zero) senses score above the threshold, we do not associate a meaning to the tag since we cannot reliably choose a correct sense. By iterating through all tags associated with a user (i.e. through Delicious or Flickr), we are able to build a candidate resource list of interests $C$. Details of our disambiguation algorithm and some initial experiments can be found in [7].

### 4.4 Calculating Interest Weights

For each interest (i.e. DBpedia resource) $r \in C$, we calculate a weight $w = f_r * u_r$, where $f_r$ is the total frequency of all tags disambiguated to sense $r$, and $u_r$ is a time decay factor. Therefore, tags that have been used more recently will receive a higher weight than those used earlier in time. If many tags of a given user are associated with the same interest, then the weight for that interest will increase accordingly. The final list of interests contains only those with a weight above the average weight for that user.

### 4.5 Creating the Profile of Interest

The lists of interests produced by the previous processes are used to generate an RDF Profile of Interest (POI) for each users using the FOAF `interest` property to link the person to the relevant Wikipedia categories. If more than 50 candidate interests have been found, we rank them by weight and suggest the top 50.

### 4.6 User Verification

Once POIs are generated, the users can browse the list of interests and edit as required, removing or adding new interests as they see fit (Figure 4). Users may wish to remove an interest for various reasons, for example if it was incorrectly identified (e.g wrong disambiguation or filtering), if it is not an interest (or a historical one), or if the user chooses not to share it with the community (private or deemed irrelevant). Users can authorise LSS to use their profiles by clicking on a *save* button.

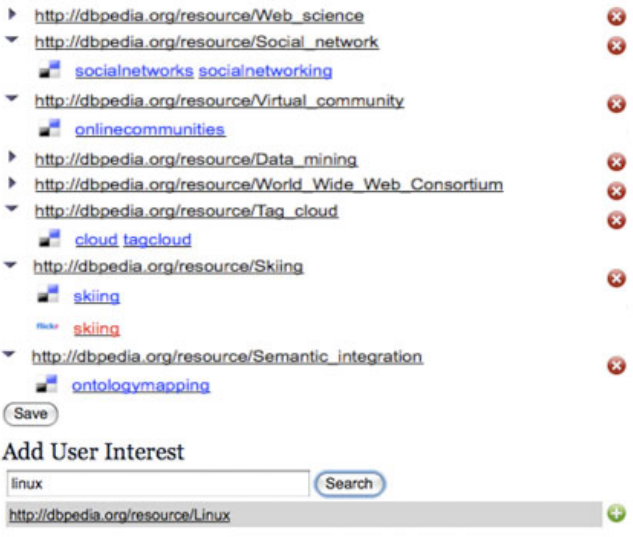



**Fig. 4.** Users can browse and edit their profiles of interest before authorising their use. The tags and their sources that led to each interest are shown. These profiles are automatically generated from users' public tagging activities.

## 5 Experiments and Results

The LSS application was deployed for a total of 7 days at the following two events:

- European Semantic Web Conference (ESWC) in Crete, 1-4 June 2009: This conference was attended by 305 people, out of which 187 participated in LSS. Out of the 187 who collected an RFID badge, 139 of them also created accounts on our application site.
- HyperText (HT), Turin, June 29-July 1, 2009: Attended by around 150 people. 113 of them collected an RFID, and 97 registered with LSS.

Each participant was issued a uniquely numbered RFID badge. Users were asked to enter their RFID ID number on the dedicated LSS website. On this website, users were also able to provide their Delicious, Flickr, and lastFM account names, as well as activating a Facebook application that collected their social contacts. The results reported below focus on user participation and SNS account declaration and POI generation. Results and statistics of RFID use can be found in [5].

**Participation results.** As mentioned above, out of a total of 455 attendees of the ESWC and HT conferences, 300 of them took part in Live Social Semantics (187 at ESWC and 113 at HT). Out of these 300 users, 236 of them created an account on the application site (139 at ESWC and 97 at HT). Hence around 21% of the users who collected an RFID badge did not register to submit any information about themselves (e.g. name, email, social network accounts). F2F contacts of such users were captured, but were not associated with any personal profile.

**Table 1.** Number of social networking accounts entered into LSS by 236 users during two field experiments

| Experiment \ Account | Facebook | Delicious | lastFM | Flickr | **Total** |
|---|---|---|---|---|---|
| **ESWC09** | 78 | 59 | 57 | 52 | **246** |
| **HT09** | 48 | 28 | 26 | 23 | **125** |
| Total | 126 | 87 | 83 | 75 | **371** |

**Table 2.** Number of users who entered 0, 1, 2, 3 or 4 social networking accounts into the Live Social Semantics site during experiments at ESWC09 and HT09 conferences

| Experiment \ Number of accounts | 0 | 1 | 2 | 3 | 4 | **Total** |
|---|---|---|---|---|---|---|
| **ESWC09 users** | 49 | 36 | 28 | 13 | 13 | **139** |
| **HT09 users** | 35 | 18 | 23 | 8 | 13 | **97** |
| Total | 84 | 54 | 51 | 21 | 26 | **236** |

**Table 3.** Table shows the number of interests generated from tags taken from Delicious, Flickr, or lastFM, and how many were removed by users. These statistics are based on 72 POIs verified and saved by their owners.

| | Global | Delicious | Flickr | lastFM |
|---|---|---|---|---|
| Concepts Generated | 2114 | 1615 | 456 | 43 |
| Concepts Removed | 449 (21%) | 307 (19%) | 133 (29%) | 9 (21%) |

**Declaration of social networking accounts.** Users were able to declare on the LSS site their accounts for Delicious, Flickr, lastFM, and Facebook. The numbers of such accounts that our 236 registered users (i.e. users with LSS accounts) declared on the LSS site are shown in Table 1. The number of social networking accounts declared on LSS site by each individual user varied from 0 (i.e. did not enter any accounts), to 4 (i.e. entered an account for each of Delicious, Flickr, lastFM, and Facebook). Table 2 shows that about 36% of our 236 registered users did not declare any social networking accounts. It also shows that around 58% of our users declared more than one social networking account.

**Semantic Profiles-of-Interest results.** We analysed 72 POIs that were verified and activated by our users (Section 4.6). Table 3 shows the total number of interests that were automatically generated, and those that were removed manually by users during both field experiments. A total of 2114 DBpedia concepts were proposed, out of which 449 were removed by users (21%). Although a facility was included on the website for users to add new interests, only 19 new concepts were added.

## 6 Evaluation

Table 3 above showed that 29% of interests suggested from Flickr tags were removed by users, in comparison to 19% and 21% for Delicious and lastFM respectively. This suggests that Delicious and lastFM are perhaps more reliable sources of user interests

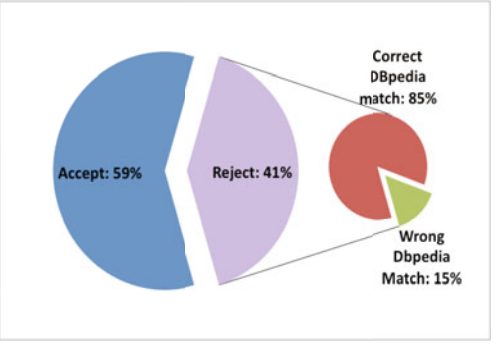



**Fig. 5.** 11 users edited their POIs in HT09. They accepted 59% of the interests that our system generated, and rejected 41%, out of which 15% were matched by our system to incorrect DBpedia URIs (6% of all suggested interests).

than Flickr. Inspection of the concepts removed shows that Flickr was likely to suggest concepts referring to years, names, or to places that users visited in the past, but are no longer interested in.

To evaluate the accuracy of our interest suggestions, we examined the interests that our users removed from their profiles during the HT09 experiment[10]. Users may choose to delete an interest because it is simply inaccurate (i.e. wrong DBpedia match), it does not reflect an actual interest (i.e. is a very general concept), or it is something they prefer to keep private. Users seem to have different perceptions of what an interest is, or which ones are worthy of sharing in this context. Some users were very conservative and only kept a few of the interests that our system generated for them, while others kept almost all their proposed interests. In future LSS implementations we intend to allow users to instantly input their rationale for removing an interest. Understanding these drivers will help us to better design and tune the POI generation process. However, for this evaluation, we will focus on finding out how many of the removed interests were based on tags that our POI process matched to irrelevant DBpedia URIs.

Although 36 of our users at HT09 activated their POIs (by saving them - Section 4.6), only 11 of them removed any interests. The other 25 users might have been totally satisfied with their original POIs, or perhaps they saved their profiles without reviewing them. To be on the safe side, in this evaluation we focus on the POIs that were clearly *scrutinised* and *corrected* by their owners. On average, those 11 HT09 users kept 63%, 57%, and 49% of the interests that the system suggested based on their Delicious, Flickr, and lastFM tagging activities. Several users removed lastFM-based interests, although those interests referred to the music bands that these users listened to the most.

Figure 5 shows the percentages of tags that these 11 users removed from their automatically generated POIs. Although these users removed 41% of their POIs, only 15% (30 tags out of 203 removed ones) of these removed tags were given the wrong

[10] Profiles of interests from ESWC 2009 were later anonymised and hence could not be included in this detailed evaluation since it requires an examination of users' original tags and tagged resources.

**Table 4.** Reasons why some users didn't enter any social network accounts to our application site

| Option | Reason | No. Users | % |
|---|---|---|---|
| a | don't have those accounts (or rarely use them) | 16 | 44% |
| b | use different networking sites | 10 | 28% |
| c | don't like to share them | 3 | 8% |
| d | didn't get a chance to share them (e.g. no computer, slow Internet) | 6 | 17% |
| e | other | 1 | 3% |
| | Total | 36 | 100 |

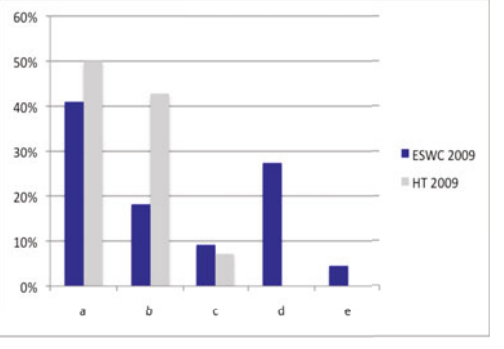



**Fig. 6.** Comparison of users' answers to our participation survey for the two experiments

semantics (i.e. matched to the wrong DBpedia URIs). For example, for most users the tag "km" was wrongly matched to the concept "Kilometre" in DBpedia, instead of "Knowledge Management". With a closer look at those 15% of tags, we find that 2% of them originated from Flickr, and the rest came from Delicious. This is hardly surprising since Delicious tags tend to be more diverse than those from Flickr. The majority of Flickr tags referred to known geographical places that have dedicated DBpedia URIs.

In addition to the above, we have also evaluated the shareability of SNS accounts by our users. As mentioned in Section 5, a total of 84 registered users did not enter any social networking accounts on the LSS site. To understand the drivers behind this, we ran a survey where we asked each of these users to pick their main reason out of the 5 options shown in Table 4. We received 36 responses to our survey so far, and the main reasons the users picked are listed in Table 4. It is clear that not having any SNS account is the most common reason for not declaring any. LinkedIn[11] and xing[12] were mentioned by several users as alternative SNS accounts, which LSS does not yet support. Although several users mentioned privacy concerns, only 8% of the users selected this as their primary reason.

Figure 6 shows a comparison of the answers to our survey (Table 4) from the ESWC and HT experiments. It is interesting to see that answer *d* was very common for ESWC attendees, who often blamed the unreliable Internet connection at the venue for their inactive participation, whereas this was not an issue for HT.

[11] `http://www.linkedin.com`
[12] `http://www.xing.com`

## 7 Discussion and Future Work

The first phase of LSS development, which led to the prototype tested at ESWC09 and HT09, was focused on architectural design and technology integration to demonstrate a novel proof-of-concept application. The second phase of development will focus on scalability, extendibility, and services.

The LSS application has so far only supported 4 currently popular SNSs: Delicious, Flickr, lastFM, and Facebook. We plan to extend LSS to allow users to submit their FOAF files, and to support other networking sites (e.g. Twitter, LinkedIn, xing). We also plan to develop an open plug-in architecture to allow external parties to develop connection to other networking systems to LSS.

The generation of profiles of interests from social tagging systems produced promising results. These POIs highlighted various general interests of users that usually cannot be inferred from their publications or project descriptions (e.g. “skiing”, “iPhone”, “sewing”, “The Beatles”). However, many users did not take the crucial step of verifying and editing these profiles. This might be due to a misunderstanding of the purpose or value for taking this step. We hope that more users will be encouraged to edit their profiles once we provide additional services that use these profiles, for example, to highlight people with similar interests.

Users who saved their profiles during both experiments removed approximately 21% of the interests the system suggested. This goes up to 41% if we exclude users who saved their profiles without any modifications. Although there could be many reasons for why users choose to remove an interest, our investigation showed that only 15% of the removed interests were totally inaccurate tag-to-concept associations (Section 6). The other 85% were proper associations, but did not necessarily represent an interest. This is a clear indication that we need to develop more sophisticated methods for determining what constitutes an interest and what does not. One promising approach is to tap into our users’ collective intelligence to improve our POI generation process, for example by filtering out the interests that most users tend to reject (“Tutorial”, “API”) or those that are too common or too general (e.g. “web 2.0”, “Semantic Web”).

Next step for interest identification will be to model users’ interests in semantic hierarchies, enabling us to represent interests at different levels of granularity. For example, if someone is interested in “Visualbasic”, “Perl”, and “C++”, then one can infer that this person is interested in “Programming languages”. The hierarchy can show how general the user interest is, so one user may use the tag “music” very often, while another might tag with “jazz” or “Hip hop”, which are more specific concepts than “music”. People tag with different levels of specificity, and this usually reflects their level of expertise in the subject [9].

Extractions of POIs has so far been limited to users’ online tagging activities. However, many of the participants have authored papers which can be used to determine their research interests, and some of these interests are already available on `semanticweb.org` in the form of paper keywords. Acquiring such interests can be added to the system and used to improve recommendations on talks or sessions to attend, or people to meet. Also, information from social networking accounts can be used to avoid recommending existing friends.

Many users expressed their interest in retrieving their data after the conferences. The next version of LSS will give users permanent access to their LSS accounts, to enable them to revisit their logs of face-to-face contacts, to modify or regenerate their POIs, and to access all the services LSS provides. This will not only enable them to access their activity log, but it will also allow them to carry their accounts across conferences where this application is deployed.

More services will be provided in future LSS deployments, such as a 'search for person', 'I want to meet', and 'find people with similar interests'. Data from RFIDs can be used to identify 'best attended session or talk'. Social contacts from social networking systems and COPs could be used to find out who has made new contacts, especially if we can compare data over several LSS deployments.

## 8 Conclusions

The Live Social Semantics application is pioneering the full integration of active RFIDs with semantics and social networking systems. The paper described and evaluated the generation of Profiles of Interests for individuals by analysing their public tagging activities on Flickr, Delicious, and lastFm. The paper reported results from deploying this application at two international conferences, ESWC 2009 and HyperText 2009, during which 300 people took part in LSS. 236 users shared 371 SNS accounts on the LSS site. A POI was generated for each of these users, and saved by 72 of them. Overall, 21% of the interests suggested by our system were removed by users. When analysing logs of 11 HT09 users who clearly edited their POIs, we found that 15% of the interests they rejected were due to incorrect semantic association. Further research is required to better understand users' rational for removing/keeping interests, and for using their collective intelligence to improve our POI generation processes.

## Acknowledgement

This research was supported by the TAGora project (Future and Emerging Technologies program of the European Commission, contract IST-34721) and WeGov (EU ICT-248512).

## References

1. Alani, H., Szomszor, M., Cattuto, C., den Broeck, W.V., Correndo, G., Barrat, A.: Live social semantics. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 698–714. Springer, Heidelberg (2009)
2. Angeletou, S., Sabou, M., Motta, E.: Semantically enriching folksonomies with flor. In: Workshop on Collective Intelligence & the Semantic Web (CISWeb), ESWC, Tenerife, Spain (2008)
3. Barrat, A., Cattuto, C., Colizza, V., Pinton, J.-F., den Broeck, W.V., Vespignani, A.: High resolution dynamical mapping of social interactions with active RFID (2008), http://arxiv.org/abs/0811.4170
4. Begelman, G., Keller, P., Smadja, F.: Automated tag clustering: Improving search and exploration in the tag space. In: Proc. 17th Int. World Wide Web Conf., Edinburgh, UK (2006)

5. den Broeck, W.V., Cattuto, C., Barrat, A., Szomszor, M., Correndo, G., Alani, H.: The live social semantics application: a platform for integrating face-to-face proximity with on-line social networking. In: Workshop on Communication, Collaboration and Social Networking in Pervasive Computing Environments (PerCol 2010), IEEE Int. Conf. on Pervasive Computing and Communications (PerCom), Mannheim, Germany (2010)
6. Eagle, N., Pentland (Sandy), A.: Reality mining: sensing complex social systems. Personal Ubiquitous Comput. 10(4), 255–268 (2006)
7. Garcá-Silva, A., Szomszor, M., Alani, H., Corcho, O.: Preliminary results in tag disambiguation using dbpedia. In: Knowledge Capture (K-Cap 2009) - Workshop on Collective Knowledge Capturing and Representation - CKCaR 2009, CA, USA (2009)
8. Glaser, H., Millard, I., Jaffri, A.: Rkbexplorer.com: A knowledge driven infrastructure for linked data providers. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 797–801. Springer, Heidelberg (2008)
9. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science 32, 198–208 (2006)
10. Guy, M., Tonkin, E.: Tidying up tags? D-Lib Magazine 12(1) (2006)
11. Hayes, C., Avesani, P., Veeramachaneni, S.: An analysis of the use of tags in a log recommender system. In: Int. Joint Conf. Artificial Intelligence (IJCAI), Hyderabad, India (2007)
12. Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., Diot, C.: Pocket switched networks and human mobility in conference environments. In: WDTN 2005: Proc. 2005 ACM SIGCOMM workshop on Delay-tolerant networking. ACM, New York (2005)
13. Li, X., Guo, L., Zhao, Y.E.: Tag-based social interest discovery. In: Proc. 19th Int. World Wide Web Conf. (WWW), Beijing, China (2008)
14. Mathes, A.: Folksonomies - cooperative classification and communication through shared metadata. In: Computer Mediated Communication - LIS590CMC (December 2004)
15. Passant, A., Laublet, P.: Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. In: Workshop on Linked Data on the Web (LDOW), Int. Word Wide Web Conference, Beijing, China (2008)
16. Scherrer, A., Borgnat, P., Fleury, E., Guillaume, J.-L., Robardet, C.: Description and simulation of dynamic mobility networks. Comput. Netw. 52(15), 2842–2858 (2008)
17. Specia, L., Motta, E.: Integrating folksonomies with the semantic web. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 624–639. Springer, Heidelberg (2007)
18. Szomszor, M., Alani, H., Cantador, I., O'Hara, K., Shadbolt, N.: Semantic modelling of user interests based on cross-folksonomy analysis. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 632–648. Springer, Heidelberg (2008)
19. Szomszor, M., Cantador, I., Alani, H.: Correlating user profiles from multiple folksonomies. In: Proc. Int. Conf. Hypertext (HT 2008), Pittsburgh, PA, USA (2008)
20. Tesconi, M., Ronzano, F., Marchetti, A., Minutoli, S.: Semantify del.icio.us: automatically turn your tags into senses. In: Social Data on the Web, Workshop at the 7th ISWC (2008)
21. Thibodeau, P.: IBM uses RFID to track conference attendees (2007), `http://pcworld.about.com/od/businesscenter/IBM-uses-RFID-to-track-confere.htm`
22. Wu, L., Waber, B., Aral, S., Brynjolfsson, E., Pentland, S.: Mining face-to-face interaction networks using sociometric badges: Evidence predicting productivity in it configuration. In: The 2008 Winter Conference on Business Intelligence, University of Utah (2008)
23. Yeung, C.-M.A., Gibbins, N., Shadbolt, N.: Tag meaning disambiguation through analysis of tripartite structure of folksonomies. In: 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology Workshops, pp. 3–6 (2007)

# LESS - Template-Based Syndication and Presentation of Linked Data

Sören Auer[1], Raphael Doehring[2], and Sebastian Dietzold[1]

[1] Universität Leipzig, Institut für Informatik,
Postfach 100920, D-04009 Leipzig, Germany
{auer,dietzold}@informatik.uni-leipzig.de
http://aksw.org
[2] Netresearch GmbH & Co. KG,
Nonnenstrasse 11d, D-04229 Leipzig
raphael.doehring@netresearch.de
http://netresearch.de

**Abstract.** Recently, the publishing of structured, semantic information as linked data has gained quite some momentum. For ordinary users on the Internet, however, this information is not yet very visible and (re-) usable. With LESS we present an *end-to-end approach* for the syndication and use of linked data based on the definition of templates for linked data resources and SPARQL query results. Such syndication templates are edited, published and shared by using a collaborative Web platform. Templates for common types of entities can then be combined with specific, linked data resources or SPARQL query results and integrated into a wide range of applications, such as personal homepages, blogs/wikis, mobile widgets etc. In order to *improve reliability and performance* of linked data, LESS caches versions either for a certain time span or for the case of inaccessibility of the original source. LESS supports the integration of information from various sources as well as any text-based output formats. This allows not only to generate HTML, but also diagrams, RSS feeds or even complete data mashups without any programming involved.

## 1 Introduction

Recently, the publishing of structured, semantic information as linked data has gained much momentum. A large number of linked data providers meanwhile publishes more than 200 interlinked datasets amounting to 13 billion facts[1]. Despite this initial success, there are a number of significant obstacles, which hinder the large-scale deployment and use of the linked data web. These obstacles are primarily related to the *quality and coherence* of linked data as well as to providing *direct benefits to end users*. In particular for ordinary users of the Internet, linked data is not yet sufficiently visible and (re-) usable.

[1] http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 211–224, 2010.

With the template-based syndication and presentation approach LESS presented in this article, we target primarily the *usability* aspect of linked data for end users. Another important problem of the linked data web tackled by LESS are quality issues, in particular with regard to *performance* and *reliability* of linked data endpoints, the importance of which has, for example, been noted earlier in [4].

LESS represents an end-to-end approach for the syndication and use of linked data based on the definition of templates for the visual presentation of linked data resources and SPARQL query results. LESS allows to edit and publish syndication templates and to share them by using a collaborative Web platform. Templates for common types of entities can then be combined with specific, linked data resources or SPARQL query results and integrated into a wide range of applications, such as personal homepages, blogs/wikis, mobile widgets etc.

As a result, a blogger writing about a recent trip to Berlin can easily integrate a nicely formatted fact box with important information about Berlin obtained from Wikipedia into her blog post. A community of science fiction fans can integrate lists on a recent BBC programming matching their preferences into their community portal. Wikipedia authors can use LESS to generate pages with lists from DBpedia [6] content. Citizen scientists interested in earthquakes can display recent earth crust activity in their region on a map without having to do any programming.

LESS supports the integration of information from various sources as well as any text-based output formats. This allows not only to generate HTML, but also diagrams, RSS feeds or even complete data mashups without any programming involved. In order to improve reliability and performance of linked data, LESS caches versions of the retrieved linked data resource descriptions or SPARQL query results either for a certain time span (thus improving the performance) or for the case of inaccessibility of the original source (thus improving reliability).

LESS represents an *end-to-end solution* by not only focusing on a single aspect, but tackling the whole life-cycle from template definition, processing, integration, authoring to sharing in a coherent way. Concrete usage scenarios tackled by LESS include for example:

- the flexible visualization of linked data resources,
- the creation of views on the linked data web,
- the integration of linked data into existing applications such as Content Management Systems, Wikis, Weblogs etc.
- the integration and compilation of information from various sources,
- the end-user-driven generation of linked data mashups.

The paper is structured as follows: We present the high-level LESS concept and architecture in Section 2. We report about our LESS implementation in Section 3. We present a number of complementary use cases for LESS in Section 4. We review related work in Section 5 and conclude with an outlook on future work in Section 6.

## 2 Concept and Architecture

From the usage scenarios mentioned in the introduction (and described in more detail in Section 4) we derived the following requirements:

- support for various data access paradigms, in particular direct linked data URI requests and SPARQL queries,
- simple template language, which is, on the one hand, easy to learn and, on the other hand, flexible enough to accommodate a wide range of usage scenarios (in particular the ones described in the introduction),
- mitigation of the current reliability and performance problems of linked data endpoints,
- facile integration of the processed templates into various Web applications, e.g. via a REST API,
- enable the collaborative authoring, sharing and re-use of templates.

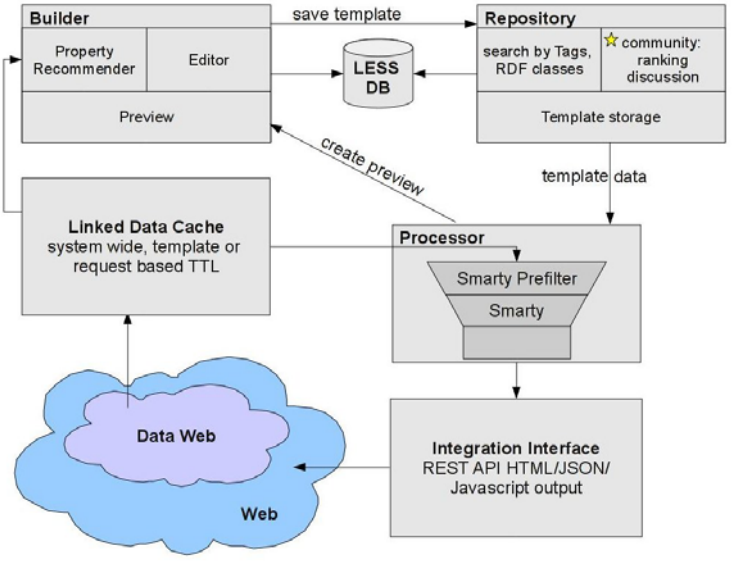



**Fig. 1.** LESS system architecture

In order to fulfill these requirements, we developed the LESS architecture as presented in Figure 1. The main LESS components are:

- *template language:* defines a declarative language for creating textual output form RDF resources and SPARQL results,
- *template builder:* allows users to define and edit LESS templates in a collaborative manner,
- *repository:* stores template revisions and metadata, allows to retrieve templates based on various annotations,

– *processor:* combines a template with concrete data retrieved from the data web,
– *linked data cache:* stores retrieved linked data resources and SPARQL queries for reuse.

In the following subsections we describe these components in more depth.

### 2.1 Template Language

A core part of LESS is the LESS Template Language (LeTL), which allows to define arbitrary text-based output representations. In general, the use of various template languages (such as Fresnel [8] or Tal4RDF [2]) is possible with LESS. However, in order to lower the entrance barrier for end users, we decided to specify a template language tailored for simplicity, convenience and versatility. LeTL is based on the popular Smarty[2] template language, but uses some custom extensions. From Smarty LeTL inherits custom functions, variable modifiers, loops, conditional parts based on the evaluation of complex expressions, cache handling, a plugin architecture and many other features. However, LeTL additionally comprises functionality for direct interaction with RDF data and SPARQL query results.

A LeTL template is applied to each individual resource described in an RDF document or every row in a SPARQL result set (unless the template application is restricted to resources of a certain class). The properties (or columns) of the resource (or SPARQL result row) are made available for reference within the LeTL template by using the syntax showcased in Table 1. The LeTL syntax allows to iterate through multiple property values and to reference recursively (sub-)templates, which are applied to dereferenced object property values. An example of such a recursive template, which iterates through a list of `foaf:Person` resources, dereferences these resources and calls another template for each one of them, is shown in Figure 2.

### 2.2 Template Builder

The template builder (depicted in Figure 3) enables end users to create LESS templates. The template builder comprises the template editor, a property recommender and a template rendering preview. The template editor supports syntax highlighting for HTML. The property recommender suggests properties based on a given, example linked data resource or SPARQL query. Once a certain property has been selected, the corresponding Smarty code to display the value of the property is added to the current cursor position in the template editor. Once a stable version of a certain template has been created by using the template builder, additional metadata (such as a template name and tags) can be attached to the template and a revision can be stored in the template repository.

[2] `http://www.smarty.net/`

**Table 1.** LeTL syntax extensions to the Smarty template language

| Syntax | Description | Example |
|---|---|---|
| `{{property}}` | Refers to the value of the property 'property' | `{{foaf:name}}` |
| `{{p@lang}}` | Refers to the value with language tag 'lang' of the property 'p' | `{{rdfs:comment@en}}` |
| `{{p^^dtype}}` | Refers to the value with datatype 'dtype' of the property 'p' | `{{dbp:birth^^xsd:date}}` |
| `{{p1->p2}}` | Refers to the value of the property 'p2' of the resources referred to by the property 'p1' | `{{foaf:currentProject ->rdfs:label}}` |
| `{template id="id" uri="uri" instances="" sortBy=""}` | Processes the template with id 'id' and resource 'uri' (using the optional attributes 'instances' iteration can be restricted and sorted with 'sortBy') | `{template id="5" uri="{$var}"}` |
| `{template id="id" sparql="query" endpoint="srv"}` | Processes the template with id 'id' and SPARQL query 'query' from endpoint 'srv' | `{template id="3" sparql="SELECT * WHERE {}" endpoint="http://dbpedia.org"}` |

### 2.3 Template Repository

The template repository allows to publish templates, to browse existing templates based on their supported RDF classes and user-defined tags as well as to rate, comment and reuse existing templates. The template repository is implemented on top of a relational database, but made available as RDF by using Triplify [1] at `http://less.aksw.org/triplify`. For each template, a unique template id is assigned. As soon as a template is changed, its revision id is incremented. Each registered LESS user can only change her own templates in order to prevent conflicts. However, a user can create her own copy of any of the templates stored in the template repository. The public availability of templates in the repository has a number of advantages: templates serve as examples for new users, they can be used by other third-party applications and the reuse of templates facilitates a natural modularization.

### 2.4 Template Processor

The template processor is the actual LESS execution environment. It takes a LESS template and a linked data resource or SPARQL query and renders the respective output. By default the LESS template processor iterates through all the resource descriptions or SPARQL query result items and applies the defined template to each of them. This allows to apply LESS also to RDF documents, which contain more than one resource description. However, in certain cases, the template application should be restricted to resources of a certain type. For these cases, the LESS template can be associated with a certain RDF class. This can be, for example, the class `foaf:Person` for a FOAF profile, thus preventing the application of the template for a person's projects described in the same FOAF

```
<h2>{{foaf:name}}</h2>
{{dc:description}} <br />
<a href="{{foaf:homepage}}">web</a>
<div id="members">
 {foreach {{foaf:member}} as $var}
  {template id="5" uri="{$var}"
    parameter_language="en"}
 {/foreach}
</div>
```

```
<div id="foaf-card-block">
 {if {{foaf:depiction}} !== ''}
  <img class='photo'
    src="{{foaf:depiction}}" />
 {/if}
 <div id="name">{{foaf:name}}</div>
 ...
 {if {{foaf:phone}} !== ''}
  <div id="tel">{{foaf:phone}}</div>
 {/if}
 <div id="email">
  <a href="{{foaf:mbox}}">
   <img src="email.png" />
   {if $language === 'en'}
     email {else} E-Mail
   {/if}
  </a>
 </div>
</div>
```

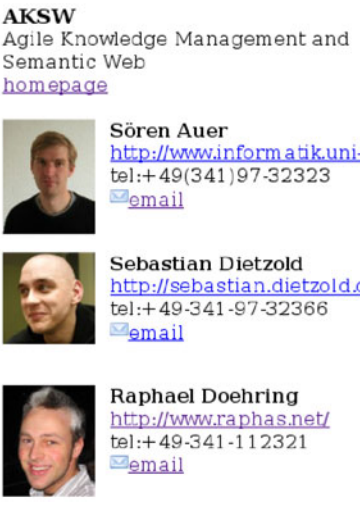



**Fig. 2.** Left: LESS templates for displaying a group profile including information about group members, obtained from separate FOAF profiles; Right: rendered output

file. When there is more than one resource of the type selected, the `sortBy` parameter can be used to order the resources based on the values of a certain property. The template processing can be additionally influenced by user-defined parameters, which can be accessed from within the template definition. Based on a system, template or request configuration, the template processor can be instructed to use a locally cached version of the linked data resource or SPARQL query. This particularly facilitates situations, where linked data endpoints are temporarily not available or respond slowly.

### 2.5 Integration Interface

In order to integrate the output of the template processor into external applications, LESS provides an REST API via the URL `http://less.aksw.org/build`. Required URL parameters are the `id` of the template, the `revision` of the template (can be omitted for accessing the last revision) and either `uri` or `sparql` for defining the linked data resource or the SPARQL query to be used. By default this REST function call returns the rendered output as text (in most cases HTML snippets). By using the optional URL parameter `output`, Javascript or JSON can be alternatively selected as output format. A further optional URL parameter is `ttl`, which specifies the time-to-live of a previously cached version

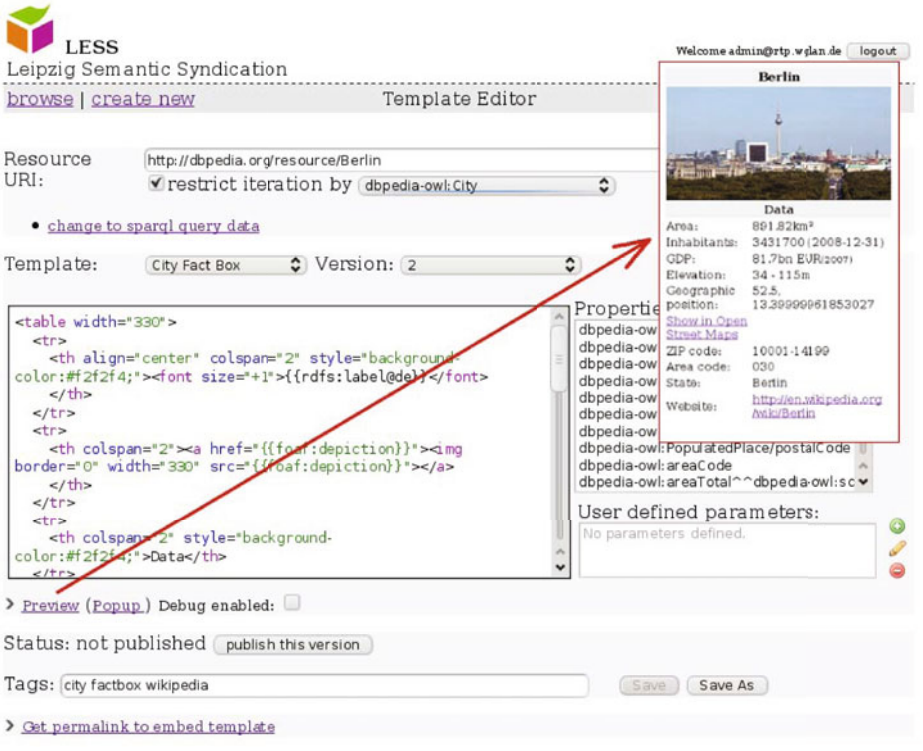



**Fig. 3.** Template Builder comprising editor, property recommender and rendering preview

of the linked data resource or SPARQL query. In case this parameter is missing, template- or system-based defaults are used. An example of a REST request (whose result is depicted in Figure 5) would look as follows:

```
http://less.aksw.org/build?id=2&
                    uri=http%3A%2F%2Fdbpedia.org%2Fresource%2FBerlin
```

## 3 Implementation

The LESS implementation was performed in PHP by using the Zend Framework[3] and the Erfurt framework for the development of semantic web applications [3]. The implementation follows the generic MVC architecture model and the convention-over-configuration paradigm, which assumes reasonable defaults for all possible configuration parameters. A demonstration platform with the LESS implementation is available at: `http://less.aksw.org`

A UML sequence diagram illustrating the template processing is displayed in Figure 4. After a template instance is requested by a user via the integration

[3] `http://framework.zend.com`

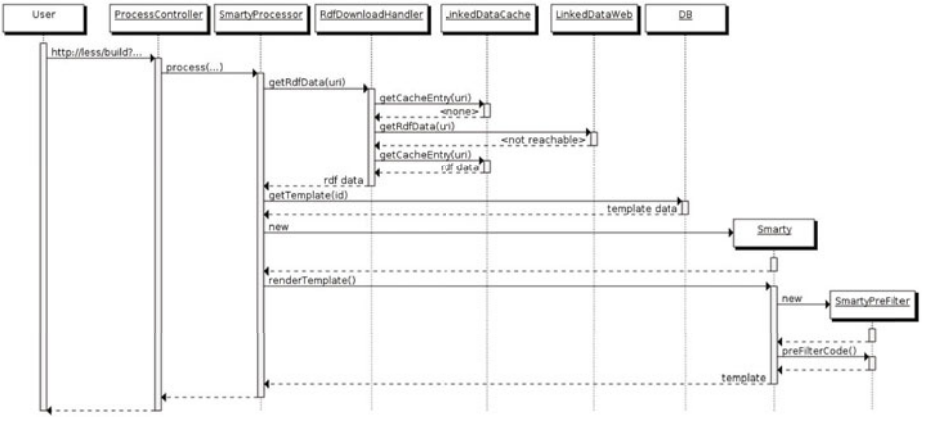



**Fig. 4.** Template processing UML sequence diagram

interface, the template processor checks the local cache or retrieves the data. Based on the template id, the template data and metadata are obtained from the relational database. The LeTL template is then compiled by the `SmartyPreFilter` into a valid Smarty template and compiled by the Smarty processor into PHP code. The Smarty processor also takes care of caching-compiled templates in order to increase the performance.

## 4 Usage Scenarios

In this section we present examples for employing LESS for the various usage scenarios identified in the introduction. The examples presented are available in the LESS template repository with tag ESWC[4].

### 4.1 Flexible Resource Visualization

Although there are some approaches for rendering RDF (e.g. [8,2]), it is currently quite cumbersome to visualize RDF and linked data in user-defined ways. LESS provides a very flexible and easy-to-learn mechanism for visualizing linked data resources and SPARQL query results, since end users can directly create output fragments with placeholders for RDF data. Figure 5 demonstrates how a LESS template can be used to visualize a linked data resource and to integrate the resulting output into a Wordpress blog. By annotating the generated visual representations with RDFa, the resource descriptions can be easily obtained and reused not only from the original data source, but also from the syndicated content representations.

[4] `http://less.aksw.org/browse?tags=%2Beswc`

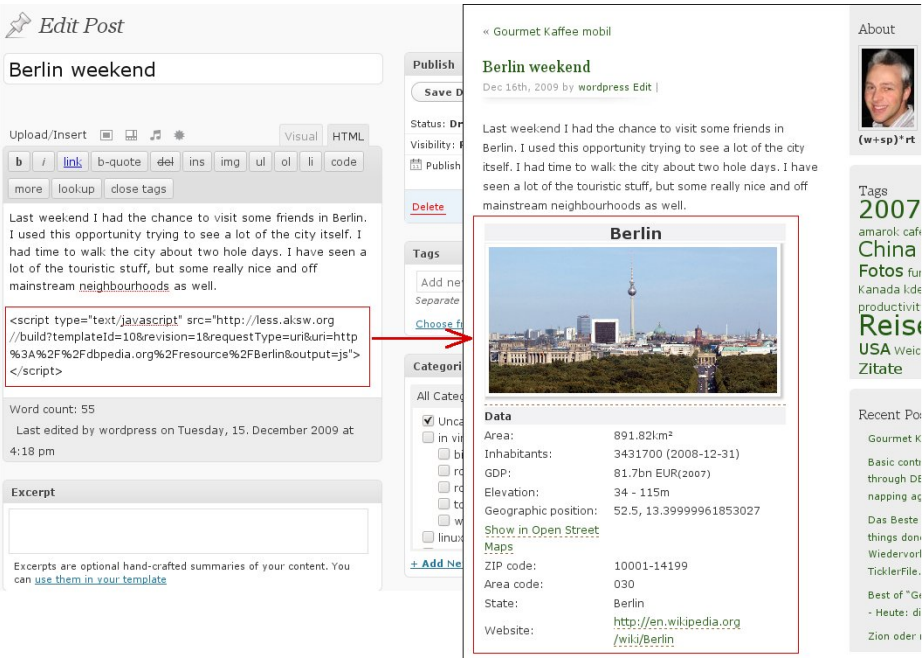



**Fig. 5.** Integration of a LESS template visualizing data obtained from DBpedia into a Wordpress blog post by using a Javascript snippet

### 4.2 Linked Data View Creation and Visualization

In many cases not only a single linked data resource, but information from various interlinked linked data resources or SPARQL queries should be displayed. Figure 6 showcases a rendered template, which obtains information about recent BBC programmings related to the series 'Mountain'. The information is obtained by querying a SPARQL endpoint containing the BBC program information and rendering the query result by employing a LESS template.

### 4.3 Integration of Information from Various Sources

LESS is not limited to create visual representations of single RDF or SPARQL sources. The possibility to dereference linked data resources or call (sub-) templates from within a LESS template (as described in Table 1) allows to integrate information from various sources into a coherent visual representation. In Figure 2 we illustrate this usage scenario with a template, which iterates through members of a group obtained from a group FOAF file and processes an individual FOAF template for each member with data from their personal FOAF profiles.

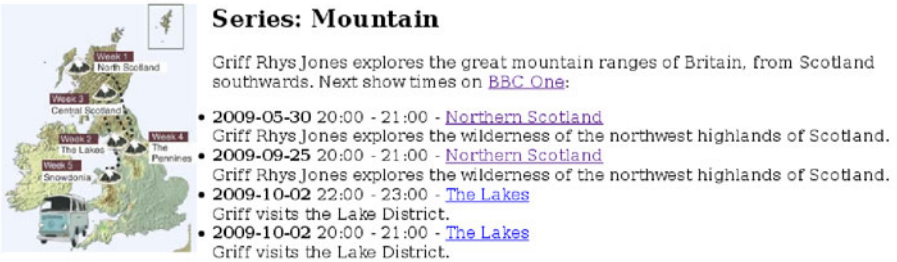



**Fig. 6.** Rendered LESS template representing a view on recent BBC programmings

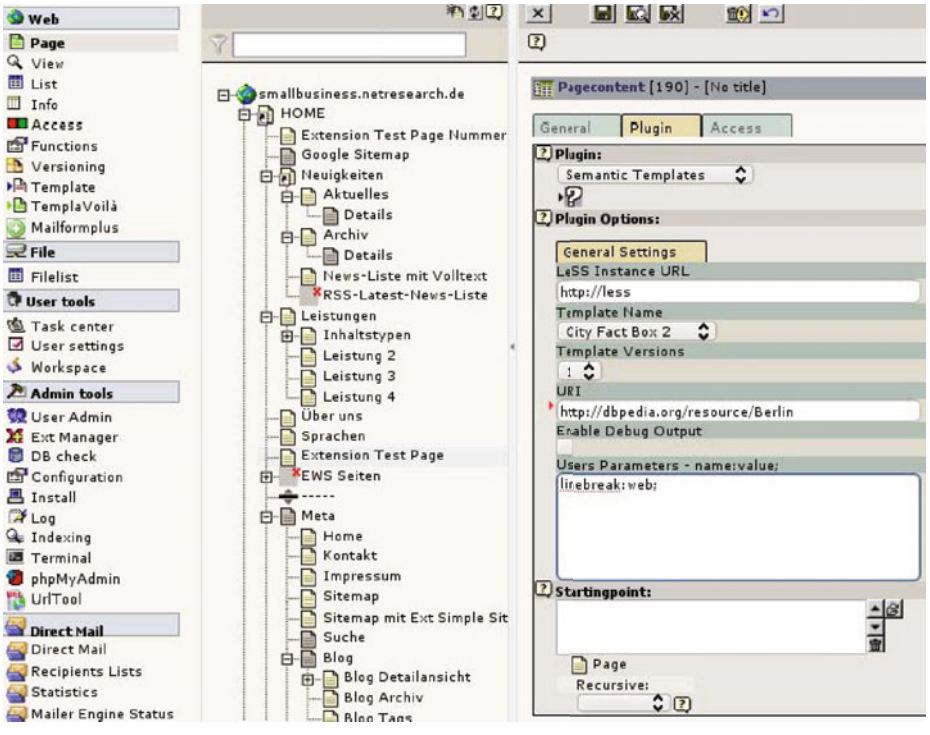



**Fig. 7.** LESS integration into Typo3: After specifying the LESS instance to use, the user is provided with a list of possible templates. The user is then able to specify the necessary template parameters through a user interface rather than by manually creating the REST URI. This simplifies integrating a LESS template into Typo3 even further.

### 4.4 Template Integration into Existing Applications

With LESS' REST style integration interface it is very easy to integrate rendered templates into existing Web applications, be it Weblogs, CMS, Wikis, traditional Web pages or any other Web application. LESS offers the template integration into existing application via employing HTML snippets, Javascripts, JSON or


```
<img src="http://chart.apis.google.com/chart?chs=450x250&chd=t:
{$data_de}&cht=lc&chds=500000,1500000&chtt=Number+of+live+births
+per+year+in+Germany+since+1950&chxt=x,y&chxl=0:|1950|1965|1980|
1995|2007|1:|500'000|1Mio|1.5Mio">
```

**Fig. 8.** Mashup template combining the Google Charts API with Eurostats data. The code in the lower part shows the image link template used to access the API.

IFRAMEs. The integration of linked data content can be even further simplified, if Web applications offer to integrate LESS templates directly, i.e. without the detour to the LESS homepage. In order to showcase how such a LESS integration can be performed, we developed a Typo3 extension, which allows to define Typo3 page content objects based on LESS templates directly from within the Typo3 user interface (cf. Figure 7).

### 4.5 Mashups

LESS is not limited to employing RDF and solely creating HTML output. Due to the flexible template language, arbitrary text-based output can also be generated, including Javascript, various XML formats (such as RSS), CSV etc. In particular, LESS enables the combination of RDF data with WebAPIs in order to create data mashups. Figure 8 shows a LESS template processing data from Eurostats and combining it with the Google Charts API in order to create a chart with births per year over time. The Google Charts API is accessed by creating a special URL, which returns the generated chart as an image. In the example the URL is constructed by means of a LESS template. Using LESS, users do not have to perform any programing in order to construct the URL for accessing the API. Likewise, the LESS output (i.e. the diagram in this example) dynamically changes as the underlying data changes.

## 5 Related Work

Related work dealing with the visual presentation and (re-)use of RDF data can be divided into two main groups. On the one hand, there are interactive,

user-interface-centric approaches, which enable the user to select and combine data interactively. The entrance barrier of these approaches is relatively low, since user interfaces are intuitive and easy to understand. On the other hand, there are programmatic-oriented approaches that specify a transformation or template language for RDF data. Although more effort is necessary to get acquainted with them, the latter provide more versatility and flexibility.

Sig.ma[5] is part of the first group and offers a fixed, table-oriented presentation of linked data. The user is enabled to select a certain data source, to edit the ordering of data and to integrate the result into an existing web page. Apart from the order the user does not have any influence on the layout of the information.

Marbles[6] is also part of the user-interface-centric group of approaches. Marbles is a web application that allows to aggregate and present RDF data from different sources by using Fresnel *lenses* and *formats* (see also below). Its focus lies on the aggregation and selection of new sources for an RDF resource as well as the selection of data. Marbles is limited to the presentation of data for XHTML clients through Fresnel-based views. There are only three views currently in use. The produced format is limited to the views.

With regard to the first group of related user-interface-centric approaches there is also a large body of work into mashups. This includes Yahoo! Pipes[7] or the (now discontinued) Google Mashup editor. Semantic Pipes [7] is a mashup technology which can make use of Linked Data. However, its focus lies more on the choreography of processing information from different sources, than on provisioning of output for direct integration into Web pages.

With regard to the second group of programmatic-oriented approaches, Fresnel [8] offers a very generic way to solve the problem of presentation and transformation of RDF data. Fresnel is a vocabulary for describing the basic concepts of RDF data presentation. It offers two main components: *lenses* and *formats*. While *lenses* are a filter to select the data to use, *formats* describe the way a set of data is presented. The result of a Fresnel transformation is not a final presentation of the data but a tree structure of data annotated with presentation-related information. The actual interpretation of this structure is delegated to the browser software using Fresnel. In addition to Marbles, other RDF browsers implementing Fresnel are IsaViz[8] (W3C/INRIA) and Longwell[9] / Piggy Bank [5]. Although very related, Fresnel is quite different from LESS, since it defines its own mechanisms for data selection (LESS uses simple projections and SPARQL), its formatting approach is more difficult to learn (due to the RDF representation of formatters) and it is less aligned with the linked data paradigm.

Tal4RDF [2] is a programming library written in Python based on Zope's Template Attribute Language. The goal of this approach is to create a light-weight template language for RDF data. The library allows to transform RDF

[5] http://sig.ma/
[6] http://marbles.sourceforge.net/
[7] http://pipes.yahoo.com
[8] http://www.w3.org/2001/11/IsaViz/
[9] http://simile.mit.edu/longwell/

data into any text-based format. Similar to LESS, a web service to render the templates is provided. However, Tal4RDF does not offer any means to publish and share templates, there is no support for SPARQL and dynamic linked data dereferencing.

The Visualization Providers for Ontology ElemenTs (VPOET) is a tool developed for the presentation of ontology elements [10]. It is based on the Fortunata library [9]. The user is enabled to create a template presenting RDF data, and a web service for rendering the templates is provided. Different to LESS, VPOET is limited to the presentation of resources belonging to a single ontology. VPOET's template language appears to be slightly cumbersome (e.g. with regard to the data referencing), as it does not support loops for iterating through resources, and the conditional evaluation of template parts cannot be based on complex conditions such as the comparisons.

## 6 Conclusions and Future Work

With LESS we aimed at contributing to mitigate the largest obstacles for a large-scale deployment of the linked data web: the demonstration of direct benefits to end users and improving the reliability and performance of linked data endpoints. LESS represents an end-to-end solution by not only focusing on a single aspect, but tackling template definition, processing, authoring and sharing in a coherent and pragmatic way. As a result, end users are empowered to make use of linked data without the need to program or gain deep understanding of the technologies involved. Despite its simplicity, LESS is very flexible and versatile. It supports a wide-range of application scenarios ranging from simple resource visualization to complex data mashup generation.

We see LESS as a first step towards bringing the data web closer to potential end users. As a continuing effort, we aim, in particular, to tackle the following enhancements and improvements in future work:

- usability improvements, such as a visual template designer supporting drag-and-drop of template elements and the integration of OntoWiki's SPARQL query builder for a more user-friendly construction of complex queries,
- development of plugins for standard Web applications, which enable users to define and integrate LESS templates from a single environment,
- better integration with data web services and search indexes such as Sindice,
- template language extensions, such as a template-in-template mechanism, support for property groups representing common information and support for different template languages such as Fresnel and Tal4RDF.

## Acknowledgments

We would like to thank our colleague Christian Weiske for the helpful comments and inspiring discussions during the development of LESS. This work was supported by a grant from the German Federal Ministry of Education and Research (BMBF), provided for the project LE4SW.

## References

1. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: light-weight linked data publication from relational databases. In: 18th International Conference on World Wide Web, WWW 2009, pp. 621–630. ACM, New York (2009)
2. Champin, P.-A.: T4R: Lightweight presentation for the Semantic Web. In: Bizer, G.A.G.C., Auer, S. (eds.) Scripting for the Semantic Web, workshop at ESWC 2009 (June 2009)
3. Heino, N., Dietzold, S., Martin, M., Auer, S.: Developing semantic web applications with the ontowiki framework. In: Networked Knowledge - Networked Media. SCI, vol. 221, pp. 61–77. Springer, Heidelberg (2009)
4. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL class descriptions on very large knowledge bases. International Journal on Semantic Web and Information Systems (2009)
5. Huynh, D., Mazzocchi, S., Karger, D.: Piggy Bank: Experience the Semantic Web inside your web browser. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 413–430. Springer, Heidelberg (2005)
6. Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Journal of Web Semantics (2009)
7. Le Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: Quemada, J., León, G., Maarek, Y.S., Nejdl, W. (eds.) WWW, pp. 581–590. ACM, New York (2009)
8. Pietriga, E., Bizer, C., Karger, D., Lee, R.: Fresnel: A browser-independent presentation vocabulary for RDF. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 158–171. Springer, Heidelberg (2006)
9. Rico, M., Camacho, D., Corcho, Ó.: A contribution-based framework for the creation of semantically-enabled web applications. Information Sciences (2009)
10. Rico, M., Camacho, D., Corcho, Ó.: VPOET Templates to Handle the Presentation of Semantic Data Sources in Wikis. In: Proceedings of the Fourth Workshop on Semantic Wikis The Semantic Wiki Web, Hersonissos, Crete, Greece, pp. 186–190 (2009)

# Hierarchical Link Analysis for Ranking Web Data

Renaud Delbru[1], Nickolai Toupikov[1], Michele Catasta[2,⋆], Giovanni Tummarello[1,3], and Stefan Decker[1]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland
firstname.lastname@deri.org
[2] School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
firstname.lastname@epfl.ch
[3] Fondazione Bruno Kessler
Trento, Italy
lastname@fbk.eu

**Abstract.** On the Web of Data, entities are often interconnected in a way similar to web documents. Previous works have shown how PageRank can be adapted to achieve entity ranking. In this paper, we propose to exploit locality on the Web of Data by taking a layered approach, similar to hierarchical PageRank approaches. We provide justifications for a two-layer model of the Web of Data, and introduce DING (Dataset Ranking) a novel ranking methodology based on this two-layer model. DING uses links between datasets to compute dataset ranks and combines the resulting values with semantic-dependent entity ranking strategies. We quantify the effectiveness of the approach with other link-based algorithms on large datasets coming from the Sindice search engine. The evaluation which includes a user study indicates that the resulting rank is better than the other approaches. Also, the resulting algorithm is shown to have desirable computational properties such as parallelisation.

## 1 Introduction

A growing number of data management scenarios have to deal with heterogeneous and inter-linked data sources. On the Web, more and more structured and semi-structured data sources are becoming available: millions of databases supporting simple web applications or more advanced Web 2.0 services (e.g. Wordpress, Facebook), hundreds of millions of documents embedding semi-structured data (e.g. HTML tables, Microformats such as Last.fm, RDFa such as BestBuy, etc.), and recently the rapidly growing amount of online Semantic Web

⋆ The author contributed to this work while he was a master student in DERI.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 225–239, 2010.

data repositories (e.g. the Linked Data[1] initiative). In enterprises, integration of structured data (databases) and semi-structured data (XML, documents, etc.) is a common scenario. There is a growing demand for the exploitation of these data sources, and therefore a need for searching and retrieval.

These inter-linked datasets constitute the Web of Data. The content of every dataset can be transposed into a graph model, representing entities (i.e. information resources) and their relationships. Compared to the Web of Documents, the unit of information is of smaller granularity. As a consequence the number of nodes and links is orders of magnitude larger than on the Web of Documents. As the Web of Data graph is very large, containing billions of nodes and edges, developing scalable link analysis algorithm for computing popularity score on web-scale data graph is becoming an important requirement.

Current link analysis algorithms [1,2] for the Web of Data consider exclusively the graph of entities. In addition to their high computational complexity, they suffer from not taking into account the semantics of datasets which produce sub-optimal results. For example, given a dataset about employees, one would like to find the most skilled employee and not the most popular one. In this paper, we introduce a two-layer model for the Web of Data and provide justifications for this two-layer model. Based on this model, we propose a novel ranking algorithm called DING (for Dataset rankING). DING performs ranking in three steps: 1. dataset ranks are computed by performing link analysis on the top layer (i.e. the dataset graph); 2. for each dataset, entity ranks are computed by performing link analysis on the local entity collection; 3. the popularity of the dataset is propagated to its entities and combined with their local ranks to estimate a global entity rank.

## 2 Background

In this section, we introduce a model of the Web of Data that will serve as framework in the rest of the paper. We then show how the original PageRank [3] algorithm can be adapted to this model.

### 2.1 Web Data Model

For the purpose of this paper, we need a generic graph model that encompasses the different use cases discussed previously. Therefore, we define a labelled directed graph model that covers the various data sources found on the Web of Data, i.e. Microformats, RDFa, Semantic Web repositories, etc. This graph model represents entities and their relationships. We denote by entity a self-contained unit of information that has relationships with other entities. Typical examples of entities include documents, persons, events, products, etc.

Let $U$ be a set of node labels, and $V$ a set of edge labels. The Web of Data is defined as a graph over $U$ and $V$, and is a tuple $G = \langle E, L, \lambda \rangle$ where $E$ is a set of

[1] Linked Data: http://linkeddata.org/

nodes representing entities, $L \subseteq \{(e_1, \sigma, e_2) | e_1, e_2 \in E, \sigma \in V\}$ a set of labelled edges representing the relationships (or links) and $\lambda : E \rightarrow U$ a node labelling function. The components of an edge $l \in L$ will be denoted by $source(l)$, $label(l)$ and $target(l)$ respectively.

Let a dataset $D$ be a subgraph of $G$. A dataset $D$ is a tuple $D = \langle E_D, L_D, \lambda, \Delta_D \rangle$ with $E_D \subseteq E$ and $L_D \subseteq L$. Two datasets are not mutually exclusive and their nodes may overlap, i.e. $E_{D_1} \cap E_{D_2} \neq \emptyset$. We identify a subset $\Delta_D \subseteq U$ as a set of internal node labels to a dataset $D$, i.e. the set of entity identifiers that originates from this dataset. For example, such a set could be the URIs defined by the naming authority of the dataset [4]. A node $e$ of a graph $D$ is said to be *internal* if $\lambda(e) \in \Delta_D$, otherwise it is said to be *external* (i.e. it identifies an entity from another dataset). Analogously, an edge $l$ of a graph $D$ is said to be *intra-dataset* if $\lambda(source(l)) \in \Delta_D, \lambda(target(l)) \in \Delta_D$, otherwise it is said to be *inter-dataset*.

Within a dataset graph, edges connecting two external nodes are simply ignored to avoid possibility of link spam. Any dataset could possibly create an arbitrary number of links between two external entities which may lead to anomalies in the graph and affect the link analysis algorithms.

### 2.2 PageRank

PageRank [3] is a ranking system that originates from works on Web search engines. The ranking system, based on a random walk algorithm, evaluates the probability of finding a random web surfer on any given page. The algorithm assumes a hyperlink from a page $i$ to a page $j$ as an evidence of the importance of page $j$. In addition, the amount of importance that $i$ is conferring to $j$ is determined by the importance of $i$ itself and inversely proportional to the number of pages $i$ points to. PageRank can easily be adapted to the Web of Data model. By regarding pages as entities and hyperlinks as links between entities, we can formulate PageRank as follow:

Let $L(i) = \{target(l) | \forall l \in L, source(l) = i\}$ be the set of entities linked by an entity $i$ and $B(j) = \{source(l) | \forall l \in L, target(l) = j\}$ be the set of entities that points to $j$. The PageRank $r(j)$ of an entity $j \in E$ is given by:

$$r^k(j) = \alpha \sum_{i \in B(j)} \frac{r^{k-1}(i)}{|L(i)|} + \frac{(1-\alpha)}{|E|} \,. \tag{1}$$

A fixed-point iteration approach is commonly used where the computation of a rank score $r^k(j)$ at a step $k$ uses the rank scores of the step $k-1$. The operation is repeated until all scores $r$ stabilise to within some threshold.

The PageRank formula is composed of two parts weighted by a damping factor $\alpha$, usually set to 0.85. The first component provides the contribution $\sum_{i \in B_j} \frac{r^{k-1}(i)}{|L(i)|}$ of incoming links to entity $j$. The factor $\frac{1}{|L(i)|}$ defines a uniform distribution of the importance of entity $i$ to all the entities $i$ points to. This distribution, later referred to as $w_{i,j}$, can be modified in order to provide a

distribution based on the weight of a link. The second component $\frac{1}{|E|}$ denotes the probability that the surfer will jump to $j$ from any other random entity from the collection.

## 3 Related Work

Link analysis such as PageRank [3] has been successfully applied for query independent ranking (also called static ranking). Several extensions have been developed to take into consideration weighted links, hierarchical link structure or the Semantic Web model.

**Weighted Link Analysis.** When working with more heterogeneous links, standard approaches do not provide accurate results since links of different types can have various impact on the ranking computation. In [5,6], the authors extend PageRank to consider different types of relations between entities. PopRank [7], an object-level link analysis, proposes a machine learning approach to assign a "popularity propagation factor" to each type of relation. ObjectRank [8] applies authority-based ranking to keyword search in databases. However, these works do not consider the features of links such as their specificity and cardinality to assign weights in an unsupervised fashion. Furthermore, these approaches are too complex to apply on web-scale since they will require multiple times the current processing power of current web search engines. A major task is to bring down this computational power requirement.

**Hierarchical Link Analysis.** Recent works [9,10,11,12,13,14] exploit the hierarchical structure of distributed environments and of the Web. [9] suggests a hierarchical model of the Web and shows the desirable computational properties of such approach. In [12], the authors show that hierarchical ranking algorithm outperforms qualitatively other well-known algorithms, including PageRank. However, such models have never been applied on semi-structured data sources with distinct semantics and none of them are considering weighted relations between supernodes.

**Semantic Web Link Analysis.** SemRank [15] proposes a method to rank semantic relations using information-theory techniques but is solely focussed on ranking and retrieval of relationships. The Swoogle search engine [1] is the first one to propose OntoRank, an adaptation of PageRank for Semantic Web resources. In ReconRank [2], a link analysis is applied at query time for computing the popularity of resources and documents. The above algorithms only consider the individual web resources, disregarding the semantics and structure of the Web of Data. They are therefore costly on a web-scale and likely provide sub-optimal results. We are aware of one recent study [4] that has analysed the effectiveness of PageRank on the domain level for ranking Semantic Web resources. However, their approach disregards the link structure between entities within a domain, and does not consider weighted relations.

**Our Contribution.** To address the Web of Data scenario previously described, we first illustrate a two-layer model of the Web of Data. We introduce and evaluate a hybrid algorithm that combines both weighted and hierarchical link analysis methods. The model operates in a hierarchical fashion between a dataset and entity layer leveraging an unsupervised method that considers both the specificity and cardinality of links for assigning them appropriate weights. First an extension of weighted PageRank is applied on the dataset layer. Then, the importance of each dataset node is distributed to its individuals entities, and combined with local entity ranks which can be dependent of the dataset semantic.

## 4 A Two-Layer Model for Ranking Web Data

In this section, we introduce a two layer model for the Web of Data, pictured in Fig. 1. The top layer (dataset layer) is composed of a collection of inter-connected datasets whereas the lower layer (entity layer) is composed of independent graphs of entities.

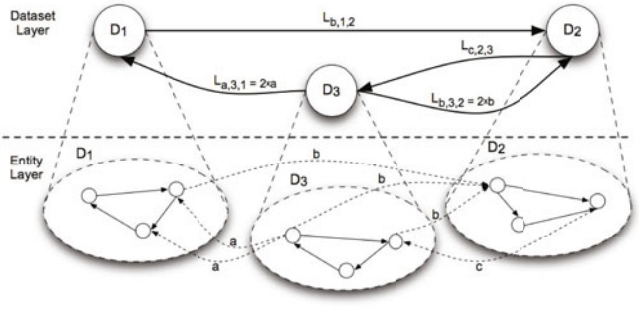



**Fig. 1.** The two-layer model of the Web of Data. Dashed edges on the entity layer represent inter-dataset links.

### 4.1 Quantifying the Two-Layer on the Web of Data

In this section, we provide evidence of the two-layer model and its desirable computational properties by quantifying the locality characteristics of links and the dataset size distribution. We perform the following simple experiments. We first take the datasets described below and count how many of the links are intra-dataset and how many are inter-dataset. Then, we analyse the dataset size distribution on a subset of the Web of Data.

**DBpedia** is a semi-structured version of Wikipedia and contains 17.7 million of entities[2].

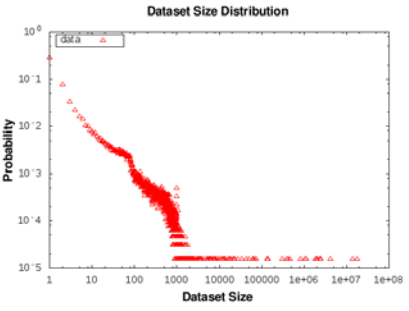
**Citeseer** is a semi-structured version of Citeseer from the RKBExplorer initiative and contains 2.48 million of entities[3] .

[2] DBpedia: http://dbpedia.org/

[3] Citeseer: http://citeseer.rkbexplorer.com/

| *Dataset* | *Intra* | *Inter* |
|---|---|---|
| Full | 287M (78.8%) | 77M (21.2%) |
| DBpedia | 88M (93.2%) | 6.4M (6.8%) |
| Citeseer | 12.9M (77.7%) | 3.7M (22.3%) |
| Geonames | 59M (98.3%) | 1M (1.7%) |

(a) Ratio intra / inter dataset links




(b) Distribution of the size of datasets

**Fig. 2.** Statistics about link locality and dataset size

**Geonames** is a geographical database and contains 13.8 million of entities[4].

**Sindice's Page-Repository** contains 60 million of entities among 50.000 datasets (including the previous). It is a representative subset of the Web of Data. It is composed of Semantic Web online repositories and pages with microformats or RDFa markups crawled on a regular basis for more than two years[5].

Table 2(a) shows that 78.8% of the links are intra-dataset. Such connectivity statistics are not far from the previous results of [9] where 83.9% of the links from a large web pages dataset are intra-domain links. On individual datasets, inter-dataset links in DBpedia represent only 6.8% of its total links. For Citeseer, the number of inter-dataset links is higher than other datasets but can be explained by the fact that this dataset is using an external ontology to describe its data, hence most of its inter-dataset links point to only one external dataset (the dataset ontology). Geonames is representative of a "dataset sink", a dataset loosely linked with other datasets. These numbers confirm a high degree of locality on the Web of Data, and suggest the two-layer model proposed in this paper.

Fig. 2(b) depicts the distribution of the size of all datasets found in Sindice's page-repository. The distribution nearly follows a powerlaw and corresponds to previous research on the size of web sites [13]. We observe that the majority of the datasets contain less than 1000 nodes which indicates that local rank computation within these graphs can be performed in an efficient manner in memory.

### 4.2 The Dataset Graph

The top layer, or dataset graph, can be seen as an approximation of the global graph $G$. Instead of considering entities and links between these entities, we are using higher-level information such as datasets and *linksets*. Given a dataset $D$, we denote a linkset with $L_{\sigma,i,j} = \{l | label(l) = \sigma, source(l) \in D_i, target(l) \in D_j\}$ the

[4] Geonames: `http://www.geonames.org/`

[5] Sindice: `http://www.sindice.com/`

set of edges having the same label $\sigma$ and connecting the dataset $D_i$ to the dataset $D_j$. For example, in Fig. 1 the inter-dataset links (dashed-edges labelled $a$) between $D_3$ and $D_1$ are aggregated to form the linkset $L_{a,3,1}$ on the dataset layer.

The resulting graph (50.000 nodes, 1.2M of linksets) is orders of magnitude smaller than the original graph $G$ (60M nodes, 364M of links). As a consequence, it can be easily kept in memory (in the case of Sindice) and dataset ranks can be computed on demand.

### 4.3 The Entity Graph

The lower layer, or entity graph, is composed of disjoint graphs $D$ each of them being a collection of internal nodes and intra-dataset edges. The direct consequence is that the computation of ranks at entity level can be computed in an independent manner (on a per dataset basis) and can be easily parallelised. Since computations are performed independently, the complexity that would dominate is that of the largest dataset, e.g. DBpedia in our case. However, the majority of the graphs has a small number of nodes as shown in Fig. 2(b). This means that the graphs can be easily kept in memory and rank computation can be performed without the performance penalty of IO access generally encountered when processing very large graphs.

## 5 The DING Model

In this section, we start by introducing an unsupervised method for weighting links and linksets. Next, the DING algorithm is described. We first reformulate the original PageRank algorithm for computing dataset ranks. We present two local entity ranking algorithms as well as a list of semantic-dependent algorithms that is known to outperform standard algorithms for certain type of dataset. We finally explain how to combine dataset ranks with local entity ranks in order to estimate a global entity ranking.

### 5.1 Unsupervised Link Weighting

In Fig. 1 the probability of the user going from $D_3$ to $D_1$ is likely to be different from the probability of going to $D_2$ since the label and number of links associated to $L_{a,3,1}$ are not the same as the ones associated to $L_{b,3,2}$. The goal is to define a linkset weighting function $w_{\sigma,i,j}$.

Weights can be assigned based both on the number of links contained in a linkset and on the general importance of the label involved in the link. Our approach is derived from TF-IDF to measure the relevance of a label given its frequency in a data collection. We define the linkset weighting function $w$ using the Link Frequency - Inverse Dataset Frequency (LF-IDF):

$$w\sigma, i, j = LF(L_{\sigma,i,j}) \times IDF(\sigma) = \frac{|L_{\sigma,i,j}|}{\sum_{L\tau,i,k} |L_{\tau,i,k}|} \times \log \frac{N}{1 + freq(\sigma)} \,. \qquad (2)$$

In Eq. (2), $N$ denotes the number of datasets and $freq(\sigma)$ is the frequency of occurrence of the label $\sigma$ in the collection of datasets. These weights can

be computed dynamically given accumulated statistical information in the database. The LF-IDF scheme assigns a higher degree of importance to link with a high frequency (in a given dataset) and low dataset frequency in the dataset collection. For example, this weighting scheme will favour links such as `foaf:knows` compared to very frequent links such as `rdfs:seeAlso`. Former results [16] have shown that link weighting improves dataset ranking.

### 5.2 DING Algorithm

The DING algorithm is an extension of PageRank (Eq. 1) for the two-layer graph structure presented in Sect. 2.1. Instead of visiting web pages, the random surfer browses datasets. The random walk model is as follows:

1. At the beginning of each browsing session, a user randomly selects a dataset.
2. Then, the user may choose one of the following actions:
   (a) Selecting randomly an entity in the current dataset.
   (b) Jumping to another datasets that is linked by the current dataset.
   (c) Ending the browsing.

According to the hierarchical random walk model, we can apply a two-stage computation. In the first stage, we calculate the importance of the top level dataset nodes which is explained next. The second stage calculates the importance of entities within a dataset, as explained in Sect. 5.4.

### 5.3 Computing DatasetRank

The dataset surfing behaviour is the same as in PageRank. We can obtain the importance of dataset nodes by applying PageRank on the weighted dataset graph.

As in (1), the rank score $r(D_j)$ of a dataset is composed of a part corresponding to the rank contribution from the datasets linking to $D_j$ and of a part corresponding to the probability of a random jump to $D_j$ from any dataset in the collection. The probability of selecting a dataset during a random jump is proportional to its size, i.e. $|E_{D_j}|$. The distribution factor $w_{\sigma,i,j}$ is defined by Eq. 2. The final DatasetRank formula is given below. The two parts are combined using the damping factor $\alpha = 0.85$, since we observed that this value provides also good results in our experimental evaluation.

$$r^k(D_j) = \alpha \sum_{L\sigma,i,j} r^{k-1}(D_i) w_{\sigma,i,j} + (1-\alpha) \frac{|E_{D_j}|}{\sum_{D \in \mathcal{G}} |E_D|} \,. \tag{3}$$

### 5.4 Computing Local Entity Rank

A method used in layered ranking algorithms is to assign to the page the importance of the supernode [10,4]. In our case this would correspond to assign the DatasetRank score of a dataset to all its entities. In large datasets, such as DBpedia, this approach does not hold. A query is likely to return many entities

from a same dataset with the same rank. This unnecessarily pushes part of the ranking problem at query time. Instead we can assign a score combining both the importance of a dataset and the importance of an entity within its dataset.

Next, we present two generic algorithms, the weighted EntityRank and the weighted LinkCount, that compute entity ranks on any type of graphs. However, we argue that entity ranking is strongly dependent of the semantic of the dataset. A list of existing semantic-dependent algorithms is discussed afterwards.

**Weighted EntityRank.** The Weighted EntityRank method uses the PageRank algorithm from Eq. 1 applied on the internal entities and intra-links of a dataset in order to compute the importance of an entity node within a dataset. In our experimental setup, we use the LF-IDF weighting scheme from Eq. 2 on single links between entities. Like PageRank, the robustness against spam of the EntityRank method makes it a good choice for datasets build on non-controlled user inputs like Livejournal or Last.fm.

**Weighted LinkCount.** The Weighted LinkCount is a variant of the *in-degree counting links* method [17], an alternative to EntityRank when the dataset can be assumed mostly deduplicated and spam-free. This is often true for very well curated user-input datasets like DBpedia. For each entity $j$, its rank $r(j)$ is given by $r(j) = \sum_{l_{\sigma,i,j}} w(l_{\sigma,i,j})$ where $w(l_{\sigma,i,j})$ is the weight of the link from $i$ to $j$. The weighting scheme is the same as the one used in EntityRank. LinkCount is more efficient to compute than EntityRank, since it needs only one "iteration" over the data collection.

**Semantic-Dependent Entity Ranking.** Datasets on the Web of Data may have their own semantic with a variety of graph structures. For example, we can mention generic graphs coming from user inputs, hierarchical graphs, bipartite graphs, etc. A complete taxonomy of the different graph structures among existing datasets is beyond the scope of the paper, but several examples are presented in Tab. 1.

While EntityRank and LinkCount represent good generic solutions for local entity ranking, as shown in Sect. 7.2, an approach which takes into account the peculiar properties of each dataset will give better results. Considering that in literature there are already a notable amount of ranking algorithms that are dataset specific, such as [18,19] for citation networks or Dissipative Heat Conductance [12] for strongly hierarchical datasets like taxonomies or geo-databases, DING has been designed to exploit better alternatives to LinkCount and EntityRank. One can also define its own algorithm using dataset-dependent ranking criteria. If the given local entity ranking is modelled as a probability distribution, combining it with DatasetRank means simply calculating the joint probability as explained next.

### 5.5 Combining DatasetRank and Entity Rank

A straightforward approach for combining dataset and local entity ranks is to adopt a purely probabilistic point of view by interpreting the dataset rank $r(D)$

**Table 1.** List of various graph structures with targeted algorithms

| *Graph Structure* | *Dataset* | *Algorithm* |
|---|---|---|
| Generic, Controlled | DBpedia | LinkCount |
| Generic, Open | Social Communities | EntityRank |
| Hierarchical | Geonames, Taxonomies | DHC |
| Bipartite | DBLP | CiteRank |

as the probability of selecting the dataset and the local entity rank $r(e)$ as the probability of selecting an entity within this dataset. Hence we would have the global score $r_g(e)$ defined as $r_g(e) = P(e \cap D) = r(e) * r(D)$.

But this approach favours smaller datasets. The local entity ranks is much higher in small datasets than in larger ones, since in the probabilistic model all ranks in a dataset sum to 1. As a consequence any small dataset receiving even a single link is likely to have its top entities score way above many of the top ones from larger datasets. The solution is to normalize the local ranks to a same *average* based on the dataset size. In our experiments we use the following formula for ranking an entity $e$ in a dataset $D$: $r_g(e) = r(D) * r(e) * \frac{|E_D|}{\sum_{D' \in G} |E'_D|}$ .

## 6 Scalability of the DING Approach

A precise evaluation of the scalability of our approach is not the goal of this paper. Moreover, [9] has shown that hierarchical ranking algorithms provide speedup in computation compared to standard approaches. However, we report here some performance results from the DING method when applied on a real use-case scenario, i.e., the Sindice search engine.

Given the small size of the dataset graph as shown in Sect. 4.2, the graph can be fully held in memory and rank computation can be performed on demand. A single iteration of DatasetRank computation takes 200ms to process 50k datasets on commodity hardware (Intel Xeon E5410 Quad Cores), a good quality rank can hence be obtained in a matter of seconds. If we define a measure of convergence of the algorithm at an iteration $k + 1$ as in Eq. 4, the algorithm converges to a 0.1% threshold in 32 iterations, which represents 5 seconds.

$$\rho(k+1) = \max_{D_i \in \mathcal{D}} \frac{|r^{k+1}(D_i) - r^k(D_i)|}{r^k(D_i)} \tag{4}$$

Since the size of the majority of the datasets is in order of thousands of nodes as shown in Sect. 4.3, their entity graph can also be held entirely in memory making more effective the computation of entity ranks. Moreover, since the computation of entity ranks in one dataset is independent of the entity rank from another datasets, the computation can be easily distributed over a cluster of machines.

On the other hand, the computation of entity ranks in large datasets can become a heavy operation considering that the largest dataset (i.e., DBpedia) is containing over ten million entities and tenths of millions links. For such

dataset, we fall back on standard methods to parallelise the computation such as the *Map-Reduce* programming model [20]. Computing[6] the local entity ranks of the DBpedia dataset with a 0.1% precision took 55 iterations of 1 minute each on a Map-Reduce cluster composed of three Intel Xeon quad cores. In the case of LinkCount, the computation would require only one such iteration.

In addition, separating dataset ranks and local entity ranks minimizes the amount of computation required when updating the data collection. For example, a new dataset $D_i$ which has links to several other datasets has to be indexed by Sindice. With standard non-hierarchical ranking models, the ranks of all entities would have to be recomputed. In contrast, with the DING model the only set of ranks to be computed are 1. the ranks of entities in dataset $D_i$; and 2. the dataset ranks which can be recomputed in a matter of seconds. This decreases the cost of the update from being proportional to the size of the Web of Data to being proportional to the size of the dataset $D_i$ and of the dataset graph.

## 7 Experiments and Results

We introduced a novel ranking model, showed that it can cope with dataset semantics and gave evidences about its desirable computational properties. But it is not yet clear if the DING model provides worst, similar or better performance than standard approaches. In order to assess the performance of DING, we conduct two experiments. The baseline algorithm that we use for comparison is a global version of EntityRank (GER). This algorithm is similar to the one described in Sect. 5.4 with the only difference that it operates on the full Web of Data graph $G$. We use the datasets presented in Sect. 4.1 for the two experiments.

The first experiment investigates the impact of link locality on the Web of Data by comparing the performance of the two generic local algorithms, the local EntityRank (LER) and local LinkCount (LLC), with GER. This first experiment is done without user intervention by measuring the rank correlation between the algorithms. While this experiment provides strong evidence of the effectiveness of LLC and LER, it does not assess the quality of retrieval. The second experiment evaluates the effectiveness of the local algorithms and of DING through a user study in order to judge if they provide worst, similar or better performance than the baseline approach.

### 7.1 Accuracy of Local Entity Rank

The goal of this experiment is to compare the static ranks in a query independent manner of the local algorithms (LER and LLC) with the global one (GER) in order to judge the impact of the link locality on the Web of Data. We measure the Spearman's correlation [21] of the two local algorithms with GER using the full entity rank list of three datasets: DBpedia, Citesser and Geonames. The Spearman's correlation has already been employed in [18] to compare several

[6] Including Input/Output disk operations as well as data preprocessing.

**Table 2.** Spearman's correlation between LLC and LER with GER

| *Algorithm* | *DBpedia* | *Citeseer* | *Geonames* |
|---|---|---|---|
| LLC | 0.79 | 0.86 | 0.73 |
| LER | 0.88 | 0.97 | 0.78 |

ranking algorithms. The results are presented in Table 2[7]. While LLC performs slightly worse than LER, the Spearman's correlation indicates a strong correlation of the two algorithms with GER. These results confirm that, on individual datasets, GER can be well approximated with computational methods of lower complexity such as LER or LLC due to the high degree of link locality.

### 7.2 User Study

Information Retrieval experiments focus on retrieval effectiveness, expressed in terms of recall and precision. In general, data collections such as the one provided by TREC or INEX are employed to assess the ranking produced by systems. The TREC or INEX entity tracks are corpus created for the evaluation of entity-related searches. However, they are not suitable for our use cases where queries of various complexity are used and where the goal is to measure the effectiveness of ranking among inter-linked datasets. Therefore, in order to evaluate qualitatively the DING methodology, we decided to perform a user study where users provide relevance judgements for each algorithm.

*Design.* The user study is divided into two experiments: 1. the first one (Exp-A) assesses the performance of local entity ranking on the DBpedia dataset; 2. the second one (Exp-B) assesses the performance of local entity ranking on the full Sindice's page-repository. Each experiment includes 10 queries, varying from simple keyword search to more complex structured queries (SPARQL). Each participant receives a questionnaire containing a description of the query in human language, and three lists of top-10 results. Each result is described by the human-readable label and the URI of the entity. The first list corresponds to the ranking results of GER. For Exp-A, the second and third lists correspond to the ranking results of LER and LLC while for Exp-B the ranking results are from DatasetRank combined with LER and LLC (DR-LER and DR-LLC resp.). The second and third lists are named randomly "Ranking A" or "Ranking B" so no information about the ranking algorithm and no correlation between Ranking A and B on two questionnaires can be inferred. For these experiments, we consider the effect of link-based features in combination with textual features. Therefore, the three lists of results are ordered using both the static rank from the link analysis algorithms and a query-dependent ranking (similar to BM25), combined using a simple linear combination.

[7] The Spearman's correlation coefficient tests the strength of the relationship between two variables, i.e. ranks produced by LER or LLC and GER. The values varies between 1 (a perfect positive correlation) and -1 (a perfect negative correlation). A value of 0 means no particular correlation.

**Table 3.** Chi-square test for Exp-A and Exp-B. The column $\%\chi^2$ gives, for each modality, its contribution to $\chi^2$ (in relative value).

(a) LER

| Rate | $O_i$ | $E_i$ | $\%\chi^2$ |
|---|---|---|---|
| **B** | 0 | 6.2 | −13% |
| **SB** | 7 | 6.2 | +0% |
| **S** | 21 | 6.2 | +71% |
| **SW** | 3 | 6.2 | −3% |
| **W** | 0 | 6.2 | −13% |
| **Totals** | 31 | 31 | |

(b) LLC

| Rate | $O_i$ | $E_i$ | $\%\chi^2$ |
|---|---|---|---|
| **B** | 3 | 6.2 | −12% |
| **SB** | 8 | 6.2 | +4% |
| **S** | 13 | 6.2 | +53% |
| **SW** | 6 | 6.2 | −0% |
| **W** | 1 | 6.2 | −31% |
| **Totals** | 31 | 31 | |

(c) DR-LER

| Rate | $O_i$ | $E_i$ | $\%\chi^2$ |
|---|---|---|---|
| **B** | 12 | 11.6 | +0% |
| **SB** | 12 | 11.6 | +0% |
| **S** | 22 | 11.6 | +57% |
| **SW** | 9 | 11.6 | −4% |
| **W** | 3 | 11.6 | −39% |
| **Totals** | 58 | 58 | |

(d) DR-LLC

| Rate | $O_i$ | $E_i$ | $\%\chi^2$ |
|---|---|---|---|
| **B** | 7 | 11.6 | −9% |
| **SB** | 24 | 11.6 | +65% |
| **S** | 13 | 11.6 | +1% |
| **SW** | 10 | 11.6 | −1% |
| **W** | 4 | 11.6 | −24% |
| **Totals** | 58 | 58 | |

*Participants.* Exp-A evaluation is performed on 31 participants, and Exp-B evaluation on 58 participants. The participants consist of researchers, doctoral and master students and technicians. All of the participants are familiar with search engines, but a few of them familiar with entity search engines.

*Task.* The task is to rate "Ranking A" in relation to the standard one using categorical variable, then to rate "Ranking B" in relation to the standard one. The participants have to choose between 5 categories: Better (B), Slightly Better (SB), Similar (S), Slightly Worse (SW), Worse (W). The questionnaires and the raw results of the user study can be downloaded at `http://ding.sindice.com/`

*Measure.* We use the Pearson's chi-square to perform the test of "goodness of fit" between $O$, the observed frequency distribution (the participant's judgements) of the previous categories, and $E$, an expected theoretical uniform distribution (equiprobable) of these categories, in order to establish whether or not the observed distribution differs from the theoretical distribution. Our null hypothesis is that the observed frequency distribution is uniform. We then interpret the contribution to chi-square of each category.

**Exp-A Results.** The tables 3(a) and 3(b) show the results[8] of the chi-square test for LER and LLC respectively. For the tests to be significant at the 1% level, with 4 degrees of freedom, the value for chi-square has to be at least 13.3. Since the chi-square test yields 49.48 for LER and 14 for LLC, we can reject the null hypothesis for the two tests. It bears out that a large proportion of the population (+71% of contribution to $\chi^2$) considers LER similar to the GER. For LLC, a majority of the population (+53% of contribution to $\chi^2$) considers it similar to GER, and this is reinforced by the fact that a minority (−31% of contribution to $\chi^2$) considers it worse.

To conclude, at 1% significance level, LER and LLC provides similar results than GER. However, there is a more significant proportion of the population that considers LER more similar to GER.

**Exp-B Results.** The tables 3(c) and 3(d) show the results of the chi-square test for DR-LER and DR-LLC respectively. For the tests to be significant at the

[8] Intermediate calculation steps are omitted.

1% level, with 4 degrees of freedom, the value for chi-square has to be at least 13.3. Since the chi-square test yields 16.31 for DR-LER and 20.45 for DR-LLC, we can reject the null hypothesis for the two tests. It bears out that a good proportion of the population (+57% of contribution to $\chi^2$) considers DR-LER similar to GER, strengthen by the fact that a minority (−39% of contribution to $\chi^2$) considers it worse. For DR-LLC, a large proportion of the population (+65% of contribution to $\chi^2$) considers it slightly better than GER, which is comforted by the fact that a minority (−24% of contribution to $\chi^2$) considers it worse.

To conclude, at 1% significance level, the two algorithms give a profile of preference quite different. It appears that DR-LLC provides a better effectiveness. Indeed, a large proportion of the population finds its results slightly better than GER, and this is reinforced by a few number of people finding it worse.

## 8 Conclusion and Future Work

We presented DING, a novel two-layer ranking model for the Web of Data. DING is specifically designed to address the Web of Data scenario, computing the popularity score of entities on web-scale graph. As opposed to alternative approaches, we explain its desirable computational properties and display experimental evidence of improved ranking quality. Furthermore, since DING allows for improved local ranking by using dataset-specific ranking algorithms, further works will be done in the area of automation of graph structure recognition. This would allow better match of specific ranking algorithms to a graph semantic, and will improve the performance of the ranking on a heterogeneous Web of Data.

## Acknowledgments

This material is based upon works supported by the European FP7 project *Okkam - Enabling a Web of Entities* (contract no. ICT-215032), and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

## References

1. Ding, L., Pan, R., Finin, T.W., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the semantic web. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 156–170. Springer, Heidelberg (2005)
2. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: Proceedings of Second International Workshop on Scalable Semantic Web Knowledge Base Systems, Athens, GA, USA (November 2006)
3. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (1999)
4. 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)

5. Xing, W., Ghorbani, A.: Weighted pagerank algorithm. In: CNSR 2004: Proceedings of the Second Annual Conference on Communication Networks and Services Research, Washington, DC, USA, pp. 305–314. IEEE Computer Society, Los Alamitos (2004)
6. Baeza-Yates, R., Davis, E.: Web page ranking using link attributes. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, pp. 328–329. ACM, New York (2004)
7. Nie, Z., Zhang, Y., Wen, J.R., Ma, W.Y.: Object-level ranking: bringing order to Web objects. In: Proceedings of the 14th International Conference on World Wide Web, p. 567. ACM, New York (2005)
8. Balmin, A., Hristidis, V., Papakonstantinou, Y.: Objectrank: authority-based keyword search in databases. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB Endowment, pp. 564–575 (2004)
9. Kamvar, S., Haveliwala, T., Manning, C., Golub, G.: Exploiting the block structure of the web for computing pagerank. Technical Report 2003-17, Stanford InfoLab (2003)
10. Eiron, N., McCurley, K.S., Tomlin, J.A.: Ranking the Web Frontier. In: Proceedings of the 13th Conference on World Wide Web, vol. 2, pp. 309–318. ACM Press, New York (2004)
11. Wang, Y., DeWitt, D.J.: Computing pagerank in a distributed internet search system. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB Endowment, Toronto, Canada, pp. 420–431 (2004)
12. Xue, G.R., Yang, Q., Zeng, H.J., Yu, Y., Chen, Z.: Exploiting the hierarchical structure for link analysis. In: Proceedings of the 28th Annual International ACM SIGIR Conference, pp. 186–193. ACM, New York (2005)
13. Feng, G., Liu, T.Y., Wang, Y., Bao, Y., Ma, Z., Zhang, X.D., Ma, W.Y.: Aggregaterank: bringing order to web sites. In: Proceedings of the 29th Annual International ACM SIGIR Conference, p. 75. ACM Press, New York (2006)
14. Broder, A.Z., Lempel, R., Maghoul, F., Pedersen, J.: Efficient pagerank approximation via graph aggregation. Information Retrieval 9, 123–138 (2006)
15. Anyanwu, K., Maduko, A., Sheth, A.: Semrank: ranking complex relationship search results on the semantic web. In: Proceedings of the 14th International Conference on World Wide Web, pp. 117–127. ACM, New York (2005)
16. Toupikov, N., Umbrich, J., Delbru, R., Hausenblas, M., Tummarello, G.: DING! Dataset Ranking using Formal Descriptions. In: WWW 2009 Workshop: Linked Data on the Web (LDOW 2009), Madrid, Spain (2009)
17. Najork, M.A., Zaragoza, H., Taylor, M.J.: Hits on the web: how does it compare? In: Proceedings of the 30th Annual International Annual ACM Conference on Research and Development in Information Retrieval (2007)
18. Sayyadi, H., Getoor, L.: Futurerank: Ranking scientific articles by predicting their future pagerank. In: SDM, pp. 533–544 (2009)
19. Walker, D., Xie, H., Yan, K.K., Maslov, S.: Ranking scientific publications using a simple model of network traffic. In: CoRR (2006)
20. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM 51(1), 6 (2008)
21. Melucci, M.: On rank correlation in information retrieval evaluation. SIGIR Forum 41(1), 18–33 (2007)

# A Node Indexing Scheme for Web Entity Retrieval

Renaud Delbru[1], Nickolai Toupikov[1],
Michele Catasta[2,⋆], and Giovanni Tummarello[1,3]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland
firstname.lastname@deri.org
[2] School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland
firstname.lastname@epfl.ch
[3] Fondazione Bruno Kessler
Trento, Italy
lastname@fbk.eu

**Abstract.** Now motivated also by the partial support of major search engines, hundreds of millions of documents are being published on the web embedding semi-structured data in RDF, RDFa and Microformats. This scenario calls for novel information search systems which provide effective means of retrieving relevant semi-structured information. In this paper, we present an "entity retrieval system" designed to provide entity search capabilities over datasets as large as the entire Web of Data. Our system supports full-text search, semi-structural queries and top-k query results while exhibiting a concise index and efficient incremental updates. We advocate the use of a node indexing scheme and show that it offers a good compromise between query expressiveness, query processing time and update complexity in comparison to three other indexing techniques. We then demonstrate how such system can effectively answer queries over 10 billion triples on a single commodity machine.

## 1 Introduction

On the Web, more and more structured and semi-structured data sources are becoming available encouraged by initiatives such as Linked Open Data, but now even more with the support of major search engines. Hundreds of millions of documents already embed semi-structured data in the form of RDF, RDFa and Microformats and it is easy to predict that more will join soon. Whatever the current size of the Web of Data is today, the trend is clear and so is the requirement for handling semi-structured data with a scalability in the same class of traditional search engines.

⋆ The author contributed to this work while he was a master student in DERI.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 240–256, 2010.

However, the mass publishing of data on the Web is unexploitable by semantic clients and applications if supporting tools are not made available for data discovery. Taking the e-commerce example, how can a client find products matching a certain description pattern over thousands of e-commerce semantic data sources ? By entering keyword queries into a standard web search system, the results are likely to be irrelevant since the system will return pages mentioning the keywords and not the matching products themselves. Current search systems are inadequate for this task since they have been developed for a totally different model, i.e., a Web of Documents. The shift from documents to data entities poses new challenges for web search systems.

In this paper, we present the Semantic Information Retrieval Engine, SIREn, a system based on Information Retrieval (IR) techniques and designed to search "entities" specifically according to the requirements of the Web of Data. We advocate the use of a node indexing scheme for indexing semi-structured data, a technique coming from the XML Information Retrieval world. We analyse and compare the theoretical performances and other criteria of SIREn against three other indexing techniques for entity retrieval. We show that the node indexing scheme offers a good compromise between query expressiveness, query processing time and update complexity and scales well with very large datasets. The resulting system inherits from many characteristics of IR systems such as web like scalability, incremental updates and top-k queries among others.

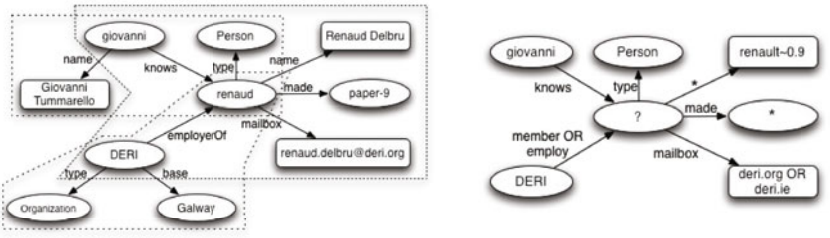
### 1.1 Web of Data: Requirements for SIREn

Developed within the Sindice project [1], SIREn is designed to be comparable in term of scalability to current web search engines so to be able to encompass, given sufficient hardware, the entire "Web of Data". The requirements are therefore:

1. Support for the multiple formats which are used on the Web of Data;
2. Support for searching an entity description given its characteristics (entity centric search);
3. Support for context (provenance) of information: entity descriptions are given in the context of a website or a dataset;
4. Support for semi-structural queries with full-text search, top-k query results, scalability over shard clusters of commodity machines, efficient caching strategy and incremental index maintenance.

With respect to point 1, the two formats which enable the annotation of entities on web pages are Microformats and RDF (RDFa is treated equally to RDF). At knowledge representation level, the main difference between Microformats and RDF is that the former can be seen as a frame model while the latter has a graph based data model. While these are major conceptual differences, it is easy to see that the RDF model can be used effectively to map Microformats[1]. Under these conditions, we have developed SIREn to cover the RDF model knowing that this would cover Microformats and likely other forms of web metadata.

[1] Any23: http://code.google.com/p/any23/




(a) Visual representation of an RDF graph. The RDF graph is divided (dashed lines) into three entities identified by the nodes *renaud*, *giovanni* and *DERI*

(b) Star-shaped query matching the entity *renaud* where *?* is the bound variable and ⋆ a wildcard

**Fig. 1.** In these graphs, oval nodes represent resources and rectangular ones represent literals. For space consideration, URIs have been replaced by their local names.

With respect to point 2 and 3, the main use case for which SIREn is developed is entity search: given a description of an entity, i.e. a star-shaped query such as the one in Fig. 1(b), locate the most relevant entities and datasets. The Fig. 1(a) shows an RDF graph and how it can be split into three entities *renaud*, *giovanni* and *DERI*. Each entity description forms a sub-graph containing the incoming and outgoing relations of the entity node which is indexed by the system.

Finally, we will see in Sect. 4 that SIREn leverages well known IR techniques to address the point 4.

### 1.2 Approaches for Entity Retrieval

Two main approaches have been taken for entity retrieval, based either on Database techniques or on Information Retrieval techniques.

**Database and Retrieval of RDF Data.** Typically, entities described in RDF are handled using systems referred to as "triplestores" or "quadstores" which usually employ techniques coming from the Database world. Some of these are built on top of existing Relational Database such as Virtuoso[2] or on top of column stores [2] while others are purposely built to handle RDF [3,4,5].

These triplestores are built to manage large amounts of RDF triples[3] or quads[4] and they do so by employing multiples indices (generally B+-Trees) for covering all kind of access patterns of the form `(s,p,o,c)`. As for Database Management Systems, the main goal of these systems is answering complex queries, e.g. those posed using the SPARQL query language[5]. The task is a superset of

[2] Virtuoso: `http://virtuoso.openlinksw.com/`

[3] Specifically a triple is a statement $s, p, o$ consisting of a subject, a predicate, and an object and asserts that a subject has a property with some value.

[4] Specifically a quad is a statement with a fourth element $c$ called "context" for naming the RDF graph, generally to keep the provenance of the RDF data.

[5] SPARQL: `http://www.w3.org/TR/rdf-sparql-query/`

entity retrieval as we defined it, and comes at the cost of maintaining complex data structures. Also they usually do not support natively top-k and full-text queries.

**Information Retrieval for Semi-Structured Data.** In the past decades, many models [6] for textual database have been designed to support queries integrating content (words, phrases, etc.) and structure (for example, the table of contents). With the increasing number of XML documents published on the Web, new structured retrieval models and query languages such as XPath/XQuery [7] have been designed. Various indexing techniques [8] have been developed to optimise the processing of the XPath query language. Amongst them, the node indexing scheme [9,10] relies on node labelling schemes [11] to encode and query the tree structure of an XML document using either a database or an inverted index.

Other communities [12,13,14] have investigated IR techniques for searching semi-structured data. [14] investigates the use of an inverted index over string sequences to support search over loosely structured datasets. Semplore [12] extends inverted index to encode RDF graph approximation and supports tree-shaped queries over RDF graphs. The first system relies on a *field-based indexing* scheme to encode attribute-value relations into the index dictionary. The second uses multiple inverted indexes to encode various structural aspects of RDF. An analysis of their limitations and advantages are discussed in Sect.5.

In the context of the Semantic Web, we are aware of one work [15] that explores the use of node labelling schemes for indexing and querying voluminous subsumption hierarchies. In comparison to SIREn, this work has focused on label querying using standard relational DBMS for subsumption check in large RDF taxonomies and can not be directly applied for the entity retrieval problem.

### 1.3 Our Contribution

Our goal is to develop an entity retrieval system that supports the previously defined requirements. In this paper, we provide the following contributions towards this goal:

- We introduce a system based on a node indexing scheme and Information Retrieval techniques for searching semi-structured representation of entities;
- We describe how a node indexing scheme can capture a semi-structured representation of an entity as well as its provenance and how it can be implemented in a inverted index;
- We compare the node indexing scheme to three other schemes. We analyse their theoretical performances, present experimental results and show that the node indexing scheme scales well with a large number of triples.

The paper is organized as follows: we first present the node-labelled data model in Sect. 2 and the associated query model in Sect. 3. We describe in Sect. 4 how to extend inverted lists as well as update and query processing algorithms to support the node index model. An analysis of the differences and

theoretical performance between SIREn and other entity retrieval systems is given in Sect. 5. In Sect. 6, experimental results are shown using large real world data collections and against other well known RDF management systems.

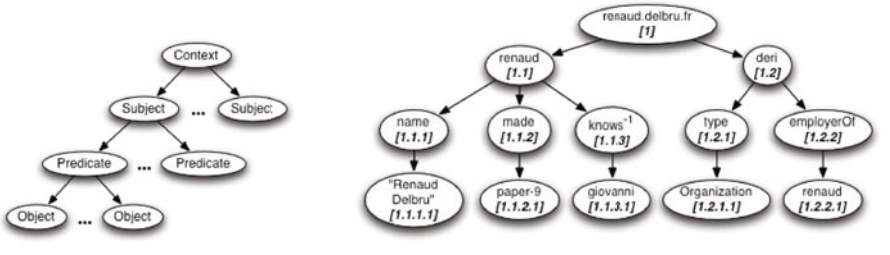
## 2 Node-Labelled Tree Model for RDF

A node-labelled tree model enables to efficiently establish relationships between nodes. The two main types of relations are *Parent-Child* and *Ancestor-Descendant* which are also core operations in XML query languages such as XPath. To support these relations, the requirement is to assign unique identifiers (node labels) that encode relationships between the nodes. Several node labelling schemes have been developed [11] but in the rest of the paper we will use a simple prefix scheme, the *Dewey Order* encoding [16]. With Dewey Order, each node is assigned a vector that represents the path from the tree's root to the node and each component of the path represents the local order of an ancestor node.

Using this labelling scheme, structural relationships between elements can be determined efficiently. An element $u$ is an ancestor of an element $v$ if label($u$) is a prefix of label($v$). Fig. 2(b) presents a data tree where nodes have been labelled using Dewey's encoding. Given the label $\langle 1.2.1.1 \rangle$ for the term `Organisation`, we can find that its parent is the predicate `rdf:type`, labelled with $\langle 1.2.1 \rangle$.

SIREn adopts a node-labelled tree model to capture datasets, entities and their RDF descriptions. The tree model is pictured in Fig. 2(a). The model has four different kind of nodes: context (dataset), subject (entity), predicate and object. Each node can refer to one or more terms. In our case, a term is not necessarily a word (from a RDF Literal), but can be an URI or a local blank node identifier.

The node-labelled model covers the quad relations CSPO (outgoing relations) and COPS (incoming relations). Incoming relations are symbolised by a predicate node with a $^{-1}$ tag in Fig. 2(b). This model is not limited to quad relations, and could in theory be used to encode longer paths such as 2-hop relations but this is beyond the scope of this paper.




(a) Conceptual representation of the node-labelled tree model

(b) Node-labelled tree model of the example dataset using Dewey's encoding

**Fig. 2.** The node-labelled tree model

## 3 Query Model

Since RDF is semi-structured, we aim to support three types of queries: 1. full-text search (keyword based) when the data structure is unknown, 2. semi-structural queries (complex queries specified in a star-shaped structure) when the data schema is known, 3. or a combination of the two (where full-text search can be used on any part of the star-shaped query) when the data structure is partially known. In this section, we present a set of query operators over the content and structure of the node-labelled tree which covers the three types of queries. We will present the operators of SIREn and whenever possible compare them with their SPARQL equivalents (in Listing 1.1).

*Content operators.* The content query operators are the only ones that access the content of a node, and are orthogonal to the structure operators. They include extended boolean operations such as boolean operators (intersection, union, difference), proximity operators (phrase, near, etc.) and fuzzy or wildcard operators.

These operations allow to express complex keyword queries for each node of the tree. Interestingly, it is possible to apply these operators not only on literals, but also on URIs if they are normalized (i.e., tokenized). For example one could just use the local name, e.g. `name`, to match `foaf:name` ignoring the namespace.

*Structure operators.* In the following, we define a set of operations over the node-labelled tree. Thanks to these operations, we are able to search content to limited nodes, to query node relationships and to retrieve paths of nodes matching a given pattern. Combination of nodes are possible using set operators, enabling the computation of entities and datasets matching a given star-shaped query.

***Ancestor-Descendant: A//D*** A node A is the ancestor of a node D if it exists a path between A and D. For example, the SPARQL query in Listing 1.1, line 1, can be interpreted as an Ancestor-Descendant operator, line 2, and will return the path $\langle 1.2.2.1 \rangle$.

***Parent-Child: P/C*** A node P is the parent of a node C if P is an ancestor of C and C is exactly one level above P. For example, the SPARQL query in Listing 1.1, line 3, can be translated into a Parent-Child operator, line 4, and will return the path $\langle 1.1.1.1 \rangle$.

**Set manipulation operators:** These operators allow to manipulate nodes (context, subject, predicate and object) as sets, implementing union ($\cup$), difference ($\setminus$) and intersection ($\cap$). For example in Listing 1.1, the SPARQL query, line 5, can be interpreted as two PC and one intersection operators, line 6.

Compared to their XML equivalent, The AD and PC operators take in consideration the level of the nodes at query processing time in order to avoid false-positive results. For example, in Listing 1.1, line 2, the keywords `deri` and `renaud` are restricted to match subject nodes and object nodes respectively. Also, operators can be nested to express longer path as shown in Listing 1.1, line 7 and 9. However, the later is possible only if deeper trees have been indexed, i.e. 2-hop relations of an entity.

**Listing 1.1.** SPARQL queries and their SIREn interpretation

```
SELECT DISTINCT ?g WHERE { GRAPH ?g { <deri> ?p <renaud> }}
deri // renaud
SELECT DISTINCT ?g ?s WHERE { GRAPH ?g { ?s <name> "Renaud Delbru" }}
name / "Renaud Delbru"
SELECT DISTINCT ?g ?o WHERE {GRAPH ?g { <giovanni> <knows> ?o. <deri> <employerOf> ?o.}}
knows^-1 / giovanni AND employerOf^-1 / deri
SELECT DISTINCT ?s WHERE { GRAPH <renaud.delbru.fr> { ?s <knows> <renaud> }}
renaud.delbru.fr // knows / renaud
SELECT DISTINCT ?g ?s WHERE { GRAPH ?g { ?s <employerOf> ?o . ?o <name> "renaud" . }}
employerOf // name / "renaud"
```

## 4 Implementing the Model

In this section, we present the data format of the inverted list and the related update and query processing algorithms. This inverted list, in addition of being able to capture quad information, has distinctive features such as efficient incremental updates of entities in an existing context and self-indexing over contexts and subjects for faster access.

### 4.1 Inverted Lists

We will now explain how the structural information associated with a term can be transposed into a postings list. The format of the inverted list is similar to the path-based model described in [17]. In this model, the inverted list stores a term occurrence with a path from the root node (context) to the node that contains the word. For example, for a term that appears in a predicate the associated path will be $\langle$*context, subject, predicate*$\rangle$ while the path for a term that appears in an object will be $\langle$*context, subject, predicate, object*$\rangle$.

The inverted index $I$ is built as a collection of $n$ inverted lists $I_{t_0}, I_{t_1}, \ldots, I_{t_n}$ where a list $I_t$ contains a posting for each occurrence of the term $t$ in the data collection. A posting list holds a sequence of term occurrences in the format shown below. The path and positional information are stored in a term-interleaved [18] manner where various parts of the posting list are stored separately. In addition to provide effective compression, it enables incremental updates of an existing context as explained in the next section.

**Listing 1.2.** Posting list format

```
Term -> <cid, tef, EntityInfo*>^tcf
EntityInfo -> <sid, freq, NodeInfo*>^tef
NodeInfo -> <pid, pos> | <pid, oid, pos> | ...
```

In Listing 1.2, each *Term* is associated to a first information block, an ordered list of context identifiers *cid* of size *tcf* ("term context frequency", the number of contexts mentioning the term). Each context identifier is immediately followed by *tef* and *EntityInfo* which correspond respectively to the "term entity frequency" (the number of entity in the context mentioning the term) and the pointer to

the entity information block. The *EntityInfo* block is an ordered list of subject identifiers *sid* of length *tef*. Each subject identifier is immediately followed by the term frequency *freq* in this entity and a pointer *NodeInfo* to the information block containing remaining path and positional information for each term occurrence. The remaining path information of a term is defined by the predicate identifier *pid* and optionally by the object identifier *oid* if and only if the term belongs to an object. The position offset *pos* of the term within a node is also stored in order to enable phrase and proximity queries. Information is ordered first by predicate identifier then by object identifier and finally by position.

To produce compact posting lists, integers are stored as a difference, or delta representation [17], using variable-length byte encoding. The key idea of the delta compression is to store the difference between consecutive values instead of the values themselves. However, more advanced compression techniques [19] could be implemented instead.

### 4.2 Incremental Update of the Inverted Lists

The proposed model supports incremental updates of datasets and entities as it is performed for documents in traditional inverted indexes [20]. Adding a dataset or entity corresponds to adding a set of statements to the inverted index. The insertion of one quad (s,p,o,c) is performed by 1. accessing the postings list of the term p and o, and 2. appending to each postings list a new entry that contains the context identifier, the subject identifier as well as the related structural and positional information of the term. The interleaved structure of the posting list enables to add a new entity to an existing context. In that case, the information block containing the list of contexts is accessed in order to increment *tef* and retrieve the pointer of the related *EntityInfo* block. Then, a new entry is appended to the *EntityInfo* and *NodeInfo* blocks.

The complexity of insertion of one quad is $O(log(n) + 1)$, where $O(log(n))$ is the cost of searching a term in a dictionary of $n$ terms and $O(1)$ is the cost of appending a posting to the list. When updates are performed by batches, the update time is linear with the number of postings to append.

Compared to triple stores, we do not support the deletion on a statement granularity, but we support the deletion of a context or subject, i.e. a set of statements. When a context or subject is removed, their identifier is inserted into a *deletion table*. During query processing, each posting entries is checked against the deletion table in $O(1)$ to ensure that it has not been deleted. The deletion table is integrated back to the inverted index only when a certain amount of deletion is sufficient to amortize the cost of such maintenance operation.

### 4.3 Query Processing

The evaluation of a query works in a bottom-up fashion. We first perform matching on the content (terms) of a node, then structural information is used during postings list intersection for filtering the result candidates that do not belong to the same node. The methodology for intersecting two postings lists is described by the following merge algorithm:

1. The postings list of each term is retrieved.
2. We then walk through the postings lists simultaneously.
3. At each step, we first compare the context and subject identifiers, then the predicate identifier and finally the object identifier. If they are the same, we put the pair `cid, sid` in the result list and advance the pointers to their next position in each postings list.

The worst-case complexity of a query evaluation is in time linear to the total number of posting entries [21]. In the average case, the complexity of an intersection is reduced to sub-linear time with an internal index (or skip lists [22]) over the context and subject identifiers to skip over unnecessary records.

Each query operator delivers output in sorted order. Multiple operators can be nested without losing the sorted order of the output, therefore enjoying the concept of interesting orderings [23] enabling the use of the effective merge-join without intermediate result sorting.

In addition, it is possible to apply existing scoring schemes such as TF-IDF or BM25F to compute top-k results at query time based on the keywords and structure of the matching sub-graphs. During the concurrent postings traversal we compute the score of one dataset-entity at a time, similarly to the document-at-a-time scoring in text database. However, more advanced top-k processing algorithms [20] could be employed instead.

## 5 Comparison among Entity Retrieval Systems

In this section, we evaluate four entity retrieval systems: SIREn based on a node-labelled index, field-based indexes [14], RDF databases [5] based on quad tables and Semplore [12]. These techniques are representative of the current approaches for entity retrieval.

Field-based indexing schemes are generally used in standard document retrieval systems (such as Apache Lucene) to support basic semi-structured information like document's field (e.g., the title). A field index constructs index terms by concatenating the field name (i.e., predicate URI) with the terms from the content of this field. For example, in the graph depicted in Fig. 1(a), the index terms for the entity "giovanni" and its predicate *name* will be represented as *name:giovanni* and *name:tummarello*.

Semplore is an Information Retrieval engine for querying Semantic Web data which supports hybrid queries, i.e. a subset of SPARQL mixed with full text search. Semplore is also built on inverted lists and relies on three inverted indexes: 1. an ontology index that stores the ontology graph (concepts and properties), 2. an individual path index that contains information for evaluating path queries, and 3. an individual content index that contains the content of the textual properties.

In the following, we assume that term dictionaries as well as quad tables are implemented with a b+-tree. The comparison is performed according to the following criteria: *Precision*, *Processing Complexity*, *Update Complexity* and *Query Expressiveness*. *Precision* evaluates if the system returns any false answers in the query result set. *Processing Complexity* evaluates the theoretical complexity for

**Table 1.** Summary of comparison among the four entity retrieval systems

| *Criteria* | *Node Index* | *Field Index* | *Quad Table* | *Semplore* |
|---|---|---|---|---|
| Precision (false positive) | No | Yes | No | Yes |
| Dictionary Lookup | $O(log(n))$ | $O(log(n*m))$ | $O(log(n))$ | $O(log(n))$ |
| Quad Lookup | $O(log(n))$ | $O(log(n*m))$ | $O(log(n)+log(k))$ | $O(log(n))$ |
| Join in Quad Lookup | Yes | No | No | No |
| Star Query Evaluation | Sub-Linear | Sub-Linear | $O(n)$ | $O(n*log(n))$ |
| Update Cost | $O(log(n))$ | $O(log(n*m))$ | $O(log(n)+log(k))$ | $O(log(n)+log(l))$ |
| Multiple Indices | No | No | Yes | Yes |
| Query Expressiveness | Star | Star | Graph | Tree |
| Full-Text | Yes | Yes (on literals) | No | Yes (on literals) |
| Multi-Valued Support | Yes | No | Yes | No |

processing a query (lookups, joins, etc.). *Update Complexity* evaluates the theoretical complexity of maintenance operations. *Query Expressiveness* indicates the type of queries supported. Table 1 summarises the comparison.

*Precision.* The field indexing scheme encodes the relation between predicates and terms in the index dictionary, but loses an important structural information: the distinction between literal objects. As a consequence, if the predicate is multi-valued, the field index may return false-positive results. Semplore suffers from a similar problem: it aggregates all the literal objects of an entity, disregarding the predicate, into a single bag of words. On the contrary, the node index and the quad table are able to distinguish distinct objects and do not produce wrong answers.

*Processing Complexity.* Since the field-based index encodes relations between predicate and terms in the dictionary, its dictionary may quickly become large when dealing with heterogeneous data. A dictionary lookup has a complexity of $O(log(n*m))$ where $n$ is the number of terms and $m$ the number of predicates. This overhead can have a significant impact on the query processing time. In contrast, the other systems stay with a term dictionary of size $n$.

To lookup a quad or triple pattern, the complexity of the node and field index is equal to the complexity of looking up a term in the dictionary. In contrast, the RDF databases should perform in addition a lookup on the quad table. The complexity is $O(log(n)+log(k))$ with $log(n)$ the complexity to lookup a term in the dictionary and $log(k)$, $k$ being the number of quads in the database, the complexity to lookup a quad in a quad table. In general, it is expected to have considerably more quads than terms, which can have a substantial impact on the query processing time for very large data collection. However, for quad patterns containing two or more terms, for example (?c,?s,p,o), the node index has to perform a merge-join between the posting lists of the two terms in order to check their relationships, but such joins can be performed on average in sub-linear time. On the contrary, the other indexes do not have to perform such joins. But, in the context of Semplore, access patterns where the predicate is not specified cause a full index scan.

For evaluating a star-shaped query (joining multiples quad patterns), each index has to perform a merge-join between the records of each quad patterns. Such join is linear with the number of records in the case of the quad table,

and sub-linear in the case of the node and field index if they use skip-lists. In contrast, Semplore has often to resort to expensive sort before merge-join operations.

*Update Complexity.* In terms of complexity of maintenance, in a b+-tree system the cost of insertion of one quad represents the cost of search of the leaf node (i.e., $O(log(n) + log(k))$), the cost of adding a leaf node if there is no available leaf node and the cost of rebalancing (overhead to keep the tree balanced). These operations become problematic with large indices and requires advanced optimizations [24] that in return cause degradations in query performance. In contrast, the cost of insertion for a node and field index is equal to the cost of a dictionary lookup as discussed in Sect. 4.2, which is $O(log(n))$ and $O(log(n * m))$ for the node index and the field index respectively. Furthermore, quad tables are specific to access patterns, hence multiple b+-tree indexes have to be updated. Concerning the size of the indexes, all of them are linear with the data.

Concerning Semplore, the original system could not perform updates or deletions of triples without a full re-indexing. The authors have recently [12] proposed an extension for incremental maintenance operations based on the *landmark* [25] technique but the update complexity remains substantial. The update cost is $O(log(n) + log(l))$ with $l$ the number of landmarks in the posting list. The fact that Semplore uses multiple indexes and landmarks considerably increase the update complexity. For example, index size and creation time reported in [12] are higher than for RDF-3X [5].

*Query Expressiveness.* In term of query expressiveness, RDF databases have been designed to answer complex graph-shaped queries which are a superset of the queries supported by the other systems. On the other hand, the other systems are especially designed to support natively full-text search which is not the case for quad table indexes. Node indexes provide more flexibility in term of full-text search since it enables keyword search on every parts of a quad. In addition, node indexes support set operations on nodes that give the ability to express set-valued queries on both URI and literal multi-valued properties.

Semplore supports relational tree-shaped queries but loses structural information since the relation between a resource and a literal is not indexed. Hence, it is not possible to restrict full-text search of a literal using a predicate, e.g. asking `(?s, <foaf:name>, "renaud")`.

## 6 Experimental Results

In this section, we compare the performance of SIREn against RDF databases (using quad tables over b+-tree indexes) based on some of the above criteria. We assess the space requirement, the index creation time and the query processing performance. The aim is to show the benefits of using a system like SIREn for web entity retrieval compared to common approaches based on RDF databases. While RDF databases are very efficient to answer complex queries,

we show that for the simpler task of entity retrieval, carefully designed systems can provide substantial benefits in term of scalability while sustaining fast query time.

The experimental setup is as follows. SIREn is implemented on top of Apache Lucene 2.4. The first RDF database is Sesame 2.0 with native backend (based on b+-tree), an open-source system which is commonly used as baseline for comparing quad store performances (e.g., in [5]). The second system is the state-of-the-art triple store RDF-3X [5]. We report that it is impossible for us to compare Semplore because at the time of the writing it is not being made available for this purpose. We also do not compare field-based index due to their query expressiveness limitations. In a previous publication [26], we reported experimental results showing the decrease of performance of field-based index compared to SIREn when the number of fields increases.

For the experiments, we use two datasets. The first one, called "Real-World" has been obtained by random sampling the content of the Sindice search engine. The real world dataset consists of 10M triples (approximately 2GB in size), and contains a balanced representation of triples coming from the Web of Data, e.g. RDF and Microformats data published online. The second dataset is the MIT Barton dataset that consists of 50M triples (approximately 6GB in size).

The machine that served for the experiment was equipped with 8GB ram, 2 quad core Intel processors running at 2.23 GHz, 7200 RPM SATA disk, Linux 2.6.24-19, Java 1.6.0.06 and GCC 4.2.4. All the following benchmarks are performed with cold-cache by flushing the kernel cache and by reloading the application after each query.

### 6.1 Index Size

The first experiment compares the index size of the three systems. The index size comprises the lexicon and the indices. Sesame is configured to create a single quad table (p,o,c,s). RDF-3X creates all the possible triple tables plus additional tables for query optimizations. SIREn creates a single inverted index.

The results are shown in Table 2(a). With respect to SIREn, Sesame exhibits at least a two-fold increase in index size on the real-world dataset and a four-fold increase on Barton. RDF-3X exhibits a four-fold increase in index size on the two datasets. With respect to the original dataset size, we observe that SIREn exhibits a index size ratio of 13-15%, whereas for Sesame and RDF-3X the ratio is approximately 50%. While the index size is linear with the size of the data collection for all the systems as discussed in Sect. 5, we can observe that the duplication of indices in RDF databases is causing a significant increase in index size compared to SIREn.

### 6.2 Insertion Time

In Table 2(b), we report the index creation time for the two datasets. For SIREn we report two cases: SIREn10 is the time to construct the index by batch of 10000 triples while SIREn100 is the time by batch of 100000 triples. Concerning

**Table 2.** Report on index size and indexing time

(a) Index size in MB per dataset and system

| | SIREn | Sesame | RDF-3X |
|---|---|---|---|
| Barton | 789 | 3400 | 3076 |
| Real-World | 296 | 799 | 1138 |

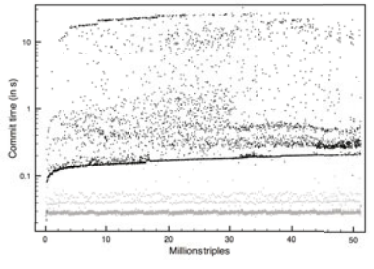
(b) Indexing time in minutes per dataset and system

| | SIREn10 | SIREn100 | Sesame | RDF-3X |
|---|---|---|---|---|
| Barton | 3 | 1.5 | 266 | 11 |
| Real-World | 1 | 0.5 | 47 | 3.6 |

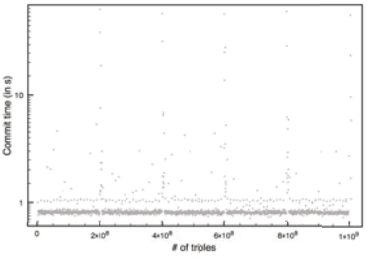
RDF-3X, it is important to notice that it does not support context, therefore it indexes triples and not quads, and that it does not support incremental indexing; RDF-3X needs the full dataset beforehand in order to construct the indexes in a single batch process, as opposed to Sesame which supports incremental updates. We can see from the results in Table 2(b) that SIREn is 50 to 100 times faster than Sesame and 3 to 6 times faster than RDF-3X.

In the next test, we plot the performance of SIREn and Sesame in an incremental update benchmark. The Fig. 3(a) shows the commit times for an incremental 10.000 triples batch on the two systems[6]. The graph is reported in logarithmic scale. While the absolute time is significant, the important result is the constant time exhibited by SIREn for incremental updates, as compared to Sesame performance which progressively decreases as the number of quads increases (as explained in Sect. 5).

In Fig. 3(b), the commit time of SIREn is plotted for a synthetic dataset constructed by replicating Barton 20 times so to reach 1 billion triples. The total index creation time is 31 minutes. We can notice that SIREn keeps a constant update time during the entire indexing. Outliers are due to periodic merges of the index segments. These results show that SIREn scales well with a large number of triples and provides significant improvement in terms of incremental update compared to other RDF databases.




(a) Plots showing the commit time every 10.000 triples during the index creation on Barton




(b) Plots showing the commit time every 500.000 triples during the index creation over one billion triples

**Fig. 3.** Dark dots are Sesame commit time records while gray dots are SIREn commit time records

[6] We omit the commit times for the Real-World dataset since the results were similar to the Barton dataset.

**Table 3.** Querying time in seconds

(a) Barton dataset

| | A1 | A2 | B1 | C1 | C2 | D1 | D2 | E |
|---|---|---|---|---|---|---|---|---|
| RDF-3X | 16.12 | 0.12 | 1.38 | 1.16 | 0.38 | 0.23 | 0.14 | X |
| SIREn | 2.79 | 0.02 | 1.33 | 1.71 | 0.95 | 0.36 | 0.03 | 0.96 |

(b) Real-World dataset

| | A1 | A2 | B1 | B2 | C1 | C2 | D1 | E |
|---|---|---|---|---|---|---|---|---|
| RDF-3X | 0.29 | 0.12 | 0.17 | 0.18 | 0.21 | 0.13 | 0.28 | X |
| SIREn | 0.23 | 0.03 | 0.04 | 0.05 | 0.09 | 0.08 | 0.16 | 0.53 |

(c) 10 Billion Triples dataset

| | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 |
|---|---|---|---|---|---|---|---|
| Time (s) | 0.75 | 1.3 | 1.4 | 0.5 | 1.5 | 1.6 | 4 |
| Hits | 7552 | 9344 | 3.5M | 57K | 448 | 8.2M | 20.7M |

### 6.3 Query Time Execution

For the query time execution benchmark, we created sets of queries with increasing complexity. The first set of queries (A*) consist of simple term lookups (URIs or literals). The second set of queries (B*) contains triple pattern lookups. The other sets consist of a combination of triple patterns using different set operators (intersection: C*, union: D*, exclusion: E). The queries are available at `http://siren.sindice.com`. For each query we average 50 query execution times without considering the final mapping between the result ids and their string values. The results are shown in Table 3(a) for Barton dataset and in Table 3(b) for Real-World dataset.

Since RDF-3X does not support native full-text search, we were unable to test queries involving this aspect. With respect to query E only SIREn was able to execute it since RDF-3X does not support the `bound` operator that is necessary to implement *exclusion* of triple patterns. With respect to Sesame, we decided not to include it in this test as during the experimentation phase we found that the results that we have obtained were consistently outperformed by RDF-3X.

The first observation is that on the Real-World dataset, SIREn performs significantly better, approximately 2 to 4 times, in 6 queries out of 7 while performing similarly in one, A1, a query which produces a very large amounts of results. In these queries and due to skewness of real-world data, SIREn are able to take advantage of its sub-linear merge-join by skipping unnecessary record comparisons.

On the Barton dataset, we notice however that for 3 queries out of 7 SIREn performs approximately 5 to 6 faster than RDF-3X, while resulting slower but comparable in 3 out of 7. In a particular query, C2, SIREn under-performs approximately 3 times. The explanation is that this query uses multiple triple access patterns that requires SIREn to perform more term lookups and merge-joins compared to RDF-3X and is therefore more expensive in term of disk I/O.

### 6.4 10 Billion Triples on a Single Machine

We evaluate SIREn scalability by indexing a dataset composed by 1 billion entities described in approximately 10 billion triples (one Terabyte of data). The dataset is derived from the billion triple challenge dataset[7]. To avoid hitting the

[7] Semantic Web Challenge: `http://challenge.semanticweb.org/`

limit of 2 billion entities due to the current implementation, we remove entities with only one or two triples and duplicate the remaining triples to reach 10 billion.

Since the dataset is different from the one in previous experiments, we use a different albeit comparable set of queries which is also provided at `http://siren.sindice.com`. The performance is given in the Table 3(c). Q1 to Q4 are property-object lookups using terms that are more or less frequent. In Q1 and Q2, we request for an infrequent property-object. The first query, while giving a result set of similar size, performs approximately two times better than Q2. The difference is that Q1 uses an infrequent predicate while Q2 a very frequent one, which in the latter case causes an overhead due to the merge-join. However, Q3 and Q4 use a very frequent property and, despite of the large increase of hits, the performance is similar or even better than Q2, which underlines that the complexity is linear with the length of the property posting list. Q5 performs a union between two infrequent property-object using a frequent property term. Again, we can observe the overhead caused by the merge-join between a frequent property and a term. However, Q6 and Q7 contain frequent properties and return a large number of results. The system scales linearly with the number of hits because the overhead of the join becomes less significant.

This scalability experiment shows that SIREn, even if there is a slight overhead when frequent properties are used in the query, scales well with a large number of triples and provides in all the cases reasonable query times, which makes it suitable for the web entity retrieval scenario.

## 7 Conclusion and Future Work

We presented SIREn, an entity retrieval system based on a node indexing scheme for searching the Web of Data. SIREn is designed for indexing and querying very large semi-structured datasets and offers constant time incremental updates and efficient entity lookup using semi-structural queries with full-text search capabilities. With respect to Database and Information Retrieval systems, SIREn positions itself somewhere in the middle as it allows semi-structural queries while retaining many desirable Information Retrieval features: single inverted index, effective caching, top-k queries and efficient index distribution over shards.

We demonstrated that a node indexing scheme provides a good compromise between query expressiveness, query processing time and update complexity. While such approach has an overhead during quad lookups due to additional joins, it provides fast enough answer time and scales well to a very large number of triples. Future works will concentrate on how to reduce the overhead of merge-joins in quad lookups, and on how to extend traditional weighting schemes to take into account RDF structural elements.

SIREn has been implemented and is in production at the core of the Sindice semantic search engine. At the time of the writing, SIREn serves over 60 million harvested web pages containing RDF or Microformats and answers several tens of thousands queries per day on a single machine.

## Acknowledgments

This material is based upon works supported by the European FP7 project *Okkam - Enabling a Web of Entities* (contract no. ICT-215032), and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

## References

1. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: A document-oriented lookup index for open linked data. International Journal of Metadata, Semantics and Ontologies 3(1) (2008)
2. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB Endowment, pp. 411–422 (2007)
3. Harth, A., Umbrich, J., Hogan, A., Decker, S.: YARS2: A Federated Repository for Querying Graph Structured Data from the Web. 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. 211–224. Springer, Heidelberg (2007)
4. Weiss, C., Karras, P., Bernstein, A.: Hexastore - sextuple indexing for semantic web data management. Proceedings of the VLDB Endowment 1(1), 1008–1019 (2008)
5. Neumann, T., Weikum, G.: RDF-3X - a RISC-style Engine for RDF. Proceedings of the VLDB Endowment 1(1), 647–659 (2008)
6. Baeza-Yates, R., Navarro, G.: Integrating contents and structure in text retrieval. SIGMOD Rec. 25(1), 67–79 (1996)
7. Walsh, N., Fernández, M., Malhotra, A., Nagy, M., Marsh, J.: XQuery 1.0 and XPath 2.0 data model (XDM). W3C recommendation, W3C (January 2007)
8. Gang, G., Chirkova, R.: Efficiently Querying Large XML Data Repositories: A Survey. IEEE Transactions on Knowledge and Data Engineering 19(10), 1381–1403 (2007)
9. Li, Q., Moon, B.: Indexing and Querying XML Data for Regular Path Expressions. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 361–370 (2001)
10. Haixun, W., Hao, H., Jun, Y., Yu, P., Yu, J.: Dual Labeling: Answering Graph Reachability Queries in Constant Time. In: Proceedings of the 22nd International Conference on Data Engineering, p. 75. IEEE, Los Alamitos (2006)
11. Su-Cheng, H., Chien-Sing, L.: Node Labeling Schemes in XML Query Optimization: A Survey and Trends. IETE Technical Review 26(2), 88 (2009)
12. Wang, H., Liu, Q., Penin, T., Fu, L., Zhang, L., Tran, T., Yu, Y., Pan, Y.: Semplore: A scalable IR approach to search the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 177–188 (2009)
13. Bast, H., Chitea, A., Suchanek, F., Weber, I.: ESTER: efficient search on text, entities, and relations. In: Proceedings of the 30th Annual International ACM SIGIR Conference, pp. 671–678. ACM, New York (2007)
14. Dong, X., Halevy, A.: Indexing dataspaces. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, p. 43 (2007)
15. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the semantic web. In: Proceedings of the 12th International Conference on World Wide Web, p. 544 (2003)

16. Beyer, K., Viglas, S.D., Tatarinov, I., Shanmugasundaram, J., Shekita, E., Zhang, C.: Storing and querying ordered XML using a relational database system. In: Proceedings of the 2002 ACM SIGMOD International Conference, pp. 204–215 (2002)
17. Sacks-davis, R., Dao, T., Thom, J.A., Zobel, J.: Indexing documents for queries on structure, content and attributes. In: Proceedings of International Symposium on Digital Media Information Base, November 1997, pp. 236–245. World Scientific, Singapore (1997)
18. Anh, V.N., Moffat, A.: Structured index organizations for high-throughput text querying. In: Crestani, F., Ferragina, P., Sanderson, M. (eds.) SPIRE 2006. LNCS, vol. 4209, pp. 304–315. Springer, Heidelberg (2006)
19. Witten, I.H., Moffat, A., Bell, T.C.: Managing gigabytes: Compressing and indexing documents and images, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco (1999)
20. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Computer Surveys 38(2), 6 (2006)
21. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
22. Moffat, A., Zobel, J.: Self-indexing inverted files for fast text retrieval. ACM Trans. Inf. Syst. 14(4), 349–379 (1996)
23. Graefe, G.: Query evaluation techniques for large databases. ACM Computing Surveys 25(2), 73 (1993)
24. Graefe, G.: B-tree indexes for high update rates. ACM SIGMOD Record 35(1), 39 (2006)
25. Lim, L., Wang, M., Padmanabhan, S., Vitter, J.S., Agarwal, R.: Dynamic maintenance of web indexes using landmarks. In: Proceedings of the 12th International Conference on World Wide Web, p. 102 (2003)
26. Delbru, R., Toupikov, N., Catasta, M., Fuller, R., Tummarello, G.: SIREn: Efficient Search on Semi- Structured Documents. In: Lucene in Action, 2nd edn. Manning Publications Co. (2009)

# Object Link Structure in the Semantic Web

Weiyi Ge[1], Jianfeng Chen[1], Wei Hu[2], and Yuzhong Qu[2]

[1] School of Computer Science and Engineering, Southeast University, China
[2] State Key Laboratory for Novel Software Technology, Nanjing University, China
wyge@seu.edu.cn, jferic@seu.edu.cn, whu@nju.edu.cn, yzqu@nju.edu.cn

**Abstract.** Lots of RDF data have been published in the Semantic Web. The RDF data model, together with the decentralized linkage nature of the Semantic Web, brings object link structure to the worldwide scope. Object links are critical to the Semantic Web and the macroscopic properties of object links are helpful for better understanding the current Data Web. In this paper, we propose a notion of object link graph (OLG) in the Semantic Web, and analyze the complex network structure of an OLG constructed from the latest dataset (FC09) collected by the Falcons search engine. We find that the OLG has the scale-free nature and the approximate effective diameter of the graph is small compared to its scale, which are also consistent with the experimental result based on our last year's dataset (FC08). The amount of RDF documents and objects by Falcons both doubled during the past year, but the object link graph remains the same density while the diameter is getting shrinking. We also repeat the complex network analysis on the two largest domain-specific subsets of FC09, namely Bio2RDF(FC09) and DBpedia(FC09). The results show that both Bio2RDF(FC09) and DBpedia(FC09) have low density in object links, which contribute to the low density of object links in FC09.

## 1 Introduction

In recent years, more and more RDF data have been published in the Web, and most of them are created to describe objects by using shared classes and properties. From Aug. 2008 to Sept. 2009, the number of RDF documents collected by the Falcons search engine [8] increases from 11.7M to 21.6M, with the number of RDF triples from 600 million to 2.9 billion, as well as the numbers of objects, classes and properties from 73.8M, 2.2M ,203K to 171.4M, 2.8M, 264K respectively.[1] As pointed out in [15], the Web is being extended with more and more RDF data sources and links between objects, even across data sources. The RDF data model, together with the decentralized linkage nature of the Semantic Web, brings object link structure to the worldwide scope, where objects are identified by URIs, and links are attributed to relational properties among objects.

The hypertext Web is considered to be a directed graph whose vertices correspond to Web pages and arcs correspond to hyperlinks between the pages, so

[1] http://ws.nju.edu.cn/falcons/statistics.jsp

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 257–271, 2010.

the page link graph in the hypertext Web is formed. We believe that the object link structure is important to the Semantic Web, as the Web page link structure to the hypertext Web.

Complex network analysis has been extensively performed on the page link graph to reveal the macroscopic properties of the hypertext Web [1,2,3,6,12]. Recently, graph analysis techniques have also been applied to the schema level of the Semantic Web, from single ontologies to a set of ontologies, even to the whole Semantic Web [9,13,16,21,24]. However, to the best of our knowledge, the macrostructure of the instance level of the Semantic Web has not yet been well studied. We argue that a simple link structure of object links reveals some useful macroscopic properties, which needs to be studied so that we can better understand the macrostructure of the current Data Web.

In this paper, we propose a notion of object link graph (OLG) in the Semantic Web, and analyze the complex network structure of an OLG constructed from the latest dataset (FC09) collected by Falcons until Sept. 2nd, 2009. We find that the OLG has the scale-free nature and the approximate effective diameter of the graph is small compared to its scale. Then, by comparing this OLG with another one constructed with our last year's dataset (FC08), we confirm our findings. Besides, the amount of RDF documents and objects in Falcons both doubled during the past year, but the object link graph remains the same density and its diameter is getting shrinking, which indicates a good evolution of the Data Web. We also repeat the complex network analysis of OLG on the two largest domain-specific subsets of FC09, namely Bio2RDF(FC09) and DBpedia(FC09). The results show that both of Bio2RDF(FC09) and DBpedia(FC09) have a low density in object links, which contributes to the low density of object links in FC09.

The remainder of this paper is organized as follows. Section 2 gives basic terminology used in this paper. Section 3 provides an overview of datasets used in the experiments and introduces our experimental methodology. Section 4 analyzes the OLG constructed from FC09. Section 5 investigates the evolution of the object link graph in the past year. In Section 6, we extract two domain-specific OLGs and compare their structures with OLG in FC09. Section 7 discusses related work. Section 8 concludes the paper with some observations and possible future work.

## 2 Terminologies

### 2.1 Graph

An undirected graph consists of a finite nonempty set of vertices $V$ and a set of edges $E$. An edge in $E$ is an unordered pair $(u, v)$ representing a connection between two vertices $u \in V$ and $v \in V$.

A connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by paths, and to which no more vertices or edges can be added while preserving its connectivity. The number of vertices in the connected component is called its size.

For each natural number $d$, let $g(d)$ denote the fraction of connected node pairs whose shortest connecting path has length at most d. The hop-plot for the graph is the set of pairs $(d, g(d))$, which denotes the cumulative distribution of distances between connected node pairs. We extend the hop-plot to a function defined over all positive real numbers by linearly interpolating between the points $(d, g(d))$ and $(d+1, g(d+1))$ for each $d$, and we define the effective diameter of the graph to be the value of $d$ at which this function achieves the value 0.9 [19].

A random variable $x$ is distributed according to a power law when its probability density function $p(x)$ is given by $p(x) = Ax^{-\gamma}$, where $A$ and $\gamma$ are positive constants, and $\gamma$ is called the power law exponent. A power law distribution plotted on a log-log scale is a line. A graph whose degree distribution follows a power law is scale-free.

### 2.2 Objects in the Semantic Web

An entity is a named resource identified by a URI in RDF data. An entity $e$ is regarded as a class (or a property) in an RDF document if the RDF graph encoded in the document entails the RDF triple $\langle e$, `rdf:type`, `rdfs:Class`$\rangle$ (or $\langle e$, `rdf:type`, `rdf:Property`$\rangle$), and it is regarded as an object if it is neither a class nor a property. In accordance with [9], we require the disjointedness of classes, properties, and objects, similar to OWL DL.

There may be more than one RDF documents in the Semantic Web that describe the same resource but give inconsistent description. For example, it is possible that a URI is stated to identify an object in one RDF document but to identify a class in another document. Inspired by [9], we developed the following heuristics to resolve the inconsistency. Firstly, we determine the identity of a URI by considering only its dereference document. If such document is not available, we will consider that a URI identifies an object only if no documents states that it identifies a class or property.

It is noteworthy that we are only interested in named resources but ignore blank nodes because a blank node cannot be directly referred outside the RDF graph it is defined by. Thus, cross-document links never happen to blank nodes so that they are not considered in the following Web-scale analysis. However, blank nodes may indirectly contribute to the Web-scale object link structure, which will be discussed in the next subsection.

### 2.3 Object Link Graph

An object link graph, denoted by $(\mathbb{O}, \mathbb{L})$, is an undirected graph, where $\mathbb{O}$ is the vertex set, each is identified by a unique URI to represent an object; $\mathbb{L}$ is the edge set, and each edge $(u, v)$ exists iff there is a sequence of $k$ triples $\{\langle a_i, p_i, b_i\rangle | 1 \leq i \leq k\}$, where $(a_1, b_k) \in \{(u, v), (v, u)\}$ and $b_i = a_{i+1}$, $b_i$ are all blank nodes for $1 \leq i \leq k-1$. Here, we do not simply assume the directionality of links between objects. According to this definition, blank nodes are not included in an object link graph, but blank nodes may still contribute to establishing links between objects.

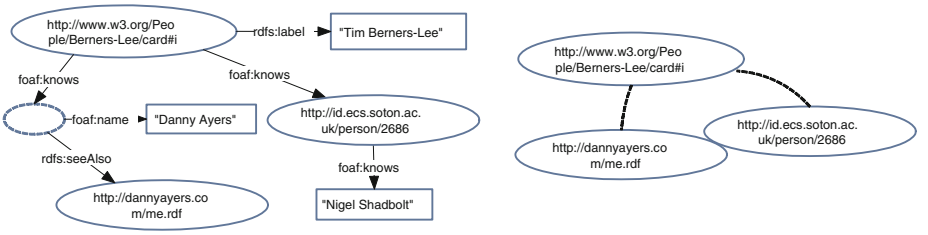



**Fig. 1.** An RDF graph (a) and its corresponding object link graph (b)

Fig. 1(a) shows a fragment of an RDF graph, which is derived from an RDF document identified by `http://www.w3.org/People/Berners-Lee/card`. From this RDF graph, we obtain an object link graph, as shown in Fig. 1(b). In particular, the link from `http://www.w3.org/People/Berners-Lee/card#i` to `http://dannyayers.com/me.rdf` is attributed to the fact that there is a simple path of length 2 via a blank node.

## 3 Datasets and Experimental Methodology

All the experimental results presented in this paper are obtained by analyzing datasets collected by the Falcons search engine. This section firstly gives an overview of the Falcons crawler. Then, datasets used in this paper are introduced. At last, experimental procedure is presented.

### 3.1 Crawler

To ensure the coverage of the Falcons crawler, we feed it a set of seed URIs of RDF documents from three sources. Firstly, we extract some keywords from the Open Directory Project,[2] and randomly combine them as queries in Swoogle[3] and Google (for "filetype:rdf" and "filetype:owl") to retrieve URIs of potential RDF documents. Secondly, as many personal RDF data are stored on several online repositories such as `pingthesemanticweb.com`, URIs of RDF documents from these repositories are added to the seed set. Thirdly, as the largest data source in the current Semantic Web, several entry-point URIs of the datasets published in the Linking Open Data project are manually submitted.

A parallel crawler is implemented to dereference URIs with content negotiation (ACCEPT application/rdf+xml) and download RDF/XML documents. Besides, this crawler follows the robots.txt protocol with the HTTP header field User-Agent setting to "Falconsbot". We do not use any filter rules or domain limits, as we want to get a relatively complete sample of the current Data Web.

[2] `http://www.dmoz.org/`
[3] A representative Semantic Web search engine, `http://swoogle.umbc.edu/`

### 3.2 Datasets

We exported two snapshots of datasets from the Falcons crawler in August 26th, 2008 and September 2nd, 2009. The 2008's dataset contains 11,719,608 RDF documents, and the 2009's dataset contains 21,639,337 RDF documents.

```
<foaf:Person rdf:nodeID="me">
  <foaf:nick>Carlita</foaf:nick>
  <foaf:knows>
    <foaf:Person>
      <foaf:nick>Gabriela</foaf:nick>
      <rdfs:seeAlso rdf:resource="http://api.hi5.com/rest/profile/foaf/212231607"/>
    </foaf:Person>
  </foaf:knows>
</foaf:Person>
```

**Fig. 2.** A fragment taken from an RDF document in hi5.com

After inspecting the top-50 domains[4] with most RDF documents, we find some social networking sites have the similar publishing style, which use blank nodes to identify objects instead of URIs. Fig. 2 shows a fragment taken from an RDF document in `hi5.com`. In the fragment, a person with nickname "Carlita" (blank node) knows another person "Gabriela" (blank node), and Gabriela has an FOAF document `http://api.hi5.com/rest/profile/foaf/212231607`. Here, blank nodes are used to identify persons. It is stressed that this publishing way seems incompatible with the Linked Data principles [5], since it does not use URIs to identify things and cannot interlink objects across data sources. Most of RDF documents from `hi5.com`, `mybloglog.com`, `buzznet.com`, `liveinternet.ru`, `deadjournal.com` and `rambler.ru` have the similar publishing style. So, we refine the datasets by excluding those RDF documents that come from the above six domains, and the refined datasets are called **FC08 (Falcons Crawl 2008)** and **FC09 (Falcons Crawl 2009)** respectively.

We find that FC08 has 11,286,186 RDF documents coming from 10,216 domains and FC09 has 18,646,011 RDF documents from 21,171 domains. That is, the FC09 dataset is doubled in the quantity of documents and the diversity of domains as compared with FC08. The top-10 domains w.r.t. the number of RDF documents are listed in Table 1.

From FC08, we identify 64,974,423 objects by using the heuristics described in Section 2.2. Among all the objects identified, 63,795,076 ones (98.18%) are identified by the HTTP URIs, and they are hosted by 621,619 domains. From FC09, we identify 110,507,074 objects, and 108,842,826 ones (98.49%) are identified by the HTTP URIs hosted by 698,753 domains. That is, the number of

[4] The domain name of the URL is the substring of the URL's hostname, without subdomain names. For example, `fu-berlin.de` is the domain name of `http://www4.wiwiss.fu-berlin.de/dblp/terms.rdf`

**Table 1.** Top-10 domains w.r.t. the number of RDF documents

(a) FC08

| Domain | #documents |
|---|---|
| bio2rdf.org | 6,636,748 |
| dbpedia.org | 2,577,748 |
| opiumfield.com | 415,534 |
| geonames.org | 359,684 |
| w3.org | 211,388 |
| l3s.de | 156,786 |
| fu-berlin.de | 129,187 |
| bibsonomy.org | 113,650 |
| rkbexplorer.com | 110,524 |
| uniprot.org | 98,159 |

(b) FC09

| Domain | #documents |
|---|---|
| bio2rdf.org | 7,685,644 |
| dbpedia.org | 3,712,453 |
| geonames.org | 1,497,089 |
| opiumfield.com | 785,223 |
| l3s.de | 719,138 |
| dbtune.org | 577,634 |
| fu-berlin.de | 564,848 |
| openlinksw.com | 498,047 |
| bibsonomy.org | 398,665 |
| w3.org | 392,516 |

**Table 2.** Top-10 domains w.r.t. the number of objects

(a) FC08

| Domain | #objects |
|---|---|
| bio2rdf.org | 28,276,823 |
| dbpedia.org | 6,671,120 |
| wikipedia.org | 3,955,286 |
| flickr.com | 2,501,768 |
| fu-berlin.de | 2,282,677 |
| uniprot.org | 1,931,325 |
| l3s.de | 1,842,994 |
| uni-trier.de | 1,464,458 |
| musicbrainz.org | 1,332,336 |
| opiumfield.com | 1,299,949 |

(b) FC09

| Domain | #objects |
|---|---|
| bio2rdf.org | 33,667,558 |
| dbpedia.org | 7,645,474 |
| opiumfield.com | 6,560,575 |
| flickr.com | 6,156,454 |
| last.fm | 5,639,584 |
| l3s.de | 5,404,294 |
| wikipedia.org | 4,195,842 |
| fu-berlin.de | 3,972,447 |
| dbtune.org | 3,659,400 |
| geonames.org | 2,803,359 |

objects is also two times more than the one of last year, and the distribution of these objects is more diverse. The top-10 domains w.r.t. the number of objects are listed in Table 2.

From Table 1(b), we find that bio2rdf.org and dbpedia.org have most RDF documents (41.22% and 19.91% respectively). Besides, most objects are also distributed in these two domains (30.47% and 6.92%), see Table 2(b). So, we export all RDF documents in bio2rdf.org and dbpedia.org to form two domain-specific datasets, namely **Bio2RDF(FC09)** and **DBpedia(FC09)**.

In Bio2RDF(FC09), we identify 36,036,254 objects derived from 7,685,644 documents. These objects are distributed in 1,720 domains. Besides, from 3,712,453 documents in DBpedia(FC09), we identify 17,414,639 objects, which are distributed in 505,132 domains. In a sense, this indicates that DBpedia is a linking-hub for interconnecting objects from various data sources.

### 3.3 Experimental Methodology

Three experiments are designed to explore object link structures in the Semantic Web.

Firstly, to understand the macroscopic characteristics of the object link structure in the current Semantic Web, we analyze the object link graph in FC09. We compute the average degree of this graph, which reflects the density of graph in some sense. Then, we analyze the distribution of isolated vertices and investigate reasons why there are so many isolated vertices. Degree distribution of the graph is depicted to reveal whether the graph has the scale-free natural. Connected components of the graph are computed to reveal whether objects in the Semantic Web are well interlinked. Effective diameter is approximately calculated to seek to understand within how many certain numbers of hops that most connected object pairs can be interlinked.

Secondly, in order to confirm our findings and to find the evolution of OLG's structure, we repeat the complex network analysis on FC08. We compare the network characteristics of two objects link graphs.

In the third experiment, we analyze two domain-specific OLGs constructed from Bio2RDF(FC09) and DBpedia(FC09) in the same way, and compare the experimental results with the OLG in FC09.

## 4 Object Link Graph in the Current Semantic Web

In this section, we analysis the degree and connectivity of the OLG from FC09.

### 4.1 Degree

This OLG contains 110,507,074 vertices and 190,201,590 edges, the average degree of the graph is 3.44, and the highest degree is 1,181,411. Broder, et al. [6] report that the web graph constructed from the traditional hypertext Web takes an average in(out)-degree of 7.86, which means its degree as an undirected graph maybe nearly doubled. Compared to the traditional hypertext Web, the small average degree of the OLG indicates a sparse link structure.

By inspecting the degree of each vertex, we find that 4,746,095 (4.29%) objects have no links to others. That is, objects do not link to other objects through a sequence of triples connected by blank nodes (see Section 2.3 for details about constructions of OLG). Most of these isolated objects are distributed in 10 domains, as listed in Table 3. In particularly, the `bio2rdf.org` domain takes 43.6% of all isolated objects found so far. After a close investigation on the RDF documents that mention these isolated objects, we find some typical patterns causing the isolation: a) Objects are not connected to any other objects, but might be connected to classes or properties, e.g. in their type declarations; b) Objects are not connected to any other objects, but connected to literals, e.g. with only literal description about these objects; and c) Objects are connected with blank nodes.

**Table 3.** Top-10 domains with most isolated vertices in the OLG from FC09

| Domain | #isolated vertices |
|---|---|
| bio2rdf.org | 2,071,885 |
| last.fm | 487,137 |
| yahoofs.com | 271,828 |
| nbii.gov | 199,979 |
| friendfeed.com | 160,656 |
| opencyc.org | 82,059 |
| umbc.edu | 79,392 |
| zitgist.com | 76,798 |
| mpii.de | 57,754 |
| rossia.org | 55,689 |

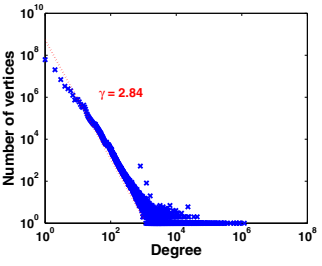



**Fig. 3.** Degree distribution of the OLG from FC09

The degree distribution is illustrated in Fig. 3. This distribution follows a power law, indicating the scale-free nature of the graph. The power law exponent of degree distributions of the OLG is 2.84, here we use a maximum-likelihood method to fit the exponent [7]. Since the vast majority of vertices in the scale-free network are with small degree, the graph is fault tolerant in the face of random failures; But if a few major high degree vertices (hubs) are removed, it may turn into a set of rather isolated graphs. Besides, anther important characteristic of scale-free network is the clustering coefficient distribution, which decreases as the vertex degree increase. That is, the low-degree vertices belong to very dense sub-graphs and in the meantime hubs connect these sub-graphs together. Since hubs connect sub-graphs each other and the number of hubs is few, we can feed them as seeds in the search engine as a good placement. In-degree and out-degree distributions of subsets of the the traditional Web follow power law with exponents 2.1 and 2.38-2.72, respectively [3,18,6]. OLG's power law exponent is a little larger than ones of the traditional Web.

### 4.2 Connectivity

**Connected Component.** Connected component analysis on the OLG shows that the largest CC has 97,391,271 (88.13%) objects. That is, 88.13% of objects in FC09 are reachable to each other by following RDF links. This value is close to the one of the traditional Web (91%) reported in [6], so the connectivity of the OLG is not bad. Except the largest CC and the trivial ones (with only one vertex), there are also 813,975 connected components. Objects in these connected components mainly come from only one single domain (62.12%), and objects in the most one only come from 1,000 domains.

**Effective Diameter.** Computing the diameter of a large graph is very costly. In the case of the OLG, we cannot compute the exact diameter, instead, we apply the Approximate Neighborhood Function (ANF) approach [20] to estimate the

effective diameter for the OLG.[5] The approximate effective diameter of the OLG is 11.53, which is a small one compared to the scale of the graph. Broder, et al. [6] find that the average path length in the traditional Web (when a path exists) is around 6.83 if all edges are considered to be undirected, which is much less than the effective diameter of our OLG. That is, there is an average longer length of the shortest path between connected objects than that of the traditional Web..

## 5 Structural Evolution of Object Link Graph

As mentioned above, in Section 3.2, the amounts of RDF documents and objects in Falcons Crawl get doubled in the past year. So, we are naturally concern about the structural change brought by the increasing RDF data. In this section, we analyze the object link graph constructed from FC08, and compare it with the OLG from FC09.

**Table 4.** Top-10 domains with most isolated vertices in the OLG from FC08

| Domain | #isolated vertices |
|---|---|
| bio2rdf.org | 2,078,204 |
| li.ru | 677,779 |
| dbpedia.org | 250,764 |
| llnwd.net | 91,914 |
| truesense.net | 91,265 |
| cyc.com | 43,568 |
| dbtune.org | 40,253 |
| mcdonaldbradley.com | 39,072 |
| rkbexplorer.com | 37,019 |
| klab.lv | 36,158 |

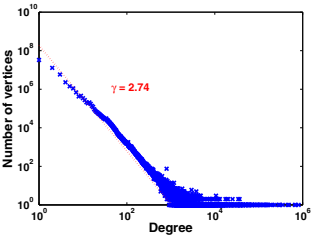



**Fig. 4.** Degree distribution of the OLG from FC08

### 5.1 Degree

From 11,286,186 RDF documents in FC08, we identify 64,974,423 objects and construct an object link graph with 64,974,423 vertices and 109,373,275 edges. The average degree of the OLG is 3.37, which is a little smaller than the one (3.44) of OLG from FC09. The slightly increasing average degree indicates that the object link graph become less sparser during the past year.

We find that 3,987,843 (6.14%) isolated vertices are distributed in 16,353 domains. Table 4 lists the top-10 domains containing most isolated objects. Besides, comparing with isolated vertices of the OLG from FC09, we find that some isolated vertices in FC08 disappear in FC09, while some new isolated vertices emerge in FC09. The percentage of isolated vertices declines in the past year (from 6.14% to 4.29%), which shows a trend that objects are getting interlinked.

[5] In our experiment, $k$ is set to 32, which ensures that ANF achieves less than 10% errors.

The degree distribution of the OLG from FC08 is depicted in Fig. 4, which also follows a power law, with the power law exponent 2.74. This exponent is similar to the one of the OLG from FC09 (2.84).

### 5.2 Connectivity

**Connected Component.** Connected component analysis on the OLG in FC08 shows that the largest connected component takes 57,122,054 (87.91%) objects and 104,675,519 (95.70%) edges. Except the largest connected component and the trivial ones, there are also 686,071 connected components. Objects in these connected components mainly come from only one single domain (46.72%), and objects in the most one also come from 1,000 domains (the same as FC09).

The proportion of the size of the largest connected component in the object link graph is a little increasing during the last year (from 87.91% to 88.13%), which indicates that the trend of connectivity is to be better slightly.

**Effective Diameter.** The approximate effective diameter of the OLG from FC08 is 12.28, which is larger than the one from FC09 (11.53). As this index is a highly-accurate approximation, it is likely that the diameter of the OLG shrinks in the past year.

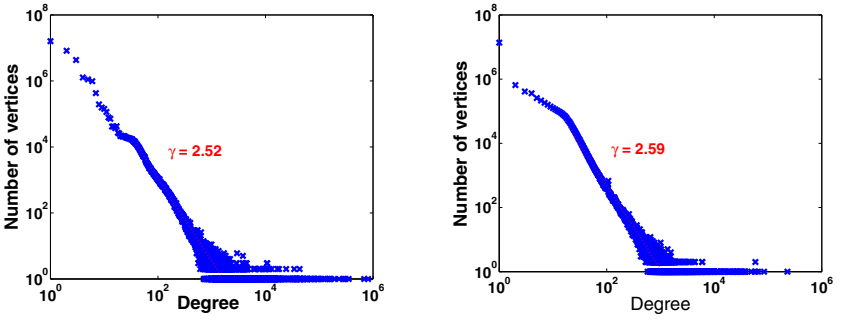
## 6 Domain-Specific Object Link Structures in the Semantic Web

To verify whether the scale-free nature and small effective diameter also hold for some domain-specific OLGs, and to find the reason why OLG has a sparse structure compared to the traditional Web, we analyze two OLGs constructed from the two largest data sources in FC09, namely Bio2RDF(FC09) and DBpedia(FC09). That is, we construct each OLG from a set of dereference documents from some certain domain (`bio2rdf.org` or `dbpedia.org`). Here, the two OLGs are called the Bio2RDF OLG and the DBpedia OLG respectively.

### 6.1 Degree

The average degrees of the Bio2RDF OLG and the DBpedia OLG are 3.56 and 3.49 respectively, which indicate a low density in object links in both Bio2RDF(FC09) and DBpedia(FC09). As the homepage of DBpedia[6] points out, DBpedia contains "807,000 links to images and 3,840,000 links to external web pages; 4,878,100 external links into other RDF datasets". In fact, DBpedia functions as a linking-hub for interconnecting various data sources to form the main part of the current Data Web. So its divergent structure results in a low density. As Bio2RDF [4] tries to make documents from public bioinformatics databases, such as Kegg, PDB, MGI, HGNC and several of NCBIs databases, available in

[6] `http://dbpedia.org/`




(a) Bio2rdf. (b) DBpedia.

**Fig. 5.** Degree distributions of domain-specific OLGs

RDF format, it contains many small data sources. These small data sources are figured out in the middle-lower part of the Linking Open Data graph.[7] So it is likely the reason that the Bio2RDF OLG has a low density.

Because Bio2rdf(FC09) and DBpedia(FC09) contain 61.13% RDF documents of the total in FC09, it is reasonable to believe that the low density of both Bio2rdf(FC09) and DBpedia(FC09) has great influence on the low density of the object links in FC09.

The number of isolated vertices in the Bio2rdf OLG and the DBpedia OLG are 2,073,663 (5.75%) and 7,193 (0.04%) respectively. Degree distributions of these two OLGs are depicted in Fig. 5. Both two distributions approximately follow power law with exponent 2.52 and 2.59 respectively.

### 6.2 Connectivity

**Connected Component.** The largest connected component of the Bio2rdf OLG takes 32,354,360 (89.78%) objects, and the one of DBpedia is 16,499,512 (94.75%). We notice that largest connected components of these two OLGs are both contained in the largest connected component of OLG from FC09, which indicates most objects in these two datasets are interlinked. Besides, both OLGs have a better connectivity than the OLG from FC09 (88.13%).

**Effective Diameter.** The approximate effective diameters of the Bio2rdf OLG and the DBpedia OLG are 7.38 and 7.81 respectively. Both of them are small as compared to the scale of the graph.

## 7 Related Work

Graph analysis has been extensively performed on page link graph to the hypertext Web. Albert, et al. [3] analyzed the distributions of incoming and outgoing

[7] http://www4.wiwiss.fu-berlin.de/bizer/pub/lod-datasets_2009-07-14.html

links between HTML documents in the World Wide Web, and observed power law tails. Adamic [1] showed that the largest strongly connected component of the graph of sites in the Web is a small world. He also counted how many links the sites received from other sites, and found that the distribution of links also follows a power law [2]. Broder, et al. [6] confirmed power law distributions of in- and out-degree. They studied the directed and undirected connected components of the Web, and showed that power laws also arise in the distribution of sizes of those connected components. They revealed that the true structure of the web graph must be somewhat subtler than a "small world" phenomenon in which a browser can pass from any web page to any other with a few clicks. Further, they figured out a bow-tie structure as the macroscopic structure of the Web. Even recently, researcher were still studying various datasets to investigate topological properties of the Web graph, such as bipartite cores, PageRank values, and some correlations [12].

Graph analysis techniques have also been applied to single ontologies or a set of ontologies. Hoser, et al. [16] illustrated the benefits of applying social network analysis to ontologies by measuring SWRC and SUMO ontologies. They interpreted an ontology as a graph: classes and properties became vertices in the graph; an arc was added from a class to its superclass, or from a property to its domain, range, and superproperty. They discussed how different notions of centrality (degree, betweenness, eigenvector, etc.) describe the core content and structure of an ontology, and compared ontologies in size, scope, etc. Zhang [24] studied NCI-Ontology, Full-Galen, and other five ontologies, and discovered that the degree distributions of these entity networks fit power laws well. Theoharis, et al. [21] analyzed graph features of 250 ontologies. For each ontology, they constructed a property graph and a class subsumption graph. The property graph is a directed graph whose vertices correspond to classes and literal types, and whose arcs point from the domain of a property to its range. They found that the majority of ontologies with a significant number of properties approximate a power law for total-degree distribution, and each ontology has a few focal classes that have numerous properties and subclasses. Gil, et al. [13] combined ontologies from the DAML Ontology Library into a single RDF graph, which included 56,592 vertices and 131,130 arcs. They observed that the graph is a small world with an average path length 4.37, and the cumulative degree distribution follows a power law with exponent $\gamma = 1.485$. Recently, Cheng and Qu [9] studied the graph structures of dependence between concepts and between vocabularies. The graphs analyzed in the experiments are constructed from a large dataset that contains more than 1 million terms in more than 3 thousand vocabularies. The results characterize the current status of schemas in the Semantic Web in many aspects, including degree distributions, reachability, and connectivity.

The Semantic Web has also been analyzed from other aspects. Hausenblas, et al. [14] attempted to answering the question: *What is the size of the Semantic Web?* through analyzing the Linking Open Data dataset. Wang, et al. [23] surveyed nearly 1,300 ontologies and analyzed their expressiveness, the use of OWL constructs, the shape of class hierarchy, etc. Tummarello, et al. [22] found that

**Table 5.** Summary of the characteristics of the OLGs from the four different datasets

| dataset | #vertices | #edges | average degree | isolated vertices | $\gamma$ | largest connected component | effective diameter |
|---|---|---|---|---|---|---|---|
| FC08 | 64,974,423 | 109,373,275 | 3.37 | 6.14% | 2.74 | 87.91% | 12.28 |
| FC09 | 110,507,074 | 190,201,590 | 3.44 | 4.29% | 2.84 | 88.13% | 11.53 |
| Bio2RDF(FC09) | 36,036,245 | 64,207,033 | 3.56 | 5.75% | 2.52 | 89.78% | 7.38 |
| DBpedia(FC09) | 17,414,639 | 30,415,886 | 3.49 | 0.04% | 2.59 | 94.75% | 7.81 |

the distribution (reuse) of URIs over documents follows a power law. Ding and Finin [10] collected 1,448,504 RDF documents and focused on the distribution of documents over hosts and the sizes of documents. They measured the complexity of terms by counting the number of RDF triples used to define them, and measured the instance space by counting the meta-usages of terms. Power laws were observed in both experiments. Ding, et al. [11] collected over 1.5 million of FOAF documents, and analyzed the empirical usage of namespace and properties in the FOAF community. The authors selected about 7,000 FOAF documents containing 50,559 instances of foaf:Person, and analyzed the social networks induced by those FOAF documents and revealed some interesting patterns. To the best of our knowledge, the macrostructure of the instance level of the Semantic Web has not yet been well studied.

## 8 Conclusion

In summary, the main contributions of this paper are as follows.

1. A notion of object link graph is proposed to model the Semantic Web structure at the instance level. Based on this notion, we construct an object link graph from a large dataset, namely FC09. We show that the object link graph has the scale-free nature and the effective diameter of the graph is 11.53, which is a small one compared to the scale of the graph.
2. We repeat the complex network analysis on another dataset, namely FC08. The results confirm that the object link graph is scale-free and its effective diameter is a small one compared to the graph's scale. Comparing the object link structure of FC09 with the one of FC08, we observe that the object link graph is not becoming sparser and its diameter is likely to shrink in the past year, though the amounts of documents and objects both doubled, which indicates a good evolution of the Data Web.
3. We repeat the complex network analysis on the two largest domain-specific subsets of FC09, namely Bio2RDF(FC09) and DBpedia(FC09). The results show that both Bio2RDF(FC09) and DBpedia(FC09) have low density in object links, which has great influence on the density of OLG from FC09.

The resulting characteristics of these graphs are summarized in Table 5. The dataset, analyzed graphs, and statistical results are available online.[8] These

[8] http://ws.nju.edu.cn/olg/

experimental results presented in this paper can indicate the current state of the object link structure in the Semantic Web.

From these results, we obtain some observations. Firstly, the object link graph inherits some characteristics of the hypertext link structure, such as the scale-free nature. Secondly, a low average degree makes the object link graph different from the page link graph. In fact, a low average degree indicates the lack of links between objects in the Semantic Web, making the object link graph be more sparse, compared with the page link graph. We believe that publishing more object links online will make the Semantic Web better. Besides, more effort is needed to investigate the URI alias phenomena [17] and study the impact of the URI alias phenomenon on the object link structure.

Hopefully, the macroscopic properties of object link structure provided by this paper can help people better understand the current Web of data. In future work, more experiments are deserved to detail the big picture of the object link graph in the Semantic Web, and the dynamic model of the object link graph in the Semantic Web needs to be investigated. Besides, the interaction behavior between instance level and schema level in the Semantic Web is an interesting topic to be studied.

## Acknowledgments

The work is supported in part by the NSFC under Grant 60973024, and in part by the NSFC under Grant 60773106. We are grateful to anonymous reviewers for their precious comments.

## References

1. Adamic, L.A.: The Small World Web. Research and Advanced Technology for Digital Libraries 1696, 443–452 (1999)
2. Adamic, L.A., Huberman, B.A.: Power-Law Distribution of the World Wide Web. Science 287(5461), 2115a (2000)
3. Albert, R., Jeong, H., Barabasi, A.L.: The Diameter of the World Wide Web. Nature 401, 130–131 (1999)
4. Belleau, F., Nolin, M., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: Towards a Mashup to Build Bioinformatics Knowledge Systems. Journal of Biomedical Informatics 41(5), 706–716 (2008)
5. Berners-Lee, T.: Linked Data - Design Issues (2006), http://www.w3.org/DesignIssues/LinkedData.html
6. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the web. Computer Networks 33(1-6), 309–320 (2000)
7. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-law Distributions in Empirical Data. SIAM Review 51, 661–703 (2009)
8. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: Proceedings of the 17th International Conference on World Wide Web, pp. 1101–1102 (2008)

9. Cheng, G., Qu, Y.: Term dependence on the semantic Web. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 665–680. Springer, Heidelberg (2008)
10. Ding, L., Finin, T.: Characterizing the Semantic Web on the Web. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 242–257. Springer, Heidelberg (2006)
11. Ding, L., Zhou, L., Finin, T., Joshi, A.: How the Semantic Web is Being Used: An Analysis of FOAF Documents. In: Proc. of 38th Annual Hawaii International Conference on System Sciences, p. 113c (2005)
12. Donato, D., Laura, L., Leonardi, S., Millozzi, S.: The Web as a graph: How far we are. ACM Transactions on Internet Technology (TOIT) 7(1) (2007)
13. Gil, R., García, R., Delgado, J.: Measuring the Semantic Web. AIS SIGSEMIS Bulletin, 69–72 (2004)
14. Hausenblas, M., Halb, W., Raimond, Y., Heath, T.: What is the Size of the Semantic Web? In: Proc. of I-Semnatics (2008)
15. Hendler, J., Shadbolt, N., Hall, W., Berners-Lee, T., Weitzner, D.: Web science: an interdisciplinary approach to understanding the web. Communications of the ACM 51(7), 60–69 (2008)
16. Hoser, B., Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Semantic Network Analysis of Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 514–529. Springer, Heidelberg (2006)
17. Jacobs, I., Walsh, N. (eds.): Architecture of the world wide web, volume one. W3C Recommendation December 15 (2004), http://www.w3.org/TR/webarch/
18. Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.: Extracting Large-scale Knowledge Bases from the Web. In: International Conference on Very Large Data Bases, pp. 639–650 (1999)
19. Leskovec, J., Kleinbery, J., Faloutsos, C.: Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In: Proc. of International Conference on Knowledge Discovery and Data Mining, pp. 177–187 (2005)
20. Palmer, C.R., Gibbons, P.B., Faloutsos, C.: ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs. In: Proc. of International Conference on Knowledge Discovery and Data Mining, pp. 81–90 (2002)
21. Theoharis, Y., Tzitzikas, Y., Kotzinos, D., Christophides, V.: On Graph Features of Semantic Web Schemas. IEEE Transactions on Knowledge and Data Engineering 20(5), 692–702 (2008)
22. Tummarello, G., Delbru, R., Oren, E.: Sindice.com:Weaving the Open Linked Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
23. Wang, T.D., Parsia, B., Hendler, J.: A Survey of the Web Ontology Landscape. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 682–694. Springer, Heidelberg (2006)
24. Zhang, H.: The Scale-Free Nature of Semantic Web Ontology. In: Proc. of the 17th International Conference on World Wide Web, pp. 1047–1048 (2008) (poster)

# ExpLOD: Summary-Based Exploration of Interlinking and RDF Usage in the Linked Open Data Cloud

Shahan Khatchadourian and Mariano P. Consens

University of Toronto
shahan@cs.toronto.edu, consens@cs.toronto.edu

**Abstract.** Publishing interlinked RDF datasets as links between data items identified using dereferenceable URIs on the web brings forward a number of issues. A key challenge is to understand the data, the schema, and the interlinks that are actually used both within and across linked datasets. Understanding actual *RDF usage* is critical in the increasingly common situations where terms from different vocabularies are mixed. In this paper we describe a tool, ExpLOD, that supports exploring summaries of RDF usage and interlinking among datasets from the Linked Open Data cloud. ExpLOD's summaries are based on a novel mechanism that combines text labels and bisimulation contractions. The labels assigned to RDF graphs are hierarchical, enabling summarization at different granularities. The bisimulation contractions are applied to subgraphs defined via queries, providing for summarization of arbitrary large or small graph neighbourhoods. Also, ExpLOD can generate SPARQL queries from a summary. Experimental results, using several collections from the Linked Open Data cloud, compare the two summary creation approaches implemented by ExpLOD (graph-based vs. SPARQL-based).

## 1 Introduction

As the web continues to evolve it incorporates new ways of publishing and interacting with information beyond simply linking documents. A promising area for extending the web is the addition of Linked Data [6] to provide a simple mechanism for sharing structured data via the creation of Resource Description Framework (RDF) [15] links between data items using dereferenceable URIs. The emergence of a web of linked open data (LOD) is being promoted by the Linking Open Data community project and is fostering the availability of many open interlinked datasets. Many interlinked datasets have been contributed to what has been referred to as the *LOD Cloud*. Some notable examples are structured datasets such as DBpedia [4] (extracted from Wikipedia), RKB Explorer [16], YAGO [25], and LinkedMDB [17]. These datasets are placed online by the community, fostering collaborative linkage of structured knowledge.

The web of interlinked data is closely intertwined with the existing web. As a dataset like DBpedia illustrates, structured data items have RDF links referencing classic web pages and the reverse is also possible. The increased presence

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 272–287, 2010.

of abundant linked data (see [18]) enables novel applications and sophisticated mashups. Linked data can be searched using semantic web search engines which leverage both page and data links, such as URI-based semantic web search engines Swoogle [13] and Sindice [26], or term-based search engines such as Falcons [8], or by web search engines returning enhanced results such as Yahoo! SearchMonkey [1].

A key challenge is to understand the data, the schema, and the interlinks that are actually used both within and across linked datasets. Using an RDF browser to explore linked data can be a tedious and time-consuming exercise for large datasets. A trial-and-error approach on a schema-conformant dataset means testing for structures permitted by the schema which becomes complicated if the dataset does not use the full schema or uses multiple schemas. In the LOD cloud, this is even more pronounced due to the possible heterogeneity of ontologies used in each dataset, so another approach is needed. Inspiring early work on *RDF usage* presented in [12] enumerates several RDF usages involving classes, predicates, schema resources, and data resources. Understanding actual RDF usage is critical for developers in the increasingly common situations where terms from many different RDFS and OWL vocabularies are mixed. This is because developers need to know a dataset's structure in order to contribute interlinks between datasets, or for scenarios involving resources described within a dataset or across datasets. We propose that, instead of examining RDF usages separately, summaries that show how different RDF usages interact with one another can be used to describe an interlinked dataset.

**Related Work.** Describing and understanding large collections is enabled by summaries. One type of summary which does this by grouping common substructures is a *structural summary*. Prior work has shown the usability of XML path summaries for a variety of scenarios within semi-structured XML collections such as XPath query answering [9] and information retrieval [3]. Recently, structural summaries have been proposed in the context of RDF data in [21], but for frequency estimation applied to query evaluation (also the most common use for summaries in the context of XML), but not for describing usage in a flexible manner as done in this work. Our work is based on the work in structural summaries that has been developed in the context of semi-structured data (XML in particular) over the last decade. We extend and generalize our earlier work in XML summaries [10] in significant ways, in particular, we do not rely on the tree nature of XML data or on acyclicity assumptions that do not hold for RDF datasets.

Several methods have been proposed to enumerate the interlinks between datasets. [20] describes a system that allows a user to visualize the namespaces and classes present in a dataset and the linkages between them. Semantic sitemaps [11] use *slices* as a way to specify how interlinked datasets may be structured, such as indicating that each document contains a single resource description. A slice can also be expressed as a Concise Bounded Description (CBDs) [24] which may traverse a fixed length path to or from the described resource. An alternative to CBDs is the Minimum Self-Contained Graph (MSG)

[27] which includes a resource's blank node closure. The voiD vocabulary [2] portrays a restricted view into the number and type of predicates that contribute to the actual interlinks between datasets. Silk [29] is a mechanism that can create interlinks between datasets declaratively and can use heterogeneous vocabularies independently of any schema.

In this paper, we address the specific challenge of creating RDF summaries that describe structures within and among interlinked datasets in a flexible manner. In particular, our work can answer the following question - *how do data, metadata, and interlinking contribute to the structure of RDF datasets from the LOD cloud?* The contributions of our work are: (1) A novel bisimulation contraction framework to explore the LOD cloud; (2) A tool, Explod, that creates RDF usage summaries based on bisimulation contractions; (3) A flexible mechanism to compute RDF usage summaries at different granularities using text labels and different subgraphs of RDF data (that can be specified using a query); (4) A way to generate a SPARQL query that returns RDF data having a particular bisimulation contraction; (5) A SPARQL-based approach to computing RDF usage summaries; and (6) A performance comparison of the SPARQL-based and graph-based approaches.

This paper is structured as follows. In Section 2, we define the summaries that ExpLOD creates. Section 3 we showcase how ExpLOD can be used to explore the LOD cloud. In Section 4, the two summary creation approaches implemented in ExpLOD are described, and a performance comparison is provided in Section 5. We conclude in Section 6.

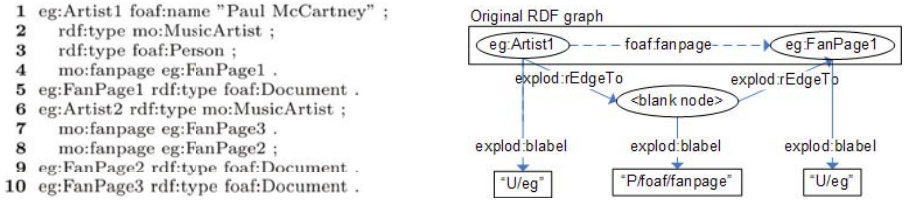
## 2 ExpLOD Bisimulation Contraction Framework

This section defines the summaries that are generated by ExpLOD. In Section 2.1, a labeled graph (with unlabeled edges) is constructed from an RDF dataset using hierarchical labels. In Section 2.2, we describe how summaries are defined by applying bisimulation contractions to neighbourhoods of the labeled graph.

### 2.1 Applying Bisimulation Labels to RDF

ExpLOD's RDF data model is based on quads. A quad consists of: a context, such as a graph name URI [7]; and a triple, a 3-tuple consisting of a *subject* URI, a *predicate* URI, and an *object* that can either be a URI or any valid XML data type [5]. A URI is represented as in [15], a pair of labels called the *prefixed name* composed of a *prefix label* (also known as a namespace prefix in XML) and a *local part* that are concatenated with a colon.

Figure 1(a) shows the triples of a sample dataset. The dataset gives information about two music artists, their fan pages, and one artist's name using the FOAF ontology (with prefix label *foaf*) to describe people and documents, and the Music Ontology (*mo*) vocabulary to describe music artists and their work. The triple on line 1, *(eg:Artist1, foaf:name, 'Paul McCartney')*, indicates that the name of the web resource given by the URI *eg:Artist1* is the string value




(a) RDF dataset (b) ExpLOD graph

**Fig. 1.** Simple example

*'Paul McCartney'*. The object of a triple whose predicate is *rdf:type* is a *class* and the subject is an *instance* of the class. The triple on line 6, *(eg:Artist2, rdf:type, mo:MusicArtist)*, instantiates *eg:Artist2* as a music artist. The context of the statements is excluded for brevity.

Summaries of RDF data are computed over a graph using each node's *bisimulation label* (BL); we describe the construction of the labeled graph in more detail later in this section. The BL of each node in the labeled graph is based on *RDF usage*. In this work, we define an RDF resource's usage that is dependent on its *neighbourhood*, a subgraph of the labeled graph. Different neighbourhoods of an RDF resource describes its usage in different ways. We consider four RDF usages that describe the interaction of data and meta-data: (i) *class instantiation*, the number of instances that are typed as a particular class; (ii) *predicate instantiation*, the number of times a predicate is used to describe all instances; (iii) *class usage*, the *sets* of classes to which instance belongs; and (iv) *predicate usage*, the *sets* of predicates used to describe an instance. For example, in Figure 1(a), the class usage of *eg:Artist2* is the singleton *mo:MusicArtist*, which is different from the class usage of *eg:Artist1*, the set of classes {*mo:MusicArtist*, *foaf:Person*}. The predicate usage of the two music artists also differs because the predicate *foaf:name* is used to describe *eg:Artist1*, but not *eg:Artist2*.

We use hierarchical BLs, similar to dimension hierarchies common to most OLAP data models [28]. In the current implementation we are using string literals for BL values due to limitations in creating URIs in SPARQL; however, it would make sense use dereferenceable URIs for BLs since this would allow us to fetch additional information about the bisimulation labeling hierarchies that are used. BL hierarchies allow describing datasets at a range of granularities. The BL hierarchy to describe RDF usage is of the following form: RDF usage prefix, context, URI prefix label, URI local part; parts of the hierarchy are concatenated with a forward slash ('/'). The RDF usage prefix is 'P' for predicates, 'C' for classes, 'I' for instances, and 'L' for literals. The BL of a URI used as a predicate or class includes the full hierarchy. The BL of an instance URI excludes the local part of its prefixed name, and the BL of literals shows the content instead of the prefix label and local part (since it does not have a URI). In the above example, the BL of instances *eg:Artist1* and *eg:FanPage1* is 'I/eg', and the BL of the predicate *foaf:fanpage* is 'P/foaf/fanpage'.

```
1  # predicates and edges
2    CONSTRUCT {
3      ?s explod:rEdgeTo [
4        explod:blabel concat("P/", str(?g), "/", namespace(?p), "/", localname(?p)) ;
5        explod:rEdgeTo ?o . ] .
6    } WHERE { GRAPH ?g { ?s ?p ?o. FILTER (!isLiteral(?o))}.}
7  # classes
8    CONSTRUCT { ?c explod:blabel concat("C/", namespace(?c), "/", localname(?c)) }
9    WHERE { ?u rdf:type ?c . }
10 # instances (everything else except literals)
11   CONSTRUCT { ?s explod:blabel concat("I/",namespace(?s))}
12   WHERE { OPTIONAL { ?s explod:blabel ?blabel }
13     FILTER (!bound(?blabel) && !isLiteral(?s)) . }
```

**Fig. 2.** SPARQL queries that construct a labeled graph

We now describe the construction of the labeled graph; the computation of bisimulation contractions using the labeled graph is shown in Section 2.2. A *labeled graph*, $R = (V, E, Label, \lambda)$, is a graph with nodes $V$, unlabeled directed edges $E \subseteq V \times V$, a set of BLs *Label*, and a function $\lambda$ that assigns each node in $V$ a BL in *Label*. $\langle v_1, v_2 \rangle \in E$ denotes a directed edge from $v_1$ to $v_2$, where $v_1, v_2 \in V$. BLs are assigned based on an adaptation of the common RDF graph representation, in which edges between nodes are labeled with a predicate, into a graph with labeled nodes and unlabeled edges. A node is created in the labeled graph for each predicate as a way to convert the predicate-labeled edge to an unlabeled edge. The creation of a labeled graph from the triple on line 4 of Figure 1(a) is depicted in Figure 1(b) above. The *ExpLOD RDF graph* is a set of RDF statements (within ExpLOD's application context) which describes the labeled graph. Oval nodes represent nodes in the ExpLOD RDF graph and each node's BL values is in a square node. Notice that representing the predicate with a node (a blank node in this case) is a form of reification [22]. Solid edges represent statements in the ExpLOD RDF graph, and dashed edges represent statements in the original RDF graph. The underlying label graph can be recovered from the ExpLOD graph.

Labeled graphs can be constructed using an ordered set of SPARQL CONSTRUCT queries that assign the BLs to RDF data; 3 example queries are shown in Figure 2. Each query generates a BL using the extension function *concat* (similar to the *fn:concat* function in XQuery) to concatenate strings. The BL of each node is ascribed as the string value of an *explod:blabel* predicate. A *explod:rEdgeTo* RDF property is used to represent directed edges between nodes.

### 2.2 Bisimulation Contractions

Bisimulation contractions are computed on sets of neighbourhoods. A *labeled bisimulation* between two labeled graphs $X$, $Y$ is a symmetric equivalence relation $\approx \subseteq V_X \times V_Y$ such that, for $x \in V_X$, $y \in V_Y$: if $x \approx y$, then $\lambda(x) = \lambda(y)$; if $x \approx y$, and $\langle x, x' \rangle \in E_X$, then $\lambda(x) = \lambda(y)$, $\langle y, y' \rangle \in E_Y$, and $x' \approx y'$. A partition of a set of nodes is a set of pairwise disjoint subsets whose union is the set of nodes; each subset is called a *partition block*. The *coarsest stable partition* (CSP) of a set of nodes is a partition such that, for each pair of partition blocks $B_1$, $B_2$, either: each node in $B_1$ has an edge to a node in $B_2$, or no nodes in $B_1$


(a) Nested View (b) ExpLOD view

**Fig. 3.** Example: Class usage summary

have an edge to a node in $B_2$. According to [14], computing the CSP produces the bisimulation contraction, also known as the *maximum bisimulation*. That is, all nodes in a partition block are equivalent according to bisimulation, and no pair of nodes from different partition blocks are equivalent.

An RDF summary is a graph with labeled nodes and unlabeled edges. To distinguish between a labeled graph node and a summary node, we refer to a summary node as a *summary block* (or just 'block'). There is a block in the RDF summary for each partition block in the CSP, and each block in the summary has an *extent* that represent its associated partition block. There is an edge in the summary from block $B_1$ to block $B_2$ if each node in the extent of block $B_1$ has an edge to a node in the extent of block $B_2$.

Figure 3(a) is a class usage summary of the RDF datasets in Figure 1(a). It is a CSP of blocks whose extents contain nodes having the same bisimulation contraction. There is an instance block (a block whose label starts with 'I/eg', the common BL of the bisimilar nodes in its extent) that contains an instance that is both music artist and a person, i.e., *eg:Artist1*. There is also an instance block that is just a music artist (*eg:Artist2*), and there is an instance block containing 3 instances in its extent that are documents (the three fanpage instances). The statement identifier is included for reference to the statements in Figure 1(a). The same class usage summary that ExpLOD displays is shown in Figure 3(b), with the extent size shown between parentheses, and a unique block identifier for instance blocks within square brackets.

We have shown the construction of each node's BL based on RDF usage, and how bisimulation contractions are computed based on BLs. In the next section, we showcase how using different BL labels and RDF usage neighbourhoods can be used to explore the LOD cloud.

## 3 Exploring the LOD Cloud Using ExpLOD

In this section we showcase examples of RDF usage summaries to describe datasets in the LOD cloud. Section 3.1 shows the flexibility of modifying the BL scheme to obtain a coarser or more detailed summary. Section 3.2 demonstrates how a larger RDF usage neighbourhood can be employed to describe interlinked datasets. Finally, Section 3.3 describes one of the main features of

ExpLOD, which is the ability to return to the user, for any one block in a summary, the SPARQL query that returns the resources grouped in the extent of a block.

### 3.1 Class and Predicate Usage

A class and predicate RDF usage summary shown in Figure 4 is used to describe Jamendo, a dataset from the LOD cloud containing information about music artists and their productions. Our goal is to understand how tracks and records are described by examining the interaction of class and predicate instantiation, and class and predicate usage, of records and tracks in Jamendo.

Class instantiation is reported in the extent size of each class block. For example, the extent size of class block 5 (with BL prefix 'C/') shows that there are 5,786 instances that are typed as records. The class usage of instance block 1430 is the singleton *mo:Track* since there is only one such path to the class block with BL 'C/mo/Track' . Predicate instantiation is reported in the extent size of each predicate block (with BL prefix 'P/'). For example, the predicate *mo:license* (with BL 'P/mo/license') has been instantiated 45,634 times. The predicate usage of all tracks in the extent of instance block 1430 is the set of predicates {*mo:license*, *mo:track_number*, *mo:available_as*, *dc:title*, *rdf:type*}, visible by the edges from the instance block to each of those predicate blocks. Notice that each instance block has a unique class and predicate usage. Amongst records, there is variation in the extent size of predicate usages - the extent size of instance block 1361 is 4,509 compared to the extent size of instance block 1388 that contains only 1 record instance.

An RDF usage summary with many blocks can be difficult to understand. Using a reduced portion of the BL hierarchy can sometimes produce a summary with fewer blocks, and as many blocks as before (in the worst case). For example, excluding the local part of each predicate's BL groups predicates by their namespace, reducing the 8 instance blocks in Figure 4 to 3 in Figure 5.

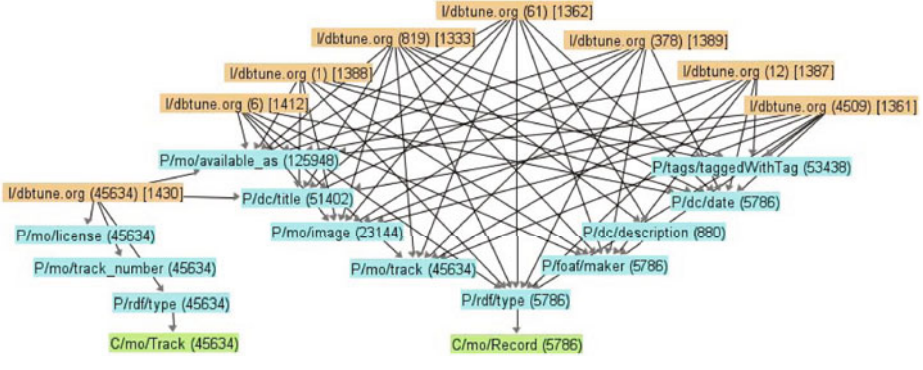



**Fig. 4.** Jamendo: RDF usage summary of class and predicate instantiation, and class and predicate usage

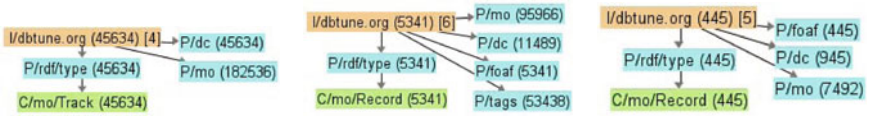



**Fig. 5.** Jamendo: predicate usage summary of records and tracks grouped by namespace

### 3.2 Interlinking

Two interlinked RDF datasets in the LOD cloud may contain information about the same real-world entity, but it is possible that each dataset uses its own unique URI to represent it. A triple with an *owl:sameAs* predicate captures the information that the subject and object URIs refer to the same real-world entity, referred to as *URI-equivalence*; such statements can be found readily within many datasets in the LOD cloud. A resource's description in one dataset may not match the description of its URI-equivalent resource in another dataset.

An *interlink usage* summary, which we define as a bisimulation contraction of URI-equivalent resources, is used to understand each dataset's contribution to a real-world entity's description. In our work, an interlink usage neighbourhood is defined as the subgraph that includes the nodes and edges on the path linking the subject of a triple whose predicate is *owl:sameAs* to the object node. Additionally, the incoming edge of each *owl:sameAs* predicate node is reversed, both in the labeled graph and the ExpLOD graph, so that it points to the subject. Since we are interested in the description of instances, we consider only statements in which the subject and object are instances. In addition to creating a summary solely based on interlink usage neighbourhoods, considering class and predicate usage neighbourhoods within each dataset provides additional information about URI-equivalent resource descriptions.

Figure 6 shows the blocks of instances that are interlinked, have the same class and predicate usage in one dataset, and have the same class usage in a second dataset. This example covers two datasets, Southampton and Newcastle, from the RKB Explorer collection. This image shows that there are 57 instances in Southampton with the same class and predicate usage whose equivalent instances in Newcastle have the same class usage. It also shows that both namespaces use the class *aktors:Publication-Reference* to the class usage of these resources and that the predicate usage of the instances from Southampton includes the predicates *aktors:portal#has-title* and *aktors:portal#has-publication-reference*. Thus,

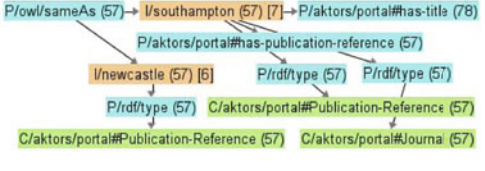



**Fig. 6.** Southampton and Newcastle: Interlink usage summary

we have shown how interlinking in the LOD cloud can be explored by computing bisimulation contractions of interlink usage neighbourhoods.

### 3.3 SPARQL Block-Extent Queries

To find which instances are in a block's extent, the extent needs to be materialized. A *SPARQL block-extent query* (SBE) is used to materialize the extent of a block. An SBE is a SELECT query that returns the instances in a block extent by using negation-by-failure (with keywords OPTIONAL, FILTER, and !BOUND) as a way to retrieve instances with a specific bisimulation contraction. The bisimulation contraction is specified in an SBE by including the portions of the neighbourhood that should be matched and using negation-by-failure to exclude the parts of the neighbourhood should not be present in the neighbourhood. Futher details of how the SBE graph pattern is generated is described in Section 4.2. The SBE of Figure 7 returns the 45,634 instances in the extent of instance block 1430 in Figure 4. The SBE specifies the class usage and predicate usage on lines 2 and 3, respectively; these are the required portions of the class usage neighbourhood. Negation-by-failure is used to exclude portions of the class usage on lines 4-6, and similarly on lines 7-9 for excluding portions of the predicate usage.

```
1 SELECT ?U WHERE {
2   ?U rdf:type mo:Track ;
3     mo:track_number [] ; dc:title [] ; mo:license [] ; mo:available_as [] .
4   OPTIONAL { ?U rdf:type ?o0 .
5     FILTER (?o0 != mo:Track) .
6   } FILTER ( !bound(?o0) ) .
7   OPTIONAL { ?U ?prop ?o .
8     FILTER (?prop != mo:available_as && ?prop != mo:track_number && ?prop != mo:license && ?prop
      != dc:title && ?prop != rdf:type ) .
9   } FILTER ( !bound(?prop) ) . }
```

**Fig. 7.** SBE to materialize extent of block 1430 in Figure 4

## 4 ExpLOD Implementation

In this section, we describe the two techniques used by ExpLOD to create RDF summaries based on bisimulation contractions. Section 4.1 references an existing partition-refinement approach. Section 4.2 focuses on a main contribution of this work, a SPARQL-based approach to compute bisimulation contractions from SPARQL-generated BLs and neighbourhoods of interest specified in SPARQL.

### 4.1 Partition-Refinement Approach

The PRAIG implementation is covered only briefly here, details are given in [19]. Neighbourhoods to summarize are obtained using an intersection-automaton construction. The set of neighbourhoods is obtained from intersecting the automaton representation of a path regular expression that selects neighbourhoods in the labeled graph, and the automaton representation of a labeled graph. From

[14], the CSP representing an equivalence relation based on bisimulation contractions is obtained by using a partition-refinement algorithm. PRAIG uses a partition refinement algorithm from [23] to create the CSP. A side-effect of partition-refinement is that each block's extent is *materialized*, that is, pointers are created originating from each block and directed to the resources in its extent. The next section shows how it is possible to create a partition of blocks without materializing the extents, while retaining the ability to materialize the extent of individual blocks at a later time.

### 4.2 SPARQL Approach

In order to compute the blocks of the summary using SPARQL, the BLs need to be known before-hand. A *SPARQL label retrieval query* (SLRQ) is used to retrieve the distinct labels of a neighbourhood over which bisimulation contractions are to be computed.

We define an SLRQ as a SPARQL graph pattern as in [15], with variables and terms (such as URIs and literals) that returns subgraphs with each variable bound to a value. The SLRQ of Figure 8(a) picks the class usage neighbourhoods, i.e., it returns neighbourhoods where instances are connected to classes via *rdf:type* predicates. The neighbourhood is based on the RDF representation of the labeled graph shown in Figure 1(b). Lines 3 and 4 specify a path from a node $n1$ to a node $n3$ via a node $n2$, where edges are represented by the predicate *explod:rEdgeTo*. Each node's BL is bound to a variable in lines 5-7, and the labels of these variables are then filtered in lines 8-10 according to a class usage neighbourhood, where an instance (with BL prefix 'I/') has an edge in the labeled graph to a predicate *rdf:type* (having a BL of 'P/rdf/type') followed by an edge to a class (with BL prefix 'C/'). Thus, an SLRQ returns the BLs that are present in a neighbourhood of interest, and next, we will show how to use the BLs returned by an SLRQ to compute the bisimulation contraction of that neighbourhood, also in SPARQL.

A SPARQL block query (SB) is a way of returning the blocks of an RDF summary without materializing any block extents. An SB query obtains the unique bisimulation contractions using the labels from the SLRQ and a *SPARQL block pattern* (SBP), a graph pattern that selects each possible bisimulation contraction for the neighbourhoods of interest. The SBP is generated from an SLRQ as follows. The SLRQ specifies a neighbourhood of interest and, after obtaining the

```
1  SELECT DISTINCT ?n1_bl ?n2_bl ?n3_bl
2  WHERE {
3    ?n1 explod:rEdgeTo ?n2 .
4    ?n2 explod:rEdgeTo ?n3 .
5    ?n1 explod:blabel ?n1_bl .
6    ?n2 explod:blabel ?n2_bl .
7    ?n3 explod:blabel ?n3_bl .
8    FILTER (regex(?n1_bl, "I/.*")) .
9    FILTER (?n2_bl = "P/rdf/type") .
10   FILTER (regex(?n3_bl, "C/.*")) . }
```

(a) SLRQ

```
1  SELECT DISTINCT count(?s) ?c0 ?c1 ?c2 WHERE {
2  OPTIONAL { ?s rdf:type ?c0 FILTER(?c0 = foaf:Person)}
3  OPTIONAL { ?s rdf:type ?c1 FILTER(?c1 = mo:MusicArtist)}
4  OPTIONAL { ?s rdf:type ?c2 FILTER(?c2 = foaf:Document)}}
```

(b) SPARQL block query

**Fig. 8.** Example: SPARQL approach to compute class usage summary

BL values that are possible for each possible block, an OPTIONAL code block is created for each of its possible BL values. The query then displays the blocks that are instantiated by each bisimulation contraction. It is also possible to use a shortcut that uses the URI directly instead of the BL, such for classes. The SBP is also used to generate an SBE such as the example shown in Section 3.3.

Figure 8(b) shows an example SB in which a shortcut is used to group instances from the RDF dataset in Figure 1(a) having the same bisimulation contraction of their class usage neighbourhoods. Line 2 specifies the neighbourhood of the SBP. Each possible distinct class BL is specified within an OPTIONAL code block (lines 2 through 4) and the keyword DISTINCT on line 1 causes the SB to return only unique bisimulation contractions based on the optional code blocks that are matched. That is, the set of possible BL values that were returned by the SLRQ for node $n3$ in Figure 8(b) are *mo:MusicArtist*, *foaf:Person*, and *foaf:Document*. The BL values for nodes $n1$ and $n2$ are excluded for brevity since their BL values are singletons, 'I/eg' and 'P/rdf/type', respectively, and their exclusion does not modify the results of using the DISTINCT keyword. Notice that the BL value of $n2$ is specified directly within the class usage neighbourhoods of lines 2 through 4 as *rdf:type*. Optionally, since some SPARQL implementations support aggregation, it is also possible to return a block's extent size as part of the SB, and this is shown on line 1 of the query as *count(?s)*. The set of results returned by running the SB would be {*(1, foaf:Person, mo:MusicArtist, null), (1, null, mo:MusicArtist, null), (3, null, null, foaf:Document)*} and this matches the class usage summary visualized in Figure 3(b).

Thus far, we have shown how each resource is assigned a BL that captures each resource's RDF usage. Then, using the BLs, bisimulation contractions are computed using one of two implementations in ExpLOD, a graph-based approach, or a SPARQL-based approach that takes advantage of SPARQL-generated BLs. We have also shown how using different BLs can help to control the size of a summary (to reduce its complexity), and how considering different neighbourhoods of interest can help analyze interlinking in the LOD cloud. This completes our study of the RDF usage summaries generated using our ExpLOD tool. In the next section, we compare the performance of graph-based and SPARQL-based approaches that are implemented in ExpLOD.

## 5 Experimental Study

In this section, we compare the performance of the two techniques used by ExpLOD and covered in the preceding section.

ExpLOD creates summaries based on the following input: the dataset, the BL scheme, and the neighbourhoods to consider. The summaries produced can be viewed and explored in an interactive graphical environment and they can also be exported in a variety of formats (including RDF). ExpLOD is a Java application developed within the Eclipse environment with support for plug-ins. Custom code was developed for all graph and automaton data structures used by PRAIG and the Jena toolkit (`jena.sourceforge.net`) is used to manage RDF data. Explod can also invoke the Virtuoso RDF store (`www.openlinksw.com/virtuoso`).

In the experiments, we use Virtuoso's default configuration, except that checkpoint logs are disabled.

Section 5.1 describes the datasets considered in the experiments. Section 5.2 gives a performance evaluation of summary creation using ExpLOD. All the experiments used a dedicated AMD Opteron 2.4 GHz server hosting a Windows XP virtual machine with 2 GB of RAM (of which 1.5 GB was assigned to the PRAIG JVM).

### 5.1 Datasets

The first 7 columns of Table 1 shows the datasets from the LOD cloud that are considered for the experiments. The table columns show the following information about each dataset: its name, the number of triples it contains, and the number of instance (whose BL starts with 'I/'), class ('C/'), and predicate ('P/') nodes in its labeled graph. The next two columns show the number of instance blocks, blocks whose label starts with 'I/', that are in the class usage summary ('#S1'), and in the class and predicate usage summary ('#S2'). The instance blocks are counted as we are interested in exploring the interaction of classes and predicates in instance descriptions, and the number of instance blocks reported is from PRAIG, except in the case of LinkedCT and LinkedMDB. These two dataset did not fit in virtual-memory and SB was used instead; however, due to query limitations in Virtuoso's SPARQL engine (described further in Section 5.2), the exact number of instance blocks in their class and predicate usage summary (column '#S2') could not be determined exactly and is a lower-bound (shown in italics).

These datasets were chosen as they vary in the amount and type of information they describe. The largest dataset (counting the number of triples) examined by PRAIG is Jamendo with over 1 million triples and its labeled graph contains 410,784 instance, 11 class, and 25 predicate nodes. The largest dataset examined using SB and SBE is LinkedCT that contains over 7 million statements and whose labeled graph contains 860,510 instance, 13 class, and 91 predicate nodes. Notice that, even though LinkedCT is the dataset with the most number of state-

**Table 1.** LOD dataset graphs and performance (in ms) of PRAIG, SB, and SBE

| | | | | | | | Class usage | | | Class and predicate usage | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Triples | I/ | C/ | P/ | #S1 | #S2 | PRAIG | SB | SBE | PRAIG | SB | SBE |
| Dailymed | 116,992 | 9,623 | 2 | 26 | 4 | 440 | 7,269 | 178 | 168 | 184,169 | 307,913 | 5,874 |
| Diseasome | 69,639 | 20,165 | 2 | 18 | 5 | 13 | 4,556 | 248 | 241 | 8,847 | 148,608 | 655 |
| EPSRC | 340,064 | 40,913 | 13 | 33 | 14 | 31 | 27,413 | 6,123 | 1,877 | 47,412 | 56,962 | 4,644 |
| ESWC2007 | 7,256 | 1,315 | 35 | 48 | 35 | 74 | 1,844 | 867 | 63 | 14,981 | 2,113 | 915 |
| ESWC2008 | 4,628 | 1,052 | 23 | 66 | 25 | 53 | 662 | 220 | 42 | 6,075 | 126,111 | 637 |
| Jamendo | 1,047,837 | 410,784 | 11 | 25 | 12 | 35 | 158,600 | 30,652 | 16,876 | 165,625 | 6,868,293 | 1,700,131 |
| LinkedCT | 7,025,488 | 860,510 | 13 | 91 | 13 | *52* | — | 91,715 | 34,972 | — | *361,075* | 1,314,477 |
| LinkedMDB | 3,579,594 | 757,878 | 41 | 148 | 39 | *49* | — | 218,091 | 19,631 | — | *452,232* | 150,998 |
| Magnatune | 169,004 | 41,199 | 7 | 24 | 8 | 8 | 12,769 | 2,028 | 1,026 | 20,875 | 9,633 | 2,973 |
| Peel | 271,369 | 76,894 | 9 | 25 | 10 | 28 | 23,638 | 5,523 | 3,338 | 35,806 | 30,460 | 31,852 |
| Southampton | 219,019 | 30,721 | 21 | 31 | 45 | 151 | 17,894 | 6,583 | 1,241 | 54,516 | 50,628 | 62,068 |

ments, there are datasets with more classes or predicates, such as LinkedMDB with 41 classes and 148 predicates. Despite the large number of variations possible in the interaction of classes and predicates in these large dataset, it is the Dailymed dataset that contains the most number of instance blocks in its class and predicate usage summary (it has 440) when it only has 2 classes. Jamendo's C-summary has 12 instance blocks and its class usage summary and 35 instance blocks in its class and predicate usage summary, and LinkedCT has 39 instance blocks in its class usage summary and 49 instance blocks in its class and predicate usage summary. Furthermore, aside from Dailymed and Diseasome, none of the datasets have as many instance blocks in their class usage summary as their combinatorial potential, showing limited variation in how classes and predicates are used.

### 5.2 Performance Results

To show the efficiency of the two techniques shown in the previous section, we conduct a performance evaluation. Performance is measured as the time taken to compute an RDF usage summary, assuming that the labeled graph has already been created. Performance of computing class usage summaries, and class and predicate usage is explored for datasets enumerated in Section 5.1.

PRAIG performance includes materializing extents (as it is part of its algorithm). We report the time of using SBE to materialize the extent of all instance blocks computed by SB. In Virtuoso, computing SB queries containing more than 7 distinct predicates sometimes took hours to complete. Additionally, SB queries could not contain more than 63 distinct predicates, disallowing generating SB queries for datasets such as LinkedMDB since it contains 148 predicates. To reduce the number of predicates an SB contained, an SB query was created for each class usage and a predicate was included in an SB only if it described at least one instance of that class usage. The workaround was used with all datasets (even if they did not violate the predicate limitation) and was implemented as follows. First, all distinct class labels present in the labeled graph we used to generate an SB to compute a class usage summary. Then, for each distinct class usage, the distinct predicate labels that described some instance having that class usage were used to generate a class and predicate usage SB whose class usage was bound. All the blocks returned from the predicate usage SBs form the final class and predicate usage summary. Despite this workaround, some modified SBs still contained more than 7 predicates, and these modified SBs are not included in the SB and SBE performance times reported.

The last 6 columns of Table 1 shows the performance (in ms) of PRAIG, SB, and SBE. PRAIG performance times are the average of 3 runs. SB and SBE performance times are the average of 8 runs. Although 10 runs were performed for SB and SBE, the first run of SB was sometimes twice as slow as the remaining runs so 8 runs were average, excluding the first run and a performance outlier,

the time with the highest absolute deviation from the average of 9 runs (that excludes the first run). PRAIG required around 7 s to compute the class usage summary of Dailymed, 178 ms using SB, and 168 ms to materialize the 4 instance blocks using SBE.

PRAIG took around 184 s to compute the class and predicate usage summary of Dailymed, 307 s using SB, and almost 6 s to materialize the 440 instance blocks using SBE. Because PRAIG cannot be used for datasets larger than virtual-memory, it was not used to compute the RDF usage summaries for the LinkedMDB and LinkedCT datasets, and is shown as '—' in the table.

We now discuss our findings on the datasets considered. SB is faster than PRAIG for computing class usage summaries. PRAIG consistently computes class and predicate usage summaries faster than SB; however, SB can be used on datasets that do not fit in virtual-memory (with some limitations). Although using SB to compute class and predicate usage summaries for Peel and Southampton is comparable to PRAIG's performance, adding SBE's time more than doubles the SPARQL-based approach's time. The cost of SBE relative to SB ranges from less than 1 percent for ESWC2008 to more than 3 times for LinkedCT.

Since no PRAIG run deviated more than a second from the times reported, the standard deviation of PRAIG's performance is not reported. By excluding the first run, the relative standard deviation of computing class usage summaries with SB decreased from around 60% to 14%, with a further reduction to around 7% by excluding the performance outlier. The relative standard deviation of computing class and predicate usage summaries with SB decreased from around 36% to 9%, then to 5% without the performance outlier. The relative standard deviation of computing SBE for the class usage summary changed from 60% to 34% then to 19%. The relative standard deviation of computing SBE for the class and predicate usage summary changed from 10% to 9% then to 4%.

## 6 Conclusion

RDF summaries constructed using labeled bisimulation contractions is a way to reveal the unique structure in RDF datasets from the LOD cloud. Different BL hierarchies allow flexible RDF summary construction such as to examine existing and construct new RDF usages. Performance of the graph-based approach shows a distinct advantage over SPARQL-based approach with the limitation that it works only with datasets that fit in main-memory. The SPARQL-based approach can be used for large datasets with some limitations showing that there is potential to explore the implementation of labeled bisimulation contractions in existing systems. Future work also involves moving the graph-based approach to a distributed environment.

**Acknowledgements.** The first author was financially supported by an IBM CAS Fellowship. We also thank the reviewers for their detailed comments.

## References


1. Yahoo! SearchMonkey, http://developer.yahoo.com/searchmonkey (last Accessed: December 21, 2009)
2. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets - On the Design and Usage of voiD, the Vocabulary of Interlinked Datasets. In: LDOW (2009)
3. Ali, M.S., Consens, M.P., Kazai, G., Lalmas, M.: Structural relevance: a common basis for the evaluation of structured document retrieval. In: CIKM, pp. 1153–1162 (2008)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: 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)
5. Biron, P.V., Ashok Malhotra, W.W.W.C.: W3C Recommendation. XML Schema Part 2: Datatypes Second Edition, October 28 (2004), http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/
6. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. IJSWIS 5(3), 1–22 (2009)
7. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs, Provenance and Trust. In: WWW, pp. 613–622 (2005)
8. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: WWW, pp. 1101–1102 (2008)
9. Consens, M.P., Rizzolo, F.: Fast answering of XPath query workloads on web collections. In: Barbosa, D., Bonifati, A., Bellahsène, Z., Hunt, E., Unland, R. (eds.) XSym 2007. LNCS, vol. 4704, pp. 31–45. Springer, Heidelberg (2007)
10. Consens, M.P., Rizzolo, F., Vaisman, A.A.: AxPRE summaries: Exploring the (semi-)structure of XML web collections. In: ICDE, pp. 1519–1521 (2008)
11. Cyganiak, R., Stenzhorn, H., Delbru, R., Decker, S., Tummarello, G.: Semantic Sitemaps: Efficient and Flexible Access to Datasets on the Semantic Web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 690–704. Springer, Heidelberg (2008)
12. Ding, L., Finin, T.: Characterizing the Semantic Web on the Web. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 242–257. Springer, Heidelberg (2006)
13. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: CIKM, pp. 652–659 (2004)
14. Dovier, A., Piazza, C., Policriti, A.: An efficient algorithm for computing bisimulation equivalence. Theor. Comput. Sci. 311(1-3), 221–256 (2004)
15. Klyne, G., Carroll, J. (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
16. Glaser, H., Millard, I., Jaffri, A.: RKBExplorer.com: A Knowledge Driven Infrastructure for Linked Data Providers. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 797–801. Springer, Heidelberg (2008)
17. Hassanzadeh, O., Consens, M.P.: Linked Movie Data Base. In: I-SEMANTICS, pp. 194–196 (2008)

18. Hausenblas, M., Halb, W., Raimond, Y., Heath, T.: What is the Size of the Semantic Web? In: I-SEMANTICS, pp. 9–16 (2008)
19. Khatchadourian, S., Consens, M.P.: ExpLOD: Exploring Interlinking and RDF Usage in the Linked Open Data Cloud. Technical Report (2009), http://www.cs.toronto.edu/~shahan/tr/explodtechreport090901.pdf
20. Kinsella, S., Bojars, U., Harth, A., Breslin, J.G., Decker, S.: An interactive map of semantic web ontology usage. In: IV, pp. 179–184 (2008)
21. Maduko, A., Anyanwu, K., Sheth, A.P., Schliekelman, P.: Graph Summaries for Subgraph Frequency Estimation. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 508–523. Springer, Heidelberg (2008)
22. Manola, F., Miller, E.: RDF Primer, W3C Recommendation, February 10 (2004), http://www.w3.org/TR/REC-rdf-syntax/
23. Paige, R., Tarjan, R.E.: Three partition refinement algorithms. SIAM J. Comput. 16(6), 973–989 (1987)
24. Stickler, P.: CBD - Concise Bounded Description, http://www.w3.org/Submission/2005/SUBM-CBD-20050603/
25. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: WWW, pp. 697–706 (2007)
26. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the Open Linked Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
27. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F.: Signing individual fragments of an RDF graph. In: WWW, pp. 1020–1021 (2005)
28. Vassiliadis, P., Sellis, T.: A survey of logical models for OLAP databases. SIGMOD Rec. 28(4), 64–69 (1999)
29. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and Maintaining Links on the Web of Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 650–665. Springer, Heidelberg (2009)

# Combining Query Translation with Query Answering for Efficient Keyword Search

Günter Ladwig and Thanh Tran

Institute AIFB, Karlsruhe Institute of Technology, Germany
{guenter.ladwig,duchthanh.tran}@kit.edu

**Abstract.** Keyword search has been regarded as an intuitive paradigm for searching not only documents but also data, especially when the users are not familiar with the data and the query language. Two types of approaches can be distinguished. Answers to keywords can be computed by searching for matching subgraphs directly in the data. The alternative to this is keyword translation, which is based on searching the data schema for matching join graphs, which are then translated to queries. Answering these queries is performed in the later stage. While clear advantages have been shown for the approaches based on query translation, we observe that processing done during query translation has some overlaps with the processing needed for query answering. We propose a tight integration of query translation with query answering. Instead of using the schema, we employ a bisimulation-based structure index graph. Searching this index for matching subgraphs results not only in queries, but also candidate answers. We propose a set of algorithms which allow for an incremental process, where intermediate results computed during query translation can be reused for query answering. In experiments, we show that this integrated approach consistently outperforms the state of the art.

## 1 Introduction

Graph structured data has attracted much attention recently, which is partly due to the massive availability of RDF. There are billions of freely available RDF triples, hosted by data providers involved in the LOD project. This number is increasing, just like the amount of RDFa data associated with Web pages.

In this paper, we present an approach for keyword search on graph structured data. Keyword search has been regarded as an intuitive paradigm for exploring and searching for data, especially in the case where the users are not familiar with the data and the query language. Approaches for computing structured answers from keywords exist for different data models, including relational, XML and graph structured data such as RDF [2,3,12,9,10,14,11].

**State of the Art.** Two different types of approaches can be distinguished. With *direct keyword query answering*, answers to keywords are computed by searching for matching subgraphs directly in the data, i.e. subgraphs which connect the data elements matching the keywords [3,12,9]. Implementing such an approach

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 288–303, 2010.

amounts to building a native engine for keyword search. Specific indexes and storage mechanisms have to be provided.

Alternatively, keyword search can be supported by computing translations in the form of queries, which can then be processed using an existing database engine. We refer to this as *keyword query translation*. Examples of database extensions for keyword search include DBXplorer [2] and Discover [11]. These systems are based on finding candidate networks, basically join expressions constructed using information found in the schema. These candidate networks are used to instantiate a number of SQL queries. Similarly, [18] extracts a schema from the RDF data graph. During online processing, this schema is augmented with elements matching the user keywords. Keyword translation is performed by finding matching subgraphs in this augmented schema representing possible queries. Processing the queries derived from them is finally performed using the underlying RDF store.

Keyword search based on translation has several advantages [18]. The produced queries can be presented to the user, thereby facilitating comprehension of the results and in particular, enabling query refinement. Moreover, keyword search can be performed efficiently: keyword translation is relatively fast, as exploration for subgraphs is performed on the schema. Since this is much smaller than the actual data, exploration for queries is much faster than exploration for answers in the data graph. For query answering, optimization capabilities of the database engine can be leveraged. For keyword search on RDF, it has been shown in [18] that combining fast query translation with state of the art query answering is faster than direct keyword query answering [3,12,9].

While clear advantages have been shown for translation-based approaches, we observe that processing done during query translation has overlaps with processing needed for query answering. We propose a tight integration of query translation with query answering. The main contributions of our approach are:

- We propose a novel *process which tightly combines query translation with query answering*. This is based on the observation that for query translation, keywords are matched against the data. Then, possible join graphs between these keyword matching elements are obtained. These operations are similar to data retrieval and join operations performed during query answering.
- Instead of a schema, we *employ a bisimulation-based structure index*. Searching this index for join graphs results not only in queries, but also candidate answers. This allows for an incremental process, where intermediate results computed during query translation can be reused for query answering.
- For efficient and incremental processing, we elaborate on new algorithms for all step in this process. We propose the *decomposition keyword queries* into segments such that search using these segments result in entities that can be further processed during query answering. The algorithm proposed for *join graph search* leverages existing strategies for exploration, but computes both join graphs and candidate answers. A special query answering algorithm is proposed for *processing join graphs*, which takes candidate answers as inputs.

In experiments, we show that this integrated and incremental process for keyword search consistently outperforms the state of the art.

**Outline.** We start with a problem definition in Section 2. A brief overview of our approach is presented in Section 3. In Sections 4, 5 and 6, we describe the major steps keyword mapping, query translation and answering. Evaluation results are provided in Section 8. We conclude with a summary in Section 9.

## 2 Problem

In this section, we present models for the data and queries. Then, we define the problem that we elaborate on.

**Data Model.** We deal with *general graph structured data* defined as follows:

**Definition 1.** *A* data graph *is an RDF graph* $G = (V = V_E \uplus V_D, L = L_R \uplus L_A, E)$ *where* $V_E$ *are entities (RDF resources),* $V_D$ *are data values (RDF literals),* $L_R$ *and* $L_A$ *are drawn from the set of object properties (relations) and data properties (attributes) labels, and* $E$ *represent edge instances.*

**Query Model.** Two different types of queries are distinguished in this setting: (1) the system query $q_s$, which is the one finally used by the query engine to retrieve answers and (2) the user query $q_u$, which is the one actually entered by the user. System queries $q_s$ are *conjunctive queries* of the following type:

**Definition 2.** *A* conjunctive query $q$ *is an expression of the form* $p_1 \wedge \ldots \wedge p_n$, *where* $p_n \in P$ *are* query atoms *of the form* $p(n_1, n_2)$ *with* $n_1$, $n_2$ *being* variables *or* constants *otherwise, and* $p_n$ *are called* predicates. *We distinguish between* relation query atoms $p_r \in L_R$ *and* attribute query atoms $p_a \in L_A$.

Regarded as the basic block for querying graph structured data (RDF), these queries have been the focus of recent work on RDF query processing [1,19,15].

The user query $q_u$ is a *keyword query*, i.e. $q_u = \{k_1, ..., k_n\}$ where each $k_i$ is a keyword. Users enter keyword queries because they are not familiar with the formal query language, the schema or the data. Fig. 1 shows a data graph and example queries.

**Conjunctive Query Answering.** Fig. 1c illustrates that since variables can interact in an arbitrary way, $q_s$ is essentially a graph pattern $q_s = (V_{var} \uplus V_{con}, L, E)$ consisting of a set of triple patterns $p(v_1, v_2)$. As usual, query answering amounts to *graph pattern matching*:

**Definition 3.** *A* match $q_s^m$ *of a query* $q_s$ *on a graph* $G$ *is a homomorphic mapping* $\mu_s$ *from the variables of* $q_s$ *to vertices of* $G$ *such that the according substitution of variables in the graph pattern would yield a subgraph of* $G$.

**Query Translation.** Translating $q_u$ means to find the queries $q_s$ representing possible interpretations of $q_u$. The focus lies on finding interpretations $q_s$ that have non empty results. In order for some graph pattern matches to exist for $q_s$,

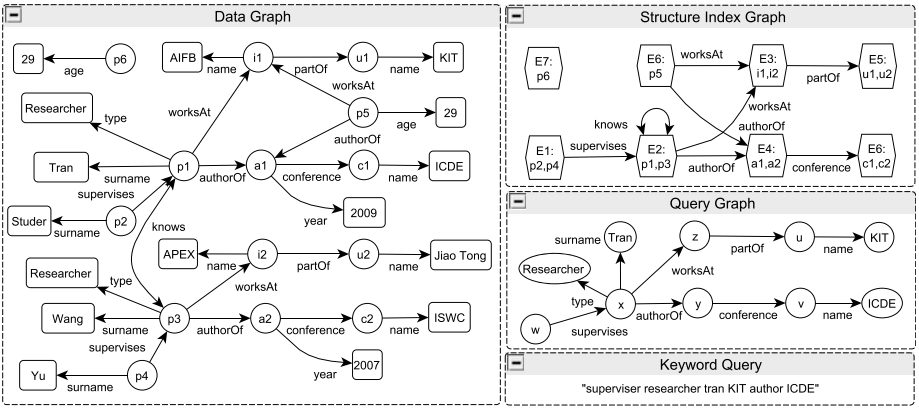



**Fig. 1.** a) A data graph, b) its structure index c) a query graph, d) a keyword query

predicates and constants of $q_s$ must correspond to some elements of $G$. Thus, we denote the space representing all possible interpretations of $q_u$ as $G_q$, a graph constructed using elements of $G$. An interpretation can be defined as a "keyword pattern" matching on $G_q$:

**Definition 4.** *An* interpretation $q_s$ *for* $q_u$ *is a subgraph of* $G_q(V_q, L_q, E_q)$ *connecting the keyword matching elements* $n \in N_k \subseteq V_q$*, where* $n$ *is obtained via a mapping* $\mu_u$ *from keywords of* $q_u$ *to elements in the set* $V_q \uplus L_q$*.*

## 3 Overview

We propose a tight integration of query translation and query answering to obtain an incremental process such that intermediate results obtained via the first step can be reused in the second step. This is based on the observation that query translation is very similar to query answering. For query answering, triple patterns are matched against the data to obtain matching triples. The entire query graph is processed by performing joins on the matching triples along the query edges. During query translation, a similar matching is needed in order to obtain keyword matching elements. Exploration for matching subgraphs connecting these keyword elements is similar to join processing – with the difference that there are no query edges that can be used to guide this process.

The overall process is as follows: We firstly *decompose* the initial keyword query into segments, and *map* them to entities and relation labels of $G$. This step operates on two indexes built for $G$, one containing all relation labels $L_R$ of $G$, called *relation label index*, and another one containing all attribute edges $G(V, L_A, E)$ called *entity index*. The exploration for queries and search for candidate answers is combined in one step called *join graph search*. For this we employ a query search space $G_q$ constructed from a bisimulation-based *structure index graph* $G^{\sim}$, which represents the different edge-labeled structures that

can be found in the data graph. Nodes of $G_q$ represent extensions which stand for entities in the data graph that are similar in structure. During join graph search, we explore $G_q$ for matching subgraphs from which candidate queries are derived. Because of the way the structure index graph is built from the data graph, these matching subgraphs are also candidate answers. That is, answers to the query are contained in the nodes (extensions) of the matching subgraphs and no other data has to be considered for query answering. Candidate answers in these extensions are refined in the subsequent *query answering* process. For this, the actual data graph edges have to be retrieved for elements in the extensions, using the *relation index* $G(V_E, L_R, E)$, and joined along the edges of the matching subgraphs in the last step.

**Comparison to Related Work.** The general top-$k$ subgraph exploration procedure proposed in [18] is adopted for join graph search. For brevity, we do not consider scoring models, while their use for top-$k$ and ranking as discussed in [18] is equally applicable to join graph search.

Different from work on direct keyword query answering [3,12,9], our approach aims to compute answers as well as queries that can be presented to the user. The exploration for subgraphs takes place on the structure index constructed from the data graph, instead of using the data graph itself.

So far, query translation has been performed independently from query answering [2,11,18]. Through the tight integration of these steps, we aim to minimize redundant processing by leveraging intermediate results. For this, we propose new algorithms for every major step of the integrated process: we decompose the keyword query into segments, from which the first initial result set of entities can be obtained. This is different to keyword mapping in previous approaches [2,11,18,3,9], where the aim is to find some matching elements (instead of entities). We propose the use of a structure index. As opposed to using the schema [2,11,18], exploring the structure index results in a set of join graphs, which represent queries and at the same time, contain candidate answers that can be refined in the subsequent step. We propose a novel query answering procedure, which, as opposed to existing approaches [1,19,15], accepts entities and the candidate answers contained in the join graphs as inputs.

## 4 Keyword Mapping = Entity Search

The first step to keyword search is keyword mapping, where the aim is to obtain a set of keyword matching elements. In direct keyword query answering, these elements are data elements, i.e. tuples or nodes of a data graph [3,9]. For translation, keywords $k$ are mapped against the query search space $G_q(V_q, L_q, E_q)$ to obtain query elements (predicates and constants), i.e. $\mu : k \mapsto V_q \cup L_q$ [2,11,18]. Since we are interested in queries with non-empty results, the computed query elements must match some elements of the data graph $G(V, L, E)$. In other words, query elements (and thus keywords) are assumed to be drawn from data graph element labels such that keyword mapping amounts to $\mu : k \mapsto V \cup L$. Thus, results of this

mapping might comprise entities $v_e \in V_E \subset V$, data values $v_d \in V_D \subset V$ and edge labels $l \in L$. Keywords are matched against labels using standard IR-style keyword matching.

In the interest of computing not only queries but also answers, we propose an alternative mapping $\mu : k \mapsto G_E \cup L_R$, where $k$ might not be a keyword, but a keyword segment $k_s \subseteq q_u$. Such a segment refers to entity descriptions $G_E$ or relation labels $L_R$ and thus, are called entity keyword query and relation keyword query respectively:

**Definition 5.** *A description of an entity $v_e \in V_E$ is a subgraph $G_E(V_e, L_e, E_e)$ of $G$ comprising all the edges $l(v_e, v_d) \in E$, where $v_d \in V_D$ (and thus $l \in L_A$). A keyword $k$ denotes an entity $e$ if there is a mapping $\mu : k \mapsto G_E$, where $G_E$ are descriptions of entities $e \in E$.*

*An* entity keyword query *is a set of keywords $K_e$, where the intersection of the matching elements of all keywords is not empty, i.e. $\bigcap_{k \in K_e} G_{E_k} \neq \emptyset$ with $G_{E_k}$ being the matching descriptions and $e \in E$ being the matching elements for a keyword $k$.*

*A keyword $k$ denotes a relation $r$ if there is a mapping $\mu : k \mapsto L_R$. Analogously, a* relation keyword query *is a set of keywords $K_r$, where the intersection of the matching relations of all keywords is not empty, i.e. $\bigcap_{k \in K_r} R_k \neq \emptyset$ with $R_k$ being the matching relation labels for keyword $k$.*

In other words, we map keywords to entities and relation labels. For this, the keyword query $q = K$ is decomposed into a set of entity keyword queries $K_E = K_{e_1}, \ldots, K_{e_n}$ and relation keyword queries $K_R = K_{r_1}, \ldots, K_{r_n}$. In particular, we are interested in partitions of the keywords, i.e. each keyword occurs in exactly one segment and there is no overlap between segments.

To create these partitions, we first generate segments of $K$ with non-empty results, i.e. matching segments. For this, the algorithm iterates through all subsets $S$ of $K$ in a bottom-up fashion, starting with single-element segments. The level corresponds to segment size such that at level 1, segments are of size 1. At every level, a segment $s_n \in S$ is added to the set of matching segments $S^m$, if the following conditions are satisfied: its subsegment $s_{n-1}$ containing the keywords $\{k_1, \ldots, k_{n-1}\}$ has results (this is the case if $s_{n-1} \in S^m$); the additional keyword $k_n \in s_n$ has a result; and the intersection of these two sets of results is not empty. The final result, i.e. all valid combinations of keyword segments, is obtained via an algorithm to generate all partitions of the set $K$ [5], using the valid segments in $S^m$.

*Example 1.* For the keyword query in Fig. 1d, we start with segments with length 1, to see that all of them match some elements in $G$. At level 2, we find that only the combination $researcher, tran$ is valid: $\mu(researcher) \cap \mu(tran) = \{p1\}$. Based on these valid segments, two partitions of the query are computed, i.e. $\{\{researcher, tran\}, \{KIT\}, \{ICDE\}, \{supervises\}, \{author\}\}$ and $\{\{researcher\}, \{tran\}, \{KIT\}, \{ICDE\}, \{supervises\}, \{author\}\}$.

# 5 Query Translation = Join Graph Search

## 5.1 Structure Index for Graph Structured Data

Structure indexes have been widely used for semi-structured and XML data [4,13,16]. A well-known concept for this is the dataguide [7], which basically is a structural description for rooted data graphs. Similar to this concept, a structure index has been proposed for general data graphs [17]. Nodes of this structure index stand for groups of data elements that have equal structural "neighborhood", where equal structural neighborhood is defined by the well-known notion of *bisimulation*. According to this notion, two graph nodes $v_1, v_2$ are *bisimilar* (written: $v_1 \sim v_2$) if they cannot be distinguished by looking only at their outgoing or incoming "edge-labeled trees". Pairwise bisimilar nodes form an extension. Applying the bisimulation $\sim$ to the subgraph $G(V_E, L_R, E)$ of our data graph that contains relation edges only, results in a set of such *extensions* $\{[v]_\sim : v \in V_E\}$ with $[v]_\sim := \{w \in V_E : v \sim w\}$. These extensions form a complete partition of the entity nodes $V_E$ of the data graph, i.e. form a family $\mathcal{P}_\sim$ of pairwise disjoint sets whose union is $V_E$.

Based on this notion of bisimulation, the *structure index* $G^\sim$ of $G(V_E, L_R, E)$ can be defined in terms of extensions and relations between them. In particular, extensions from the partition $\mathcal{P}_\sim$ form the vertices of $G^\sim$. An edge with label $l$ links two extensions $E_1, E_2 \in \mathcal{P}_\sim$ of $G^\sim$ exactly if $G$ contains at least one $l_r$-edge linking an element in the extension $E_1$ to some element in the extension $E_2$.

*Example 2.* The data graph shown in Fig. 1a can be partitioned into 8 extensions, shown as nodes of the index graph in Fig. 1b. Nodes $p1$ and $p3$ are grouped into extension E2 as they are bisimilar, i.e. both have incoming *supervise* and *knows* edges and both have the same outgoing edgs paths *knows*, $(worksAt, partOf)$ and $(authorOf, conference)$.

Note that similar to the structure index, a schema also represents a structural description of the data. However, a structure index is a description of the structures actually exhibited by the data, constructed from the data. In particular, it has the following property [17]:

*Property 1.* Whenever there is a match of a query graph $q$ on a data graph $G$ (homomorphism from $q$ into $G$) the query also matches on the index graph $G^\sim$ (homomorphism from $q$ into $G^\sim$). Moreover, nodes of the index graph matches will contain all data graph matches, i.e. the bindings to query variables.

## 5.2 Construction of Query Search Space

As opposed to direct keyword query answering, keyword query translation does not operate on the actual data $G(V, L, E)$, but on a query search space $G_q$. Essentially, this search space consists of two parts, (1) the keyword matching elements and (2) the structures that might connect these elements. Recall that in order for the computed queries $q_s$ to produce non empty results, all predicates

and constants of $q_s$ must match elements of $G$. Hence, the query search space shall be constructed from elements in $G$. Recall that keyword matching elements comprise edge labels $l \in L_R$ and entities $e \in V_E \subseteq V$. Typically, the schema is used to represent the possible structures [2,11,18]. Instead, we propose the use of a structure index to obtain the following query search space:

**Definition 6.** *Given the set of keyword matching elements* $n \in N^k \subseteq (L_R \uplus V_E)$ *and the data graph* $G$*, the* query search space $G_q$ *is defined as the index graph* $G^{\sim}(V^{\sim}, L^{\sim}, E^{\sim})$ *of* $G$*, extended with a special type of edges of the form* $contains([v]_{\sim}, n)$ *iff* $n \in V_E$ *and* $n$ *is in the extension* $[v]_{\sim} \in V^{\sim}$*.*

The query search space is thus the structure index graph, extended with keyword matching entity nodes. Since all different edge-labelled structures found in the data are represented in the structure index, all possible structures of queries with non empty results are completely captured by the proposed query search space. This result follows directly from Property 1 of the structure index:

**Theorem 1.** *Whenever a query graph pattern* $q$ *matches a data graph* $G$ *(homomorphism from* $q$ *into* $G$*), there are some subgraphs of* $G_q$ *that match* $q$ *(homomorphism from* $q$ *into* $G_q$*).*

Compared to the schema-based search space, the one proposed here is more appropriate for investigating possible translations. Since the schema does not necessarily represent actual structures in the data, many queries found through schema exploration might have empty results.

*Example 3.* Fig. 2a shows an example query search space, consisting of the structure index in Fig. 1b extended with the keyword matching entities $p1, u1, c1$.

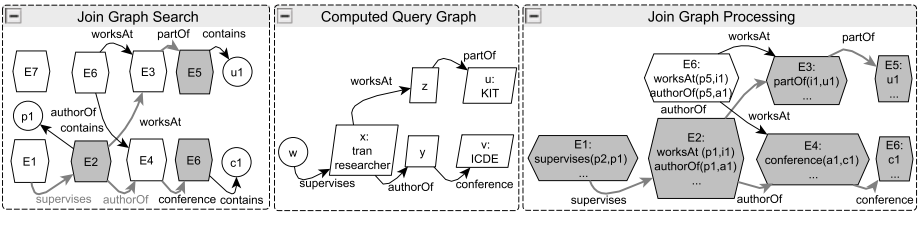



**Fig. 2.** a) Query space consisting of structure index, extended with keyword matching elements $p1, u1, c1$, join graph found in this query space is highlighted, b) structured query corresponding to the join graph and c) the data retrieved and joined during join graph processing, join paths are highlighted.

### 5.3 Search for Join Graphs

During this step, we explore the query search space for subgraphs connecting the keyword elements. More precisely, these subgraphs called *join graphs* are formalized as follows:

**Definition 7.** *Let* $G_q = (V_q, L_q, E_q)$ *be the query search space,* $K = \{k_1, \ldots, k_n\}$ *be the set of keyword segments and let* $f : K \rightarrow V_q \cup L_q$ *be a function that maps keyword segments to sets of corresponding graph elements. A* $K$-matching join graph *of* $G_q$ *is a graph* $G_q^m = (V_q^m, L_q^m, E_q^m)$ *with* $V_q^m \subseteq V_q$ *and* $L_q^m \subseteq L_q$ *such that*

- *for every* $k \in K$, $f(k) \cap (V_q^m \cup L_q^m) \neq \emptyset$, *i.e.* $G_q^m$ *contains at least one representative keyword matching element for every keyword from* $K$, *and*
- $G_q^m$ *is connected such that there exists a path from every graph element to every other graph element from* $G_q^m$.

In our approach, we search for graphs, which contain the keyword matching edges $l \in L_R$, and additionally, can be used to join the keyword matching entities $e \in V_E$. Such join graphs are constructed using edges $e([e_1]_\sim, [e_2]_\sim) \in E^\sim$ of the structure index part of the query search space $G_q$. As several keyword matching elements might be contained in the same extension, we firstly compute the sets of extensions for every keyword segment $k$, which we use as starting points for the exploration.

We use the top-$k$ exploration algorithm as described in [18], to which we refer for an in-depth discussion of the algorithm. The algorithm is based on the concept of cursors, which represent an exploration path from a start element to some node in the structure index graph. Initially, each cursor is associated with a keyword segment and the corresponding extensions previously computed as starting elements. At each step, the cursor with the lowest cost (e.g. path length, but other metrics are also possible) is chosen for expansion. Also, the element at the end of the current cursor is examined to check whether it was explored by other cursors, so that all keyword segments are covered. If this is the case, the corresponding cursors are merged to form a subgraph. The exploration terminates when the top-k subgraphs are found or the maximum exploration distance is encountered for all cursors. Discovered subgraphs that are isomorphic are aggregated to obtain one single query for those representing the same "interpretations" of the keywords.

*Example 4.* This example is about finding the join graph $G_q^m = \{supervises(E1, E2),\ worksAt(E2, E3),\ partOf(E3, E5),\ authorOf(E2, E4),\ conference(E4, E6)\}$ as highlighted in Fig. 2. To obtain this, we iterate through $N^k = \{\{p1\}, \{u1\}, \{c1\}, \{supervise\}, \{author\}\}$ to find the starting extensions $S^k = \{\{E2\}, \{E5\}, \{E6\}, \{supervise\}, \{author\}\}$. At the beginning, all cursors created for these start elements have equal length. Thus, we might start with any element from $S^k$. For instance, we pop the cursor $c(E2, E2, \emptyset, 1)$ and add it to $E2.C_{E2}$ to mark that E2 has been visited. Then, new cursors are created for neighbors $\{supervises, authorOf, worksAt, contains\}$ of $E2$. At the next iterations, remaining cursors of length 1 are taken from the queue and processed in the same manner. This proceeds until neighbor elements within distance of 4 have been explored for elements in $S^k$. Then, we find that E2 contains cursors to all other elements, i.e. $E2.(\{c(E2, E6, authorOf, 4)\},\ \{c(E2, supervises, supervises, 1)\},$

$\{c(E2,authorOf,authorOf,1)\}$, $\{c(E2,E5,worksAt,4)\}$). Paths represented by these cursors are merged to obtain the join graph $G_q^m$. Further exploration does not yield any results not isomorphic to $G_q^m$.

**Complexity.** As shown in [18], the time needed for join graph search is bounded by $|G_q|^{k_{max}}$ where $G_q$ is the query search space and $k_{max}$ is the maximum path length of query atoms considered for query translation. When compared to direct keyword query answering based on exploring the data graph [3,12,9], this makes up a crucial difference in time complexity. In our approach, $|G_q|$ corresponds to the size of the structure index, which is bigger than the schema, but typically, is order of magnitudes smaller than the data graph [17].

## 6 Query Answering = Join Graph Processing

Instead of retrieving triples and joining them along the query, as done in standard query processing, we propose query answering to be tightly coupled with query translation, such that the computed entities and join graphs corresponding to the user queries can be leveraged.

Recall that the join graphs $G_q^m$ computed during translation are subgraphs of the structure index. Every node of such a graph represents an extension, i.e. a set of entities. Due to Property 1, we can infer that the structure index enables us performing query translation and answering in an incremental way:

**Theorem 2.** *Extensions of a join graph $G_q^m$, from which the query $q$ has been derived, contain all the bindings to variables of $q$.*

**Proof 1.** *Query translation is performed based on the isomorphism from $G_q^m$ into $q$. Thus, $G_q^m$ is a match of $q$, i.e. homomorphism from $q$ into $G_q$. Since the structure index $G^\sim$ part of $G_q$ is the one used for the exploration, $G_q^m$ is in fact a homomorphism from $q$ into $G^\sim$, i.e. it is an index graph match. By Property 1, $G_q^m$ contains all the bindings to $q$.* □

Intuitively speaking, not only queries but also candidate answers were computed during translation. These answers are contained in $G_q^m$. Thus, it suffices to focus on elements in the extensions of $G_q^m$. More precisely, we can focus on the keyword matching entities in these extensions.

The query answering algorithm leveraging these intermediate results is shown in Alg. 1. Given the join graph $G_q^m$ including the keyword matching elements $e \in N^k$ contained in some nodes $[e]_\sim$ of $G_q^m$ (i.e. $e$ is connected with $[e]_\sim$ via $contains([e]_\sim, e)$), it computes all matches of this pattern on the data graph $G = (V, L, E)$. These matches are stored as tuples in the result table $R$. Just like standard query processing on graph structured data [1,19,15], edges of the query pattern $G_q^m$ are processed by retrieving triples $E(l_q^m)$ from the data graph matching the edge $l_q^m([e_1]_\sim, [e_2]_\sim)$. However, if given results exist, i.e. there are keyword matching entities contained in $[e_1]_\sim$ or $[e_2]_\sim$ such that $[e_1]_\sim.N^k$ or $[e_2]_\sim.N^k$ is not empty, they are used for further processing. Otherwise, triples

**Algorithm 1.** Join Graph Processing

**Input**: Join graph $G_q^m(V_q^m, L_q^m, E_q^m)$; Keyword matching entity nodes $N^k \subset V_E$; Subgraph $G(V_E, L_R, E)$ of the data graph G containing only entity nodes and relation edges.

**Data**: Sets of data graph edges $l_q^m(e_1, e_2) \in E(l_q^m)$ matching join graph edge $l_q^m$; The keyword matching entities $[e_1]_\sim.N^k$ contained in $[e_1]_\sim$.

**Result**: Table $R$ of answers where each row represents a match of $G_q^m$ onto $G$

```
1 foreach l_q^m([e1]~, [e2]~) ∈ E_q^m do
2     if [e1]~.N^k ≠ ∅ ∨ [e2]~.N^k ≠ ∅ then
3         E(l_q^m) ← {l_q^m(e1, e2) ∈ E | e1 ∈ [e1]~.N^k, e2 ∈ [e2]~.N^k};
4     else
5         E(l_q^m) ← {l_q^m(e1, e2) ∈ E | e1 ∈ [e1]~, e2 ∈ [e2]~};
6     end
7     R ← R ⋈ E(l_q^m);
8 end
9 return R;
```

$l_q^m(e_1, e_2)$ are retrieved from $G$. Due to the use of the structure index, only triples with $e_1 \in [e_1]_\sim$ and $e_2 \in [e_2]_\sim$ have to be retrieved. This number of triples might be considerably lower than all the triples with label $l_q^m$.

*Example 5.* For processing the translated query $q$ shown in Fig. 2b, triples are retrieved and joined along the edges $L_q^m = \{supervise, worksAt, authorOf, partOf, conference\}$ of the corresponding join graph $g$ shown in Fig. 2a. If we start with $supervise(E1, E2)$ for instance, we found that $E2.N^k = p1$. Thus, $E(l_q^m) = supervises(p2, p1)$, i.e. only one edge with $p1$ is retrieved. In this example, either $[e_1]_\sim.N^k$ or $[e_2]_\sim.N^k$ is always $\neq \emptyset$. Join graph processing for this example and the result $R = \{supervises(p2, p1), worksAt(p1, i1), partOf(i1, u1), conference(a1, c1)\}$ is illustrated in Fig. 2c.

**Complexity.** As shown in [17], for a join graph with $|E_q^m|$ edges, the time and space for computing the answer table $R$ is bounded by $O(edgemax^{|E_q^m|})$ where $edgemax$ is $|E(l_q^m)|$, with $l_q^m$ being the label instantiated by the largest number of edges. This complexity result holds also for existing query answering approaches, where $|E(l_q^m)|$ is derived from size of the tables built for edge labels $l \in L$. The crucial difference lies in this factor. In our approach, $|E(l_q^m([e_1]_\sim, [e_2]_\sim))|$ amounts to $max\{|[e_1]_\sim.N^k|, |[e_2]_\sim.N^k|\}$, i.e. the number of entities computed during query mapping, which are contained in the join graph nodes $[e_1]_\sim$ and $[e_2]_\sim$. If no entities have been obtained for either one of these nodes, $|E(l_q^m([e_1]_\sim, [e_2]_\sim))|$ amounts to $max\{|[e_2]_\sim|, |[e_2]_\sim|\}$, i.e. the size of the join graph nodes.

## 7 Evaluation

Through the experiments, we aim to compare performance of our approach against the state of the art.

**Systems.** For implementing the underlying indexes, we use a standard technique for indexing graph structured data. In particular, we use inverted indexes, as elaborated in [6]. We use an IR engine[1] for managing these indexes. This engine is also leveraged for mapping keywords to relation edge labels and entities (keyword mapping), as well as for the retrieval of relations (join graph processing). We compare our system against the keyword translation approach elaborated in [18]. It has been shown that the time for query translation achieved through [18], plus the time for query answering is lower than the time for the state of the art system based on direct keyword query answering [3,12,9]. For a detailed comparison with the state of the art approaches on direct keyword query answering, we refer to [18].

We have implemented this translation approach (QT) on top of a system for query answering (QA) based on vertical partitioning [1] and compared it against the implementation of our integrated approach (IQTQA). All implementations rely on the same engine for indexing and retrieval.

**Datasets.** The following datasets are used for the benchmark: *DBLP* captures bibliographic information about the field of Computer Science. It has been a standard dataset for evaluating keyword search [9,18]. *LUBM* is the Lehigh University benchmark, a synthetic dataset used extensively in the Semantic Web community for evaluating knowledge base systems and RDF stores [8]. We used the data generator proposed for the LUBM benchmark to create datasets for 1, 5, 10 and 50 imaginary universities.

**Table 1.** Statistics for the data graphs and indexes

| | Data [#Edges] | Data [MB] | EntityIdx [MB] | RelIdx [MB] | StrucIdx [KB] | Schema [KB] |
|---|---|---|---|---|---|---|
| DBLP | 12,920,826 | 2,084 | 2210 | 2311 | 132 | 28 |
| LUBM1 | 100,577 | 17 | 20 | 16 | 92 | 24 |
| LUBM50 | 6,654,596 | 1,132 | 1391 | 1,037 | 82 | 24 |

For these datasets, the size and number of edges are shown in Table 1. Also, the size of the associated entity index, relation index, structure index and schema is provided. One can see that the structure index is consistently bigger than the schema, but is of magnitudes smaller than the data graph. Further, its size is not dependent on the size of the data graph, but the structures contained in it.

**Queries.** In order to study the behavior of the proposed algorithms in a principled way, test queries are generated by random sampling from the data. Keyword queries are derived from generated structured queries.

**Setting.** We carried out the experiments on a Linux machine with two Intel Xeon Dual Core 2.33 GHz processors and 48GB of main memory, of which 2GB were allocated to the Java VM. All data and indexes were stored on a RAID

[1] http://lucene.apache.org/

array. All times presented are the average of 10 runs of 25 generated queries for each dataset. Between queries, we explicitly clear the operating system cache and internal caches.

**Total Processing Time.** We have measured the average processing time needed for QTQA and our approach IQTQA. For QTQA, we take the translation time + the query answering time for one query, the time needed for answering the first query output by query translation. For IQTQA, we consider query answering time needed for processing the first join graph, as well as the time needed for processing all join graphs output by query translation. As shown in Fig. 3a, query translation constitutes the greater share. Especially in our approach, query translation makes up for more than 90% of the total processing time. This is because we have chosen a generous value for $k_{max}$, resulting in a relatively large neighborhood to be explored and a large number of candidate interpretations to be computed. Consistently for all datasets, query translation using QTQA is faster than our approach IQTQA. The difference between these two approaches is greater for the LUBM datasets, where QTQA is up to 20 percent faster w.r.t query translation. This advantage of QTQA over IQTQA is however outweighed by the worse performance in query answering. While query answering takes almost half of the total time for QTQA, it makes up only a small fraction of the total time for IQTQA. This holds for all datasets. This even holds when not only the first join graph, but all join graphs are processed for IQTQA, i.e. query translation + computing answers for all computed queries using IQTQA is faster than query translation + computing answers for only one query using QTQA, w.r.t all datasets. It seems that most of the work has been done already during query translation such that the computation of answers for all computed queries consume only a small fraction of additional time.

**Effect of Data Size.** We have measured total time for LUBM of different data sizes. As illustrated in Fig. 3a, query translation time in both QTQA and IQTQA increases linearly with the size of the data. Further decomposition of this time into its two main components tells that this increase is mainly due to query mapping, while the time for join graph search remains relatively constant. This is consistent with our complexity analysis. The time for join graph search depends on the structure index. Since the size of the structure indexes (and schemas) is relatively constant in our experiments, this part of query translation is not affected by the changes in data size. However, query mapping time is partly determined by the number of mappings that can be retrieved from the data. Larger data sizes lead to larger number of mappings, which in turn, result in higher mapping time. Also query answering time increases linearly with the size of the data, for both QTQA and IQTQA. This is not surprising, as this time component is determined by the number of triples that can be retrieved for a query atom. Larger data size results in larger number of matching triples.

**Effect of Keyword Query Complexity.** We have run experiments with keyword queries of different complexity, as measured by the number of keywords. The query translation time for QTQA and IQTQA is illustrated in Fig. 3c for

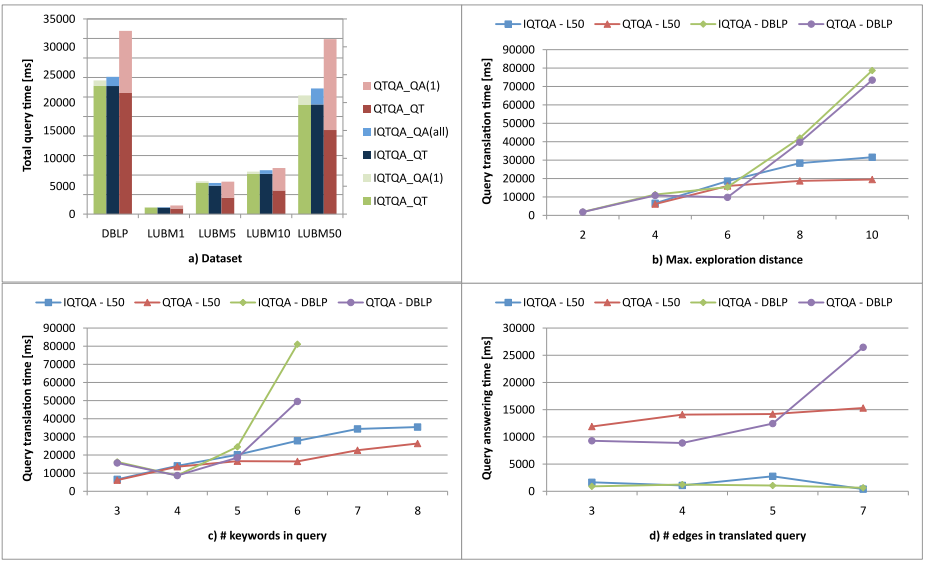



**Fig. 3.** Evaluation results

DBLP and LUBM50. Query translation time increases with the number of keywords. This is because the number of keywords has an influence on the complexity of both keyword mapping and join graph search. When the number of keywords increase, a larger number of mappings needs to be retrieved, a larger number of segments require to be tested and potentially, a larger number of starting elements have to be considered during join graph search. Compared with QTQA, the increase exhibited by IQTQA is slightly stronger. This can be explained by the larger search space employed by IQTQA, i.e. the structure index is bigger than the schema. The negative effect of a larger number of starting elements exacerbate, when the search space is larger in size.

**Effect of Maximum Neighborhood Distance.** This problem is more evident when looking at the maximum neighborhood size $k_{max}$ used for join graph search. We illustrate the effect of this factor in Fig. 3b. The increase of query translation time with greater $k_{max}$ is consistently stronger for IQTQA, especially w.r.t LUBM50. A greater $k_{max}$ means simply means a greater portion of the search space will be searched, thus resulting in higher translation time.

**Effect of Structured Query Complexity.** The negative effect due to the larger search space is overcompensated by the gain in query answering performance. In Fig. 3d, we illustrate the relation between query answering time and query complexity, where complexity is represented by the number of query edges. While time for QTQA increases linearly, time for IQTQA remains relatively constant. Here the result for QTQA is better than worst case performance because in many instances, retrieved data comes in sorted order, thus enabling fast joins.

Query times for IQTQA do not increase with the number of query edges because for most cases, intermediate results from translation have been leveraged such that only a small number of data elements are actually involved in the join.

## 8 Conclusion

We have proposed an approach for keyword search, which combines query translation with structured query answering to obtain an integrated process. Results produced during translation are reused to improve the efficiency of subsequent query answering. For this, we break down the overall process into two similar pattern matching problems: one is to match the keywords against a query search space to obtain a query, and the other is to match the query against the data. By constructing the query search space based on the data, i.e. the entities and the structures connecting them as represented by a structure index, the result of the first matching can greatly improve the efficiency of the second matching. We propose algorithms to perform these two matching tasks in an efficient and incremental way. We have established complexity bounds for these algorithms and compare these theoretical results with related work. In the benchmark against the state of the art, we show that our approach outperforms the state of the art.

**Acknowledgements.** Research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen project (grant 01IA08005K).

## References

1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB, pp. 411–422 (2007)
2. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: enabling keyword search over relational databases. In: SIGMOD Conference, p. 627 (2002)
3. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using banks. In: ICDE, pp. 431–440 (2002)
4. Buneman, P., Davidson, S., Fernandez, M., Suciu, D.: Adding Structure to Unstructured Data. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 336–350. Springer, Heidelberg (1997)
5. Djokic, B., Miyakawa, M., Sekiguchi, S., Semba, I., Stojmenovic, I.: A Fast Iterative Algorithm for Generating Set Partitions. Comput. J. 32(3), 281–282 (1989)
6. Dong, X., Halevy, A.Y.: Indexing dataspaces. In: SIGMOD Conference, pp. 43–54 (2007)
7. Goldman, R., Widom, J.: DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In: VLDB, pp. 436–445 (1997)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
9. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: SIGMOD Conference, pp. 305–316 (2007)
10. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient ir-style keyword search over relational databases. In: VLDB, pp. 850–861 (2003)

11. Hristidis, V., Papakonstantinou, Y.: DISCOVER: Keyword Search in Relational Databases. In: VLDB, pp. 670–681 (2002)
12. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional Expansion For Keyword Search on Graph Databases. In: VLDB, pp. 505–516 (2005)
13. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering indexes for branching path queries. In: SIGMOD Conference, pp. 133–144 (2002)
14. Liu, F., Yu, C.T., Meng, W., Chowdhury, A.: Effective keyword search in relational databases. In: SIGMOD Conference, pp. 563–574 (2006)
15. Neumann, T., Weikum, G.: Rdf-3x: A risc-style engine for RDF. PVLDB 1(1), 647–659 (2008)
16. Qun, C., Lim, A., Ong, K.W.: D(k)-Index: An Adaptive Structural Summary for Graph-Structured Data. In: SIGMOD Conference, pp. 134–144 (2003)
17. Tran, T., Ladwig, G., Rudolph, S.: Efficient RDF Data Management Using Structure-based Partitioning and Structure-aware Query Processing. Technical report, http://www.aifb.uni-karlsruhe.de/WBS/dtr/papers/strucidxTR.pdf
18. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In: ICDE, pp. 405–416 (2009)
19. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. PVLDB 1(1), 1008–1019 (2008)

# Improving the Performance of Semantic Web Applications with SPARQL Query Caching

Michael Martin, Jörg Unbehauen, and Sören Auer

Universität Leipzig, Institut für Informatik, Augustusplatz 10-11,
D-04109 Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org

**Abstract.** The performance of triple stores is one of the major obstacles for the deployment of semantic technologies in many usage scenarios. In particular, Semantic Web applications, which use triple stores as persistence backends, trade performance for the advantage of flexibility with regard to information structuring. In order to get closer to the performance of relational database-backed Web applications, we developed an approach for improving the performance of triple stores by caching query results and even complete application objects. The selective invalidation of cache objects, following updates of the underlying knowledge bases, is based on analysing the graph patterns of cached SPARQL queries in order to obtain information about what kind of updates will change the query result. We evaluated our approach by extending the BSBM triple store benchmark with an update dimension as well as in typical Semantic Web application scenarios.

## 1 Introduction

It has been widely acknowledged that the querying performance of triple stores is a decisive factor for the large-scale deployment of semantic technologies in many usage scenarios (cf. e.g. [9,4]). In recent years much progress has been made to improve the performance of triple stores by developing better storage, indexing and query optimization. However, compared to querying data stored in a fixed relational database schema, querying a triple store is still usually slower by a factor of 2-20 (cf. e.g. BSBM results[1]). This shortcoming is due to the fact that columns in a relational database are typed and may be indexed more efficiently. By using a triple store, this efficiency is lost to the flexibility of amending and reorganizing schema structures easily and quickly.

A circumstance currently not yet taken advantage of by triple stores is that in typical application scenarios only relatively small parts of a knowledge base change within a short period of time. The majority of triples remain unchanged. Hence, most queries will return the same results even after the occurrence of changes on the knowledge base. In addition, queries are often frequently issued,

[1] http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 304–318, 2010.

for example, when different users access the same information in a Semantic Web application. We can take advantage of this fact by caching query results, but also want to ensure that cached query results are selectively invalidated on knowledge base updates.

An analysis of a query shows what kind of changes of the knowledge base it will take to return a different result. In the meantime the results of the query can be temporarily stored for fast access. Our approach is based on examining SPARQL graph patterns. A query result is cached as long as updated triples do not match any of the triple patterns comprised by the graph pattern. Once an updated triple matches any of the triple patterns, the corresponding cache object is invalidated and will have to be recomputed by the triple store on a subsequent execution of the query.

Web applications are often composed out of smaller objects whose state depends on the execution of multiple queries. The product description page of an online shop, for example, is composed of header and footer components, a product category selection menu, the actual product description and possibly personal information of the actual user, such as the contents of his/her shopping cart etc. Traditional Web applications cache application objects or even whole parts of the generated user interface (i.e. HTML page fragments). The application logic then has to take care of invalidating these complex cache objects, for example, when new products are entered into the system or the user's shopping cart changes. We allow the caching and invalidation of more such compound application objects by associating them with all of the cached query results they depend on. The compound cache object is then invalidated when any of the associated query results change.

As a result, Semantic Web applications which frequently issue the same queries and are updated moderately are significantly accelerated. This improvement allows Semantic Web applications to get closer to conventional Web applications based on relational databases with regard to performance. In particular, we make the following contributions:

- We provide a method for selective invalidation of cached query results on triple store updates based on an analysis of SPARQL queries.
- We extended the caching of plain query results into a caching of compound application objects, based on a dependency tacking.
- We implemented the RDF query caching approach as a small proxy layer which resides between the Semantic Web application and an arbitrary SPARQL/SPARUL endpoint.
- We extended the BSBM triple store benchmark to consider updates and evaluated our approach with both the synthetic benchmark as well as in a practical Semantic Web application setting.

The paper is structured as follows: We describe the concepts and architecture of our caching solution in the Sections 2 and 3, while elaborating on the cache maintenance in Section 4. We also provide a comprehensive evaluation of the approach based on a synthetic benchmark as well as a real Semantic Web

applications in Section 5. We conclude and present related as well as future work in the Sections 6 and 7.

## 2 Concepts

In this section we describe the theoretical foundation of our approach. It is based on the SPARQL algebra and we refer to [8,7] for a more detailed description of the algebraic formalization of SPARQL. We will briefly introduce the formal SPARQL syntax and semantics and then derive a proposition about the invariance of graph pattern solutions when updates of the underlying RDF dataset do not match any of the triple patterns used in the graph pattern.

### 2.1 Syntax and Semantics of SPARQL

The SPARQL query language is based on the definition of the syntactic language features and a semantic interpretation of these syntactic features by means of set theoretical operators. We restrict ourselves to the core fragment of SPARQL over simple RDF (i.e. RDF without RDFS vocabulary and literal rules), which is sufficient for our purposes.

*Syntax.* Assume there are pairwise disjoint infinite sets $I$, $B$, and $L$ (IRIs, Blank nodes, and RDF literals, respectively). A triple $(v_1, v_2, v_3) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF *triple*. In this tuple, $v_1$ is the *subject*, $v_2$ the *predicate* and $v_3$ the *object*. We denote the union $I \cup B \cup L$ as by $T$ called RDF terms. Additionally, we assume the existence of an infinite set $V$ of variables which is disjoint from the above sets. An RDF *graph* is a set of RDF triples (also called RDF dataset, or simply a dataset).

A SPARQL graph pattern expression is defined recursively as follows:

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

SPARQL conditions are supposed to evaluate to boolean values. Additionally, we assume that for $(P$ FILTER $R)$ the condition $var(R) \subseteq var(P)$ holds, where $var(R)$ and $var(P)$ are the sets of variables occurring in $R$ and $P$ respectively.

*Semantics.* A mapping $\mu$ from $V$ to $T$ is a partial function $\mu : V \rightarrow T$. For a triple pattern $t$ we denote as by $\mu(t)$ the triple obtained by replacing the variables in $t$ according to $\mu$. The domain of $\mu$, $dom(\mu)$, is the subset of $V$ where $\mu$ is defined. Two mappings $\mu_1$ and $\mu_2$ are *compatible* when for all $x \in dom(\mu_1) \cap dom(\mu_2)$ we have $\mu_1(x) = \mu_2(x)$, i.e. when $\mu_1 \cup \mu_2$ is also a mapping. Note that two mappings with disjoint domains are always compatible and that the empty mapping (i.e. the mapping with empty domain) $\mu_\emptyset$ is compatible with any other mapping. Let $\Omega_1$ and $\Omega_2$ be sets of mappings. We define the join of, the union of and the difference between $\Omega_1$ and $\Omega_2$ as:

1. $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 | \mu_1 \in \Omega_1, \mu_2 \in \Omega_2$ are compatible mappings$\}$,
2. $\Omega_1 \cup \Omega_2 = \{\mu | \mu \in \Omega_1$ or $\mu \in \Omega_2\}$,
3. $\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 |$ for all $\mu_2 \in \Omega_2$, $\mu_1$ and $\mu_2$ are not compatible$\}$.

Now we can define the semantics of graph pattern expressions by means of a function $[[\cdot]]_D$, which takes a pattern expression and returns a set of mappings. As in [7], we assume, for the reason of simplicity, all datasets to be free of redundancies (i.e. duplicate triples).

**Definition 1 (Graph pattern evaluation).** *Let $D$ be an RDF dataset over $T$, $t$ a triple pattern, $R$ a SPARQL condition and $P_1$, $P_2$ graph patterns. Then the evaluation of a graph pattern over $D$, denoted as by $[[\cdot]]_D$, is defined recursively as follows:*

1. $[[t]]_D = \{\mu | dom(\mu) = var(t)$ *and* $\mu(t) \in D\}$,
2. $[[(P_1\ AND\ P_2)]]_D = [[P_1]]_D \bowtie [[P_2]]_D$
3. $[[(P_1\ OPT\ P_2)]]_D = ([[P_1]]_D \bowtie [[P_2]]_D) \cup ([[P_1]]_D \setminus [[P_2]]_D)$
4. $[[(P_1\ UNION\ P_2)]]_D = [[P_1]]_D \cup [[P_2]]_D$
5. $[[(P_1\ FILTER\ R)]]_D = \{\mu \in [[P_1]]_D | R(\mu)$ *evaluates to boolean true* $\}$

Note that we omitted a detailed description of the semantics of filter expressions for the purpose of brevity here. The elements $\mu$ of the result of an evaluation are also called solutions of the respective graph pattern.

### 2.2 Graph Pattern Solution Invariance

After we defined the syntax and semantics of SPARQL, we now investigate under which types of updates the results of SPARQL graph patterns change. This analysis lays the theoretical foundation for our query result caching framework, since a certain query result can be cached until an update of the underlying RDF would affect this particular query result. Speaking intuitively the solution of a graph pattern stays the same at least until a triple, which matches any of the triple patterns being part of the graph pattern, is added to or deleted from the RDF dataset.

**Proposition 1 (Graph pattern solution invariance).** *If $\Omega$ is the set of all solutions for the graph pattern $P$ with respect to a dataset $D$ and for a triple $t$ there exists no mapping $\mu$ from query variables to RDF terms such that $t \in \mu(P)$, then $\Omega$ is also the set of all solutions for $D_+ = D \cup \{t\}$ and $D_- = D \setminus \{t\}$.*

*Proof.* We first show (a) that the proposition holds when $P$ is a triple pattern and then (b) that the evaluation of a graph pattern does not change if the sets of all solutions for the triple patterns contained in the graph pattern do not change.

(a) We assume, $P$ is a triple pattern and there is a solution $\mu$ of $P$ with regard to $D_+$. According to the graph pattern evaluation (1) holds $\mu(P) \in D_+$. According to our precondition $t \notin \mu(P)$. Consequently, $\mu$ is a solution for $D_+ \setminus \{t\} = D$ and hence $\mu \in \Omega$. The proof for $D_-$ proceeds accordingly.

(b) The evaluation of graph patterns consisting of AND, OPT and UNION clauses (i.e. points 2-4 in Definition 1) is defined to be composed out of the solutions of the constituting graph patterns via the join, union and difference operators. Hence, if the set of solutions for $D$ equals the sets of solutions for $D_+$ ($D_-$) for the constituting graph patterns, so will the set of solutions for their composition. A similar argument holds for the application of a filter clause (i.e. point 5 in Definition 1): If $[[P_1]]_D = [[P_1]]_{D_+}$ ($[[P_1]]_D = [[P_1]]_{D_-}$), then the set of solutions for the filter clause stay the same, i.e. $[[(P_1 \text{ FILTER } R)]]_D = [[(P_1 \text{ FILTER } R)]]_{D_+}$ ($[[(P_1 \text{ FILTER } R)]]_D = [[(P_1 \text{ FILTER } R)]]_{D_-}$). □

## 3 Architecture

In order to employ the invariance of graph pattern solutions for caching, we have to be aware of all queries as well as of all dataset updates. Hence, we implemented our approach as a small proxy layer, which resides between the Semantic Web application and the SPARQL/SPARUL [10] endpoint. All SPARQL queries and SPARUL updates are routed through this proxy. Once the proxy receives a query, it checks whether a result for this query is cached in its local store. If that is the case, the result is directly delivered to the client without accessing the triple store. If the query was not previously stored and is not excluded from caching by user-supplied rules, the query is routed to the triple store and, before results are returned to the client, these are stored in the cache's local result store.

We developed two implementations of the SPARQL cache: Firstly, we integrated the cache as a component into the Erfurt API[2] - a middleware for Semantic Web applications used as foundation for OntoWiki [5]. As a second implementation, we developed a stand alone version in Java, which can be used in conjunction with arbitrary Semantic Web applications and SPARQL endpoints. Both implementations are evaluated and compared in Section 5.

## 4 Cache Population and Maintenance

Other than conventional Web application caching approaches, we have to accomodate two requirements: (1) we want to invalidate cache objects not only based on a unique identifier or predefined timespans, but selectively on updates of the triple store. (2) In addition to simple query results, we want to store more complex application objects which are composed out of the results of multiple queries.

In oder to accomodate these requirements, the cache object store is implemented based on a relational database. The ER diagram is visualized in Figure 1. Query results are stored in a serialized form in the `cache_query_result` table. Optionally, they are associated to surrounding cache objects stored in the `cache_object` table. Query results are firstly associated with RDF models the query is accessing and secondly, for fast invalidation, the triple patterns comprised by the graph patterns of the query are stored in the table

[2] http://aksw.org/Projects/Erfurt

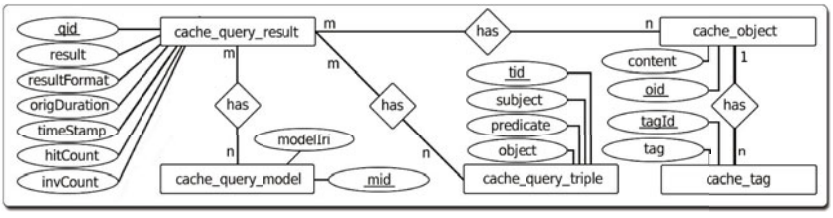



**Fig. 1.** ER diagram of the cache's relational object store

`cache_query_triple`. Variables in the triple patterns are represented as `NULL` values in this table.

### 4.1 Storing and Loading of Query Results

The general operation of our SPARQL cache is visualized in Figure 2. Once our SPARQL caching proxy receives a query, it computes the queries MD5 hash to determine quickly whether the query has already been cached or has to be (re-)executed. If the query result has not yet been stored in the cache, the SPARQL query is parsed and handed over to the original SPARQL endpoint. The returned result is stored together with the parsed query adhering to the cache schema. Currently, we use a relational database and in-memory backends to store cache objects.

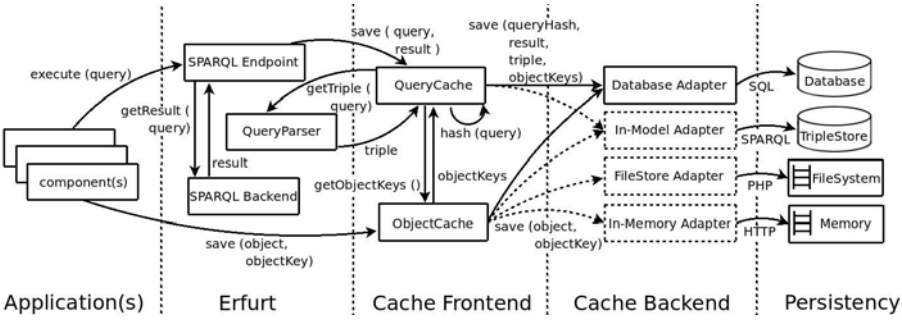



**Fig. 2.** Querying a SPARQL endpoint and storing the result using the query cache

Listing 1.1 shows an example query containing three triple patterns. Table 1 shows the rows which are added to the `cache_query_rt` and `cache_query_triple` tables. Since multiple queries might contain the same triple patterns, we store triple patterns only once and associate them with the queries (cf. Figure 1).

```
1 PREFIX aksw: <http://aksw.org/people#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 SELECT  ?classUri ?classLabel FROM <http://aksw.org/people#>
4 WHERE { ?classUri  rdfs:subClassOf  aksw:People .
5         ?classUri  aksw:sort  ?sort .
6         OPTIONAL { ?classUri  rdfs:label  ?classLabel } }
```

**Listing 1.1.** Example SPARQL query

**Table 1.** Extracted triple pattern from SPARQL query

| cache_query_rt | |
|---|---|
| **qid** | **tid** |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |

| cache_query_triple | | | |
|---|---|---|---|
| **tid** | **subject** | **predicate** | **object** |
| 1 | NULL | rdfs:subClassOf | aksw:People |
| 2 | NULL | aksw:sort | NULL |
| 3 | NULL | rdfs:label | NULL |

### 4.2 Storing and Loading of Application Objects

In addition to caching query results, our cache implementation offers to cache arbitrary application objects. For this, the cache implementation offers two functions `cacheStart($key)` and `cacheStop($key,$cacheObject)`. When the first function is called, the cache checks whether a cache object for `$key` exists and, if existent, returns this object. If a cache object is not available for the respective `$key`, an entry in the cache object store is created (table `cache_object` in Figure 1) and all subsequent SPARQL queries are associated with this cache object until the function `cacheStop` is called and the respective cache object content is stored.

### 4.3 Cache Maintenance

The graph pattern solution invariance as derived in Section 2 allows us to invalidate cache objects on triple store updates selectively, assuming that all updates (i.e. insertions or deletions of triples) are routed through the SPARQL query cache proxy. When a certain triple is inserted (or deleted) according to Proposition 1, we have to invalidate all SPARQL queries which contain a triple pattern matching the inserted (or deleted) triple. In addition, we invalidate all compound cache objects, which depend on one or more of the invalidated SPARQL query results.

Please note that the invalidation removes stored query results, but keeps the stored query structure and statistics (e.g. hit_count, inv_count) intact so that a subsequent execution of the query can reuse this information.

We illustrate the process with the addition of the following triple:

```
(G,S,P,O) = (http://aksw.org/people#,aksw:Student,rdfs:subClassOf,aksw:Person)
```

The following SQL query is subsequently used to invalidate query results:

```
UPDATE cache_query_result SET result=NULL WHERE ( qid IN (
SELECT DISTINCT(qid) FROM cache_query_rt JOIN cache_query_triple
                     ON cache_query_rt.tid = cache_query_triple.tid
WHERE ( (
 ( subject   = 'aksw:student'    OR subject   IS NULL ) AND
 ( predicate = 'rdfs:subClassOf' OR predicate IS NULL ) AND
 ( object    = 'aksw:Person'     OR object    IS NULL )
) ) ) AND qid IN (
 SELECT DISTINCT (qid) FROM cache_query_rm JOIN cache_query_model
                       ON cache_query_rm.mid = cache_query_model.mid
 WHERE ( cache_query_model.modelIri = 'http://aksw.org/people#' OR
         cache_query_model.modelIri IS NULL ) ) )
```

## 5 Evaluation

We measured the impact of our caching solution on the querying performance in two scenarios. First we employed the Berlin SPARQL Benchmark (BSBM, [4]) to demonstrate the cache's abilities by using a well-known test procedure. The second evaluation scenario measures the performance improvements for the Semantic Web application Vakantieland.

All benchmarking was done on a machine with the following configuration: Intel Core 2 Duo (P8400: 2x2.276GHz), 2x2GB of RAM, 160GB SATA HD (7,200rpm), Ubuntu 9.04 32 bit, Java 1.6, PHP 5.2.10, OpenLink Virtuoso 5.09 (NumberOfBuffers=300000, MaxDirtyBuffers=50000).

### 5.1 Berlin SPARQL Benchmark

The Berlin SPARQL Benchmark (BSBM) is based on an e-commerce use case, simulating an end-user search for products, vendors and reviews. The resulting SPARQL queries are grouped into mixes, each one consisting of 25 queries. The queries are derived from twelve different types and are instantiated by replacing parameters with concrete, randomized values. The QueryMixes per Hour (QMpH) assessment then states how many of these query mixes a certain triples store is able to execute per hour.

While in the original benchmark the probability for selecting a specific parameter is equal for each parameter, we chose to have the parameters selected according to the Pareto distribution, since this reflects practical use cases better and enables us to measure the performance gain of our caching solution in such scenarios. The probability density function can be described by (cf. [15]):

$$P(x) = \frac{ab^a}{x^{a+1}}$$

The parameter $a$ defines the shape, whereas $b$ defines the minimum value. Applied to the benchmark scenario, this implies that we have a number of products or offers that are queried more often than others. In our benchmark adoption we varied the parameter $a$ in order to see how well the caching implementation

**Table 2.** Distribution of the parameter in dependency to the choice of $a$

| Distr. parameter | linear | a=0.1 | a=0.3 | a=0.5 | a=1.0 | a=2.0 | a=4.0 |
|---|---|---|---|---|---|---|---|
| **Unique queries** | 11718 | 6205 | 4147 | 2953 | 1694 | 624 | 142 |
| **Res. distribution** | 50.5/49.5 | 64/36 | 72/28 | 78/22 | 84/16 | 88/12 | 90/10 |

adopts to a wider or narrower spectrum of repeated queries. For the Pareto principle (commonly known also as the 80/20 rule of thumb), Table 2 shows how the choice of $a$ broadens the distribution of the parameter (based on a benchmark with 10 million triples and 12,500 queries).

For example, using a linear distribution for creating 12,500 requests results in the generation of 11,718 unique queries, which represents a very limited number of repeated queries (i.e. 782). As for the other extreme of $a = 4$ a total of only 142 unique queries is generated and 90% percent of the 12,500 requests are executed with just the 10% of the unique queries, resulting in a very high level of repetition. Querying was parallelized by using 5 threads totaling 500 query mixes, which is similar to the original BSBM benchmark. For our benchmark adoption we used the Stochastic Simulation in Java library [3] to generate random numbers adhering to the Pareto distribution.

While the Java implementation of the cache was benchmarked as a SPARQL endpoint we also wanted to determine the performance impact on Web applications. Hence, for the Erfurt implementation time was measured by using JMeter[4]. The query duration was then transformed into QueryMixes per Hour (QMpH) for better comparability.

**Impact of Distribution.** This scenario demonstrates the impact of the query distribution on the performance. We tested various dataset sizes, ranging from 1 million to 25 million with various distributions. The results are summarized in Figure 3.

As expected, we found that the distribution of the queries has a high impact on performance. Performance can benefit enormously when using the cache with a high level of query repetition. For $a > 2$ performance is nearly independent of the store size, since most of the queries can be answered from the cache. Applications with a lower number of repeating queries may not benefit as much, in the Erfurt implementation, for broader distributions with $a \leq 0.3$ the SPARQL cache is not (yet) able to improve performance. For scenarios with larger datasets ($\geq 10$ million triples) and a moderate query repetition ($1.0 \geq a \geq 0.3$), performance improvements between 12% and 151% are possible.

The second parameter evaluated here is the store size, which, together with the distribution, defines the point at which an application benefits from using a cache. For the Erfurt implementation using a store with 1 million triples, caching offers improved performance starting at $a = 1$, whereas a larger store with 25 million triples profits much earlier, i.e. already from $a = 0.3$ onwards.

[3] http://www.iro.umontreal.ca/~simardr/ssj/
[4] http://jakarta.apache.org/jmeter/

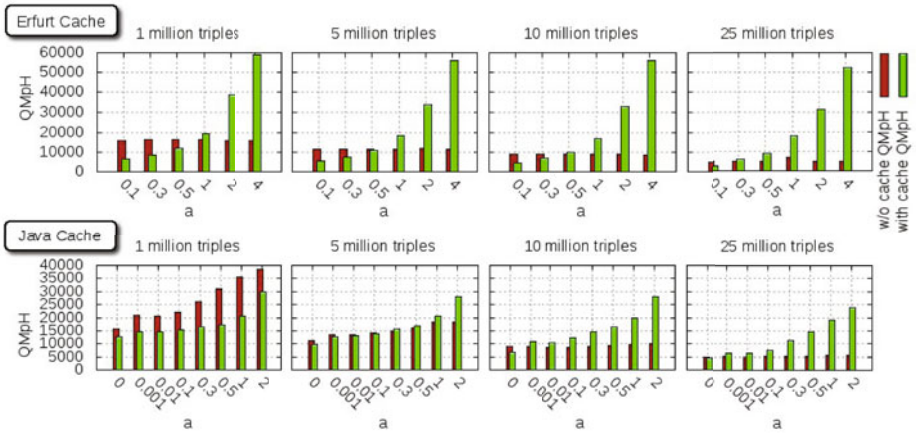



**Fig. 3.** Impact of distribution parameter on query performance

Most notably regarding the Java implementation is that, when accessed directly via SPARQL, Virtuoso shows similar performance gains with increasing query repetition in the 1 million triples scenario. With greater store size this effect diminishes. For larger store sizes, however, even for relatively low values of $a$ performance gains can be noted, which can be attributed to the light overhead the cache imposes on queries. With no query repetition the cache generates an overhead of just 8% up to 25% in the worst case.

**Impact of Updates.** Since cache objects will become stale, when the underlying dataset changes, we extended BSBM to support the modification of the queried graph. We removed a number of triples from the original graph in order to load them later, during querying, into the graph. This was implemented by adding inserts into some query mixes, with each insert containing 5 to 8 statements. Thus, we can show how the invalidation of cache objects and the subsequently required re-issuing of the query affects the performance. We compared the impact of different update frequencies with and without caching for the 5 (Erfurt) and 10 (Java) million triples dataset and with the query distribution parameter $a = 1$ (Erfurt) and $a = 0.3$ (Java). The update frequency is determined by the rate of query mixes containing an insert statement.

The results, as depicted in Figure 4, first contain a reference value without inserts, where the cache enabled version executes 60% more QMpH. For the Erfurt cache this advantage is slightly affected by an insert included in every $100^{th}$ query mix, reducing the performance gain to 48%. With an insert included in every $10^{th}$ query mix, the performance gain drops to 33%. Including an insert in every query mix and thus every $25^{th}$ query being an insert statement, reduces performance by 9% compared to the direct use of Virtuoso. For the Java implementation we measured slightly different results. The effect on performance due to the update queries is here stronger, the performance gain drops to 17%, 6%

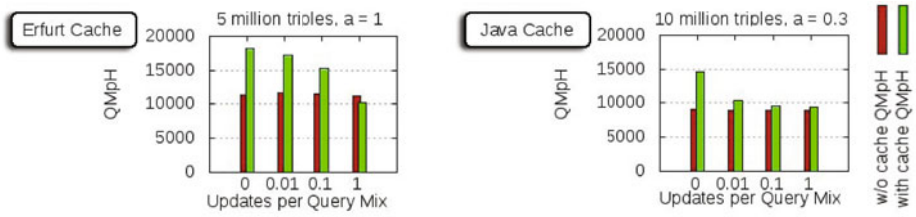



**Fig. 4.** Impact of update frequency on query performance

and 5%, but does not fall below the values of the same queries without cache. We attribute this behavior to the different cache integration approach.

While the figures presented here show that update frequency and the resulting cache maintenance affect the performance, most Web applications update frequencies are rather at the lower end of the tested values and will thus still significantly benefit from caching. Likewise, with larger datasets the positive impact of caching will be even more noticeable.

### 5.2 Benchmarking the Semantic Web Application Vakantieland

We evaluated the performance of the SPARQL query and application object caching with the Semantic Web application *Vakantieland*[5]. Vakantieland publishes comprehensive information about 20,000 touristic points-of-interest (POI) in the Netherlands such as textual descriptions, location information and opening hours. The information is stored in a knowledge base containing almost 2 million triples and is structured using approximately 1,250 properties as well as 400 classes. Vakantieland was designed according to the model/view/controller principle and uses the Erfurt API as middleware. Almost all of the information presented in Vakantieland is retrieved using SPARQL. Figure 5 marks areas of the Vakantieland user interface, which are significantly facilitated by the cache. Area 1 and 2 contain different category trees. The first is modelled hierarchically using `owl:class` and `rdfs:subClassOf`. The second category tree represents administrative areas of the Netherlands, containing provinces, districts and cities. For rendering both hierarchies, recursively executed SPARQL queries are used. The remaining two areas (numbered 3 and 4 in Figure 5) provide a collection of POIs (3) and a pagination for navigating over them (4). POIs are presented depending on given filter criteria. These criteria can be free text search, a class or a spatial-area selection, a map-bounding box or a combination of these. Every POI description, which can also be visited on a separate details page, consists of properties arranged in a property hierarchy using `rdfs:subPropertyOf`. These property hierarchies are also obtained using a set of recursively executed SPARQL queries, whose performance was substantially improved by the cache. For automatically benchmarking the behaviour of Vakantieland in combination

[5] Currently available at: `http://staging.vakantieland.nl`

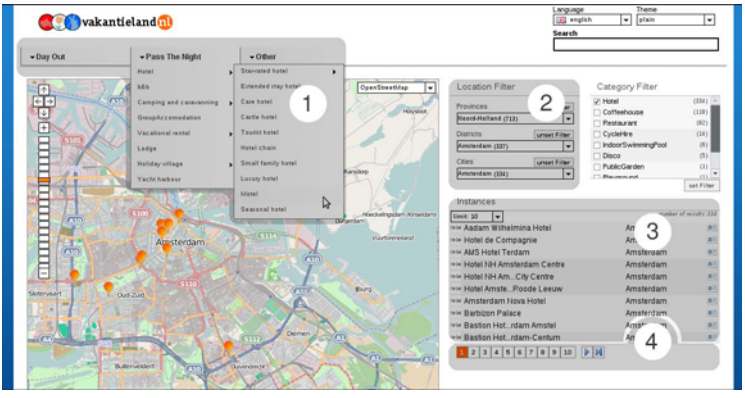



**Fig. 5.** Vakantieland GUI with marked UI components

with the cache, we defined the following usage scenarios. In every usage scenario we simulated the usage of the pagination, as depicted in Figure 5 (area 4), 200 times with different offsets.

- **Usage Scenario A** No selected filter criteria; no updates.
- **Usage Scenario B** Selected spatial-area filter; updating the `rdfs:label` of the selected spatial filter after every 25th request),
- **Usage Scenario C** Selected class filter; adding a new point-of-interest after every 25th request (`rdf:type` relation of the new resource is identical to the selected category filter),
- **Usage Scenario D** Selected tourism category filter; updating the `rdfs:label` of the selected category after every 25th request),

The results of these use cases are presented in Table 3 and show that the cache proxy implementation improves performance substantially. The application is accelerated between factor 5 (in scenario C) and factor 13 (in scenario A). For improving the performance of SPARQL queries, which filter information by geo-coordinates or search terms, we generated the additional index $P, G, S, O$ on the table RDF_QUAD in Virtuoso[6]. Due to the use of this index, such SPARQL queries can be executed five times faster. Other tested indexes do not measurably improve request times.

## 6 Related Work

To the best of our knowledge only few approaches exist aiming at improving query performance of RDF stores by means of query caching. We consequently also examine the use of caching in relational, DB-based Web applications and examine how they relate to our solution.

[6] Described at `http://docs.openlinksw.com/virtuoso/rdfperformancetuning.html`

**Table 3.** Benchmark results of application Vakantieland

| **Benchmark** | | **Sceanrio A** | **Sceanrio B** | **Sceanrio C** | **Sceanrio D** |
|---|---|---|---|---|---|
| **Cache disabled** | **pt** | ≈ 5480ms | ≈ 5262ms | ≈ 3967ms | ≈ 3992ms |
| | **c** | ⊘ 428 | ⊘ 297 | ⊘ 344 | ⊘ 344 |
| **Warm-Up** | **pt** | ≈ 3561ms | ≈ 3432ms | ≈ 3043ms | ≈ 3042ms |
| with enabled **QC** | **qch** | ⊘ 358 | ⊘ 249 | ⊘ 286 | ⊘ 286 |
| | **qci** | | ≈ 0,32 | ≈ 1,16 | ≈ 0,24 |
| **Hot Run** | **pt** | ≈ 2048ms | ≈ 2550ms | ≈ 2152ms | ≈ 1901ms |
| with enabled **QC** | **qch** | ⊘ 420 | ⊘ 289 | ⊘ 335 | ⊘ 336 |
| | **qci** | | ≈ 0,32 | ≈ 1,12 | ≈ 0,24 |
| **Warm-Up** | **pt** | ≈ 4152ms | ≈ 2783ms | ≈ 3019ms | ≈ 3088ms |
| with enabled **QC** and **OC** | **qch** | ⊘ 162 | ⊘ 48 | ⊘ 91 | ⊘ 91 |
| | **qci** | | ≈ 0,32 | ≈ 1,11 | ≈ 0,24 |
| **Hot Run** | **pt** | ≈ 403ms | ≈ 477ms | ≈ 686ms | ≈ 434ms |
| with enabled **QC** and **OC** | **qch** | ⊘ 6 | ⊘ 8 | ⊘ 5 | ⊘ 6 |
| | **qci** | | ≈ 0,32 | ≈ 1,12 | ≈ 0,24 |

**QC**:***Q**uery **C**ache*; **OC**:***O**bject **C**ache*;
**pt**:***p**rocess **t**ime per request*; **c**:*query **c**ount per request*;
**qch**:***q**uery **c**ache **h**its per request*; **qci**:***q**uery **c**ache **i**nvalidation per request*

*Caching in Web Applications.* Caching, which, in contrast to database replication, relies on intercepting queries, is distinguished in [13] into two different approaches: Content-Aware Caching (CAC) and Content-Blind Caching (CBC).

*Content-Aware Caching (CAC)* systems create upon intercepted queries new Partially Materialized Views (PMVs). This approach is, for example, implemented by DBproxy [3] or MTCache [6]. Whenever a query is executed, the CAC system checks whether the query is entailed by previously cached content and in case it is, the result is computed upon that or the query is forwarded to the database server. By proposing a query federation system, DBCache [2] is further able to relay non-cached parts of a query to the main database.

*Content-Blind Caching (CBC)* systems, in contrast, are not aware of the structure of the cached data. As demonstrated in GlobeCBC [12], storing only the result and meta-information can be an efficient approach, especially in scenarios with a high query repetition, as costs associated with the subsumption checks can be avoided. Our caching approach can, therefore, be considered to be a CBC system, as the query results are opaque to the cache and are not modified. The systems introduced here rely for cache maintenance and invalidation on database replication mechanisms which notify the caches on updates. While our invalidation mechanism is triggered by intercepted SPARUL queries, integrating this mechamism into data base replication is a possiblity for future work.

*Caching for Semantic Web Applications.* In [16] a write-through cache holding triples with commonly used subjects is described. Furthermore, property tables

as a storage scheme for RDF is introduced, similar to the idea of a vertical partitioning of an RDF store [1]. Used for frequently reoccurring query patterns, this concept can be transferred into creating an RDF Content-Aware Cache. With an intelligent materialization algorithm, the proliferation of tables, as discussed in [11], should be avoidable.

For caching in client-server or peer-to-peer and distributed database environments, query containment algorithms are developed to reuse a query result by subsuming a distinct query. In [14] this approach is applied to RDF stores. It is based on the notion of similarity of RDF queries determined by the costs of transform the results of a previous query into the result for the actual one. The paper discusses the problem of subsumption for RDF queries, presents a cost model and derives a similarity measure for RDF queries based on the cost model and the notion of graph edit distance. The author further sees the strong need to develop strategies for building and maintaining the cache, i.e. changes in the stored information have to invalidate parts of the cached results, as is the main contribution of our approach.

## 7 Conclusions and Future Work

We presented a novel approach for caching the results of querying triple stores and compound application objects containing such queries. The approach is based on the observation that large parts of a knowledge base usually do not change over time and hence only a small part of the query results are affected by updates to the knowledge base. By identifying the affected query results we are able to selectively invalidate cache objects on updates of the knowledge base such that the cache never contains outdated cache objects. We were able to show that our approach outperforms cacheless triple stores in realistic usage scenarios by more than factor 10. Only in scenarios with small knowledge bases (<1M triple) or very infrequent query repetition our cache adds some overhead. Currently our implementation is only loosely coupled with the underlying triple store. By tighter integrating the cache with the triple store even higher performance gains will be possible.

*Future Work.* We consider this work as an initial step towards closing the performance gap between relational database and triple store based applications. In order to further exploit the possibilities of caching we aim at looking how the results of a cached query can be reused in a content aware way for answering subsequent queries. In particular, the evaluation of SPARQL filter conditions is a promising candidate for further speed improvements.

Most triple stores are meanwhile equipped with support for light-weight inferencing. While our caching strategy will work well with forward-chaining reasoning approaches (the inferencing of new triples can be simply considered as updates) it still remains to explore how it can be combined with backward chaining inferencing. Another promising direction of future work in the context of the emerging Linked Data Web is how our caching approach can be employed for the acceleration of distributed and federated queries.

## References


1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable semantic web data management using vertical partitioning. In: VLDB. ACM, New York (2007)
2. Altinel, M., Bornhvd, C., Krishnamurthy, S., Mohan, C., Pirahesh, H., Reinwald, B.: Cache tables: Paving the way for an adaptive database cache. In: VLDB (2003)
3. Amiri, K., Park, S., Tewari, R., Padmanabhan, S.: Dbproxy: A dynamic data cache for web applications. In: ICDE (2003)
4. Bizer, C., Schultz, A.: The berlin SPARQL benchmark. International Journal on Semantic Web and Information Systems (2009)
5. Heino, N., Dietzold, S., Martin, M., Auer, S.: Developing semantic web applications with the ontowiki framework. In: Networked Knowledge - Networked Media. SCI, vol. 221. Springer, Heidelberg (2009)
6. Larson, P.-Å., Goldstein, J., Guo, H., Zhou, J.: Mtcache: Mid-tier database caching for SQL server. IEEE Data Eng. Bull. 27(2) (2004)
7. Perez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 30–43. Springer, Heidelberg (2006)
8. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (2008), http://www.w3.org/TR/rdf-sparql-query
9. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: Sp2bench: A SPARQL performance benchmark. In: ICDE. IEEE, Los Alamitos (2009)
10. Seaborne, A., Manjunath, G.: SPARQL/Update - a language for updating RDF graphs (2008), http://www.w3.org/Submission/SPARQL-Update/
11. Sidirourgos, L., Goncalves, R., Kersten, M.L., Nes, N., Manegold, S.: Column-store support for RDF data management: not all swans are white. PVLDB 1(2) (2008)
12. Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: GlobeCBC: Content-blind result caching for dynamic web applications. Technical Report IR-CS-022, Vrije Universiteit, Amsterdam, The Netherlands (June 2006)
13. Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for web applications. IEEE Internet Computing 11, 60–66 (2007)
14. Stuckenschmidt, H.: Similarity-based query caching. In: Christiansen, H., Hacid, M.-S., Andreasen, T., Larsen, H.L. (eds.) FQAS 2004. LNCS (LNAI), vol. 3055, pp. 295–306. Springer, Heidelberg (2004)
15. Weisstein, E.W.: Pareto distribution. MathWorld - A Wolfram Web Resource (2009)
16. Wilkinson, K., Sayers, C., Kuno, H.A., Reynolds, D.: Efficient RDF storage and retrieval in Jena2. In: SWDB, pp. 131–150 (2003)

# An Unsupervised Approach for Acquiring Ontologies and RDF Data from Online Life Science Databases

Saqib Mir[1,2], Steffen Staab[2], and Isabel Rojas[1]

[1] EML-Research Schloss-Wolfsbrunnenweg 31c, 69118 Heidelberg, Germany
[2] University of Koblenz-Landau, Koblenz, Germany
{saqib.mir,isabel.rojas}@eml-r.org, staab@uni-koblenz.edu

**Abstract.** In the Linked Open Data cloud one of the largest data sets, comprising of 2.5 billion triples, is derived from the Life Science domain. Yet this represents a small fraction of the total number of publicly available data sources on the Web. We briefly describe past attempts to transform specific Life Science sources from a plethora of open as well as proprietary formats into RDF data. In particular, we identify and tackle two bottlenecks in current practice: Acquiring ontologies to formally describe these data and creating "RDFizer" programs to convert data from legacy formats into RDF. We propose an unsupervised method, based on transformation rules, for performing these two key tasks, which makes use of our previous work on unsupervised wrapper induction for extracting labelled data from complete Life Science Web sites. We apply our approach to 13 real-world online Life Science databases. The learned ontologies are evaluated by domain experts as well as against gold standard ontologies. Furthermore, we compare the learned ontologies against ontologies that are "lifted" directly from the underlying relational schema using an existing unsupervised approach. Finally, we apply our approach to three online databases to extract RDF data. Our results indicate that this approach can be used to bootstrap and speed up the migration of life science data into the Linked Open Data cloud.

## 1 Introduction

The Life Sciences have seen a data explosion in the last decade. These data are most commonly stored in freely-accessible Web-based databases. The Nucleic Acids Research (NAR) Journal puts the current number of such Web databases that also have an application note with the journal at 1170 [1]. As the number of these databases increases, so does their size. Large Proteomics and Genomics databases actually exhibit exponential growth. These databases often contain complementary data, pertaining to narrow and specialized sub-domains. Any meaningful scientific investigation typically requires accessing many sources, manually extracting and linking data records together. This activity is made even more difficult with the fact that search engines cannot index most dynamically

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 319–333, 2010.

generated Web pages. Therefore, there is a pressing need for providing unified and integrated access to these sources.

Traditional database integration techniques require direct access to the underlying relational database. However, in the scenario presented above, this is almost never the case. In some instances, a part of the database is made available for download in diverse formats, including XML, tab-delimited text files, spreadsheets, or proprietary formats. Furthermore, these data can be often stale and incomplete.

More recently, there have been a number of approaches using Semantic Web technologies to create RDF triple stores by aggregating data from various Biochemical databases. The earliest approach was YeastHub [2], which built a centralized RDF store for Yeast data collected from eight online sources. The data was mapped on to a manually built ontology, as well as existing ontologies like RSS[1] and Dublin Core[2]. In FungalWeb [3], an OWL-DL ontology was manually developed and instantiated for describing enzymatic data. Stephens et al. [4] provided a drug-discovery use case by manually inspecting and integrating 12 biochemical databases into an RDF data model. Similarly, Pasquier [5] integrated more than 10 life science databases by manually creating an ontology and merging it with existing ontologies like GO[3] and GOA[4]. FlyWeb [6] integrates textual and image data for Drosophila from three databases, using the D2RQ [7] tool to convert relational data into RDF, as well as hand-written scripts for spreadsheet data. Bio2RDF [8] is perhaps the largest source of Biochemical RDF data. It integrates information from a variety of formats using hand-written "RDFizer" programs that populate an OWL ontology that has been manually created. The SBMM toolbox [9] shares a feature with Bio2RDF, whereby wrappers are manually created for specific source database Web sites. A user query is transmitted to search interfaces of appropriate sources, and the wrappers extract and convert the generated data into RDF.

One common limiting factor in the above mentioned approaches is the significant amount of manual work required to construct ontologies, populate data from legacy formats into this ontology and to link this data. We address the first two challenges by proposing unsupervised techniques for ontology learning and data extraction. We begin by answering the following two basic questions:

1. What sources should we utilize for ontology learning?
2. Which data format should we target for automatic data extraction?

In an ideal world, all sources would provide their data directly in RDF format. However, only a handful of sources, such as UniProt, Gene Ontology, IntAct and NCBI Taxonomy, do so. The wide variety of export formats make the task of unsupervised learning extremely hard and in many cases there is no possibility to export data at all. Furthermore, Biochemical databases are updated

[1] http://web.resource.org/rss/1.0/
[2] http://dublincore.org/
[3] http://www.geneontology.org/
[4] http://www.ebi.ac.uk/GOA/

very frequently, rendering the exported data stale and outdated, as mentioned above. Finally, we conducted a survey[5] of 100 databases listed on the NAR Journal to assess how wide-spread the adoption of direct programmatic access to databases was, such as through Web Service APIs. We discovered that only 11 large and well-established sources actually provide complete or partial access through APIs. In fact, we concluded that a Web interface was the only common access point for all databases.

In this paper, we propose to use Web interfaces of Biochemical sources to learn corresponding ontologies. This follows our previous work [10] on unsupervised wrapper generation for entire Biochemical Web sites to extract labeled data from multiple classes of pages. Together, these approaches can be used to automatically acquire ontologies and instantiate them with RDF data on-the-fly. We argue that such an approach would at least help to bootstrap the process which all of the projects mentioned above follow: that of converting Web-accessible data into linked RDF data. Additionally, our approach can be used to learn domain-specific ontologies, a task that is very challenging if performed on natural language text as opposed to labeled, (semi) structured data in our setting. This is very much in line with the vision of the self-annotating Web [11] where the Web is used to extract ontologies, which then provide semantic annotation to enable Semantic Web content creation.

Our approach to ontology learning relies on a set of transformation rules that we apply to the output of our wrapper-generation algorithms, resulting in an OWL ontology for the underlying database. We apply our algorithms to real-world online Biochemical Web sites. We perform a "hard" evaluation of our acquired ontologies against gold standard ontologies, and a "soft" evaluation using three human experts to rate T-Box statements, or axioms, from our ontologies. We also evaluate our ontologies against ontologies that are lifted directly from the relational schemata. Finally, we apply our approach to extract RDF data from three Web sources.

The rest of this paper is organized as follows. Section 2 briefly explains our wrapper generation algorithm and its output which is used for ontology learning. Section 3 presents our transformation rules which convert our wrapper algorithm's output into an ontology, while Section 4 presents the experiments that we conducted for ontology learning and their evaluations. Section 5 briefly describes the extraction of RDF data. Section 6 presents the related work, and we conclude in Section 7.

## 2 Wrapper Induction

This section presents a brief overview of our wrapper induction approach to extract labeled data from Biochemical Web sites. A complete description is available in [10].

[5] Available at http://sabiork.villa-bosch.de/ontology/servicesurvey.html

### 2.1 Page-Level Wrapper Induction

We observe that data on Biochemical Web sites are often labeled - that is, data entries on the Web pages are in proximity to descriptive text which serve the purpose of attribute names, such as "Mass" and "Temperature". Our survey[6] of 20 such Web sites revealed that, in fact, about 97% of data fields present on these Web pages were labeled. We utilize these labels to by-pass the page structure, which is dynamic and hence unpredictable, and pivot directly to the labels, and find a relative path in the DOM representation of the page to the corresponding data.

Our algorithm relies on multiple sample pages belonging to the same class - that is, having similar structure and content, and being generated from the same server-side script. We screen-scrape the individual text entries from each sample, and compare these entries across all samples. Disjoint entries are classified as data entries, where overlapping entries are a mixture of presentation text and possible labels. We then determine XML Path expressions (XPath expressions) for each text entry, from the root of the DOM tree to the node containing that entry. These XPaths are used to determine, for each data entry, the closest non-data entry, which serves as its label. Ultimately, our algorithm outputs a label with a corresponding relative XPath to the corresponding data entries. A final XPath expression resembles the form:

```
//*[text()='label']/../tr[1]/td[2]/a[1]/text()
```

The above XPath expression reads: Jump to the node which contains the text "label", follow a relative path in the DOM tree from this node, which points to the corresponding data. Our experiments indicated that the algorithm achieves a Precision of 99% and a Recall of 98% with about 9 samples.

### 2.2 Site-Wide Wrapper Induction

Data in Life Science Web sites are often scattered across many pages belonging to many different classes. In order to extract all data from underlying databases, we must learn wrappers for each of these classes. We tackle this problem using the concept of *Labeled Link Collections*. Link collections are hyperlink tag(s) which appear grouped together under the same parent node. A labeled link collection implies a link collection that has been associated with a label by our algorithm in Section 2.1, signifying these hyperlinks occur over data fields. We make two crucial observations about labeled link collections. Firstly, such collections point to data-rich pages, and secondly, all target pages of a labeled link collection belong to the same class. The latter observation allows us to automatically provide pages that have similar structure and content to our page-level wrapper induction algorithm.

The site-wide wrapper induction algorithm proceeds by learning a wrapper for the initial result page generated from probing a search form. It finds labeled link

[6] Available at http://sabiork.villa-bosch.de/ontology/labelsurvey.html

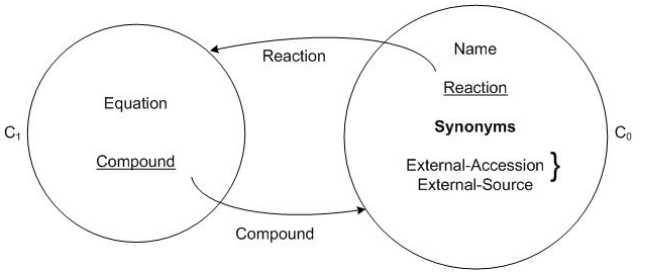



**Fig. 1.** Schematic representation of wrapper induction output

collections and follows them to their target pages, learning wrappers for those pages. Upon learning a new wrapper, it compares this new wrapper with existing wrappers by comparing the sequence of corresponding labels. If the sequence is the same, it implies the link collection points towards pages of the same class. This process is iterated for each new wrapper learnt, until data-rich pages of a Web site are explored.

This process essentially results in a Web site model as a directed labeled graph, schematically shown in Figure 1, discussed in more detail in Section 2.3. Experiments showed that the site-wide wrapper induction algorithm achieves a Precision of about 98% and a Recall of 76%.

### 2.3 Discussion of Wrapper Output

In this section, we discuss in more detail the output of our site-wide wrapper induction algorithm. The actual output is in XML, but is shown schematically in Figure 1, for a hypothetical source. The circles represent a class of pages, while their contents are the labels that were identified by our wrapper in these pages, together with XPaths of corresponding data (not shown). These label-data pairs have some additional meta-data associated with them. Certain label-data pairs have hyperlink tags on them, signifying that these are link collections (underlined in the figure), others have many data values associated with a single label (shown in bold) as opposed to some which have a single data value (normal font), while others are classified as composites (shown connected together). In addition, the figure also shows directed edges between classes of pages. For a given class of pages, an edge directed outwards represents the virtual link with a class of pages that can be reached when following a particular link collection. In Figure 1, the link collection "Reaction" in class $C_0$ when followed, leads to a class of pages $C_1$. The edge signifying this directed connection is named after the corresponding link collection "Reaction".

## 3 Ontology Learning

In this section we describe our transformation rules to convert the output presented in Section 2 into an ontology. We select OWL as the ontology language as

it is being used in current Semantic Web projects in the Life Science mentioned in Section 1. Finally, all the information required to determine the application of these rules is present in our wrapper output. Therefore, no supervision is required to apply these rules. In the subsequent sub-sections, we take Figure 1 as a running example and explain our transformation rules for constructing classes, data properties and object properties of an ontology.

### 3.1 Transformation Rules for Classes

We apply the following two rules for construction and naming of classes:

*CC - Class Construction:* Each page-class discovered by our wrapper induction algorithm is converted into a class in our ontology. This is based on our observation that a Web page typically represents a concept. In our example, this rule would construct two classes corresponding to the two page-classes discovered. In our example, two classes will be constructed corresponding to the two page-classes present.

*CN - Class Naming:* A class is named after the label of the link collection that points to it. The class corresponding to $C_1$ will be named "Reaction".

### 3.2 Transformation Rules for Properties

*DP - DataType Property Construction:* All label-data pairs discovered within a page-class that do not have hyperlink tags are converted into data type properties. The domain of these properties is the OWL class which corresponds to the page-class which contains these data-label pairs, and a suitable range is selected to describe the data. Currently, our implementation supports the XML Schema types string, integer and float. The choice between these types is based on a simple string parsing to determine whether the sample data contains characters, digits or decimals.

*OP - Object Property Construction:* All label-data pairs discovered within a page-class that have hyperlink tags are converted into object properties. The domain of these properties is the OWL class which corresponds to the page-class which contains these data-label pairs where the range is the target class to which the link collection points.

*PN - Property Naming:* The label of the corresponding data-label pair is assigned as the name of the property.

*PF - Setting a Property to Functional:* Labels discovered in the page-classes that have single data values associated with them are set as functional. This can be determined by examining the XPath expressions as described in Section 2.1. For example, for the Reaction class in Figure 1, we have the data type property

`Equation(Reaction, XSDString)`

An example of an object property is

`_Compound(Reaction, Compound)`

### 3.3 Transformation Rules for Composite Labels

Labels which appear in adjoining columns of a table have associated data values that fully describe their semantics when viewed together, which signifies an n-ary relation. Instead of constructing properties from these labels as shown in Section 3.2, we create an object property which has in its range a new class. This new class in turn is used to create functional properties from label-data pairs as shown in Section 3.2. We define the following rules to achieve this:

*CL.a - Class Construction from Composite Labels:* A class is created in the ontology for every set of composites labels.

*CL.b - Naming of Class Created from Composite Labels:* Labels are concatenated to form the name of such classes. In our running example, a class will be created with the name "`External_Accession-External_Source`".

*CL.c - Object Property Creation:* An object property is created which has a range of this new class created, and a domain of the OWL class which corresponds to the page-class which contains these data-label pairs.

Finally, transformation rules *DP* and *OP* are applied to create properties in the same manner as described in Section 3.2, except the domain of these properties is the new class that has been created by rule *CL.a*. The transformations described in this sub-section are applied for each set of composite label-data pairs that are identified by our wrapper induction algorithm.

## 4 Experiments and Evaluation

This section describes our experiments on 13 real-world Biochemical Deep Web sources, and the evaluation of the results we obtained. We implemented the transformation rules described in Section 3 in a Java program that accepts the XML output of our wrapper induction algorithms and applies these rules in an unsupervised manner to construct an OWL ontology.

The 13 online Biochemical databases that were targeted for ontology learning are presented in Table 1, together with the number of classes, object and data type properties discovered by our approach. These sources were selected as they provide basic qualitative data that is widely required in most specialized domains and are extensively used for annotation purposes by various other Web sites and in models. As Table 1 indicates, the number of classes discovered varies. In our wrapper-induction approach detailed in [10], each page type which describes data is converted into a class. The greater the number of pages across which the results are distributed, the greater will be the number of classes in the learned ontology. (In addition, some classes may also be introduced in the resulting ontology due to the transformation rules for composite labels, as described in Section 3.3). For instance, Rhea and SBO are relatively small databases and display all their results on a single result page. Therefore, the corresponding ontologies contain only a single class. On the other hand, Reactome and UniProt are large databases

**Table 1.** Targeted data sources & corresponding classes & properties of learned ontologies

| Database | Classes | Object Properties | Data Properties |
|---|---|---|---|
| Reactome http://www.reactome.org/ | 17 | 31 | 42 |
| PubChem http://pubchem.ncbi.nlm.nih.gov | 10 | 16 | 61 |
| Rhea http://www.ebi.ac.uk/rhea/ | 1 | 1 | 5 |
| PDB (USA) http://www.rcsb.org/ | 8 | 12 | 55 |
| PDBe http://www.ebi.ac.uk/pdbe/ | 3 | 3 | 21 |
| UniProt http://www.uniprot.org/ | 12 | 15 | 68 |
| KEGG http://www.genome.jp/kegg/ | 7 | 18 | 26 |
| IntAct www.ebi.ac.uk/intact/ | 7 | 9 | 23 |
| SABIO-RK http://sabio.villa-bosch.de/ | 18 | 22 | 46 |
| ChEBI http://www.ebi.ac.uk/chebi/ | 8 | 9 | 24 |
| SBO http://www.ebi.ac.uk/sbo/ | 1 | 1 | 8 |
| IntEnz http://www.ebi.ac.uk/intenz/ | 1 | 0 | 10 |
| MSDChem http://ebi.ac.uk/msd-srv/chempdb/ | 1 | 1 | 22 |

with the results spread across many different types of pages. This fact, together with the presence of composite labels, results in many classes in corresponding ontologies. We perform three evaluations for a selection of the ontologies that we learn. Firstly, we perform a "hard" evaluation against a gold standard ontology. This evaluation covers the lexical term layer and the concept hierarchy evaluation. The second "soft" evaluation is done with the help of domain experts which evaluate T-Box axioms from our ontologies in order to evaluate data and object properties (relations), as well as class and relation names. Thirdly, we "lift" ontologies from actual relational schemata, where available, using an exiting unsupervised approach. We then ask domain experts to rate T-Box axioms from these "lifted" ontologies against corresponding ontologies that are learned using our approach. These evaluations are presented subsequently in Sections 4.2, 4.3 and 4.4 respectively. Section 4.1 briefly describes the process of obtaining gold-standard ontologies for the three target sources. We provide an analysis of our results in Section 4.5, highlighting the limitations of our approach.

### 4.1 Acquiring Gold Standard Ontologies

As mentioned above, we are interested in evaluating T-Box style axioms from our learned ontologies. (Note that the evaluation of the extracted data has already been performed in [10] for our wrapper-induction algorithm). In order to do this, we need corresponding T-Box style gold standard ontologies. However, existing relational databases rarely, if at all, provide corresponding ontologies.

In the Bio2RDF project a global ontology describing many namespaces corresponding to various data sources was manually developed[7]. We extract the

[7] Available at http://bio2rdf.org/bio2rdf.owl

classes and relations in the KEGG and PDB namespace from this global ontology to use as our gold standard. In case of ChEBI and SBO, only A-Box style ontologies containing assertions about individuals are available in various formats, including OWL. We manually reverse engineer a T-Box from these ontologies to use as gold standard for these two sources respectively.

### 4.2 Evaluation against a Gold Standard

We use the framework described by Dellschaft et al. [12] to evaluate a learned ontology against a gold standard. We provide a brief overview of the evaluation measures here, and refer an interested reader to [12] for details. They provide measures for evaluating the lexical term layer and taxonomy of an ontology. For lexical evaluation, these measures include lexical precision and lexical recall, borrowed from [13] and defined as:

$$LP(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_C|} \quad , \quad LR(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_R|} \tag{1}$$

Where $O_C$ and $O_R$ are the computed and reference ontologies respectively, and $C_C$ and $C_R$ are concepts in these ontologies, identified by their names.

For taxonomic evaluation, they provide precision and recall measures which are based on using the common semantic cotopy as the characteristic extract of concepts in a taxonomy. A characteristic extract of a concept characterizes the position of a concept in a hierarchy, which can be used to determine local taxonomic precision for that concept. This in turn is used as a building block for global taxonomic evaluation of the ontology. By advising to use the common semantic cotopy, Dellschaft et al. effectively diminishing the influence of the lexical layer in the evaluation of the taxonomy, by removing concepts which are (lexically) different from the semantic cotopy, and only including common concepts in the semantic cotopy. The taxonomic precision and recall are, thus, defined as

$$TP_{SC}(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} \begin{cases} tp_{sc}(c, c, O_C, O_R) & if c \in C_R \\ 0 & if c \notin C_R \end{cases} \tag{2}$$

$$TR_{SC}(O_C, O_R) := TP_{SC}(O_R, O_C) \tag{3}$$

Where

$$tp_{sc}(c_1, c_2, O_C, O_R) := \frac{|ce(c_1, O_C) \cap ce(c_2, O_R)|}{|ce(c_1, O_C)|} \tag{4}$$

is the local taxonomic precision of a concept. The F-measures are then given by:

$$TF(O_C, O_R) := \frac{2 \cdot TP(O_C, O_R) \cdot TR(O_C, O_R)}{TP(O_C, O_R) + TR(O_C, O_R)} \tag{5}$$

$$TF'(O_C, O_R) := \frac{2 \cdot LR(O_C, O_R) \cdot TF(O_C, O_R)}{LR(O_C, O_R) + TF(O_C, O_R)} \quad (6)$$

Where $TF'$ combines the lexical level and taxonomic evaluation in a single value.

We apply these measures on our learned ontologies comparing them to reference ontologies obtained in Section 4.1. The results are presented in Table 2.

**Table 2.** Results of evaluation against reference ontologies

| | $LP$ | $LR$ | $TP$ | $TR$ | $TF$ | $TF'$ |
|---|---|---|---|---|---|---|
| KEGG | 0.428 | 0.428 | 0.428 | 0.357 | 0.39 | 0.4 |
| ChEBI | 0.13 | 1.0 | 0.125 | 1.0 | 0.22 | 0.36 |
| SBO | 0.5 | 1.0 | 0.5 | 1.0 | 0.6 | 0.8 |
| PDB USA | 0.13 | 1.0 | 0.125 | 1.0 | 0.22 | 0.36 |

*Discussion:* The lexical and taxonomic recall for our approach in case of ChEBI, SBO and PDB in Table 2 are perfect. This is because we are successfully able to discover the classes in the gold standard ontologies. The lower precision is explained by the fact that we construct additional classes (7 in the case of ChEBI and PDB, 1 in the case of SBO) in our ontologies. However, in the case of ChEBI and SBO all of these classes are constructed to express n-ary relationships between composite attributes that we discover during the wrapper induction phase. Therefore, although our approach strives to maintain the semantics of the data, it suffers with regard to the taxonomic precision due to this. In the case of PDB, the Bio2RDF ontology omits certain data offered by this source. Hence the gold standard ontology contains fewer classes than our approach can discover from the Web interface, resulting in lower precision. The results for KEGG are mixed; there are some classes we discover correctly, while others we are not able to discover. This is indicated by the relatively equal precision and recall values for both the lexical and taxonomic results. It should be mentioned that these results are quite comparable to human inter-ontology building [28].

### 4.3 Evaluation by Domain Experts

The evaluation measures in Section 4.2 have served to evaluate the names and taxonomy of classes. They do not address the data and object properties that are defined for these classes. For scientific data, especially Life Science data, this is crucial, as entities in such domains have rich relationships with each other. In this section, we describe the results of a "soft" evaluation, where we ask domain experts to rate axioms in our ontologies in order to judge how well we were able to determine these relations. These T-Box statements are of the following form

```
Synonym(Gene, XSDString)
Class RPair
```

We ask three domain experts to rate 51 such statements from our KEGG ontology, 41 from ChEBI, 11 from SBO and 54 statements from our PDB ontology. We allow them to refer to corresponding Web sites, so that they can compare our findings against what they would have selected manually from these sites. We ask our experts to classify these statements into four categories based on the degree of correctness: Wrong, Slightly Correct, Mostly Correct, and Correct. They are instructed to denote these categories for each statement by assigning values of 0, 1, 2 and 3 respectively. We then use these ratings to determine our precision and recall at three distinct points. First, using our experts' ratings we determine the categorical agreement between them. We calculate the Kappa statistic from the ratings of each ontology, using the method described in [14] for multiple raters rating into multiple categories. The Kappa values were measured to be 42.24%, 27.42%, 13.10% and 47.31% for KEGG, ChEBI, SBO and PDB respectively, which indicates that there is a low agreement. In addition, Kappa values generally tend to decrease as the number of categories increases, as in our case. The low agreement might also suggest that the ontology construction task is inherently hard or not well defined. For our evaluation, we count the data points where all three raters agree on the categorization of a given statement from our ontology, and determine the precision for each category. Our true-positives for a given categorization are the number of statements unanimously classified into this category by our raters. The false-positives are the number of statements unanimously classified into some other category. Therefore, we determine precisions for Correct (C), Mostly Correct (MC) and Slightly Correct (SC) statements, with results shown in Table 3. For completely correct statements, we have an average precision of 53.25%. There can be different factors for this, such as incorrect discovery of label-data pairs by our wrapper induction algorithm, or incorrect or unsuitable assignment of class and property names (Recall from Sections 3.2 and 3.3 that names are taken from corresponding labels of link-collections).

### 4.4 Evaluation against Lifted Ontologies

We are interested in evaluating the results of our approach to extracting ontologies from Web interfaces against ontologies that are "lifted" from direct access to the underlying relational schema using existing unsupervised approaches such as [18] or [19], which apply transformation rules to a relational schema and reverse-engineer an ontology. We opt to use the approach described in [19] as it constructs ontologies in OWL. However, from our selection of 13 databases, we are only able to get direct access to the relational schemata for two sources, namely SABIO-RK and ChEBI. We therefore lift ontologies from these schemata and perform a manual evaluation of our learned ontologies with the help of a domain expert. We provide sets of 20 corresponding statements from the lifted and learned ontologies, and instruct the expert to give a score of 1 to the superior statement and a score of 0 to the poorer statement, taking into consideration both syntax and semantics. (A score of 1 is given to both statements if they are deemed equally valid). The results are shown in Table 4, which presents the total score for corresponding lifted and learned ontologies.

**Table 3.** Precision values for three categorizations of statement. P(C) is precision of statement being completely correct, P(MC) is precision of statement either completely or mostly correct, P(SC) is precision of statement completely, mostly or slightly correct.

| Sources | P(C) | P(MC) | P(SC) |
|---|---|---|---|
| KEGG | 43.8% | 84.4% | 90.7% |
| ChEBI | 66.6% | 80.1% | 88.4% |
| SBO | 50% | 87.5% | 87.5% |
| PDB USA | 52.6% | 90.5% | 92.3% |

**Table 4.** The total score for lifted and learned ontologies

| Sources | Total Score for Lifted Ontology | Total Score for Learned Ontology |
|---|---|---|
| SABIO-RK | 9 | 16 |
| ChEBI | 19 | 14 |

*Discussion:* We note here that for SABIO-RK, 34 classes are constructed in the ontology lifted directly from the relational schema compared to the 18 classes constructed using our approach. This is because portions of the schema are not normalized, as well as the fact that the schema contains certain data which is not displayed in the Web interface at all. The ontology lifted from this schema also suffers lexically as the table and attribute names are often obscure and unnatural. On the other hand, both the ontologies for ChEBI consist of 8 classes, and the schema comprises of well-named tables and attributes, resulting in a very high score for the lifted ontology.

### 4.5 Analysis of Results

In this sub-section we present a brief analysis of the overall results and the conclusions that we can derive from it. Firstly, our transformation rules do not assist us in creating rich taxonomies. Therefore, we would expect to achieve poor taxonomic precision and recall if the reference ontology has a rich taxonomy, which was not the case in our experiments: Only KEGG has two sub-class relationships. A possible future improvement in our approach would be to use labels within page-classes in our wrapper output to determine such relations. One could determine such taxonomic relations if the set of labels of a given class is a subset of that of another class, for example, by using Formal Concept Analysis [27].

Secondly, our results from Section 4.4 indicate that extraction of ontologies from Web interfaces is a viable option especially in the absence of relational schemata. Relational schemata often suffer from not being in normalized forms, either for ease of representation or performance, and may follow ambiguous naming conventions as these are private and internal representations.

Finally, the manual approaches for ontology construction described in Section 1 tend to re-use existing ontologies, such as FOAF[8] and Dublin Core.

[8] http://www.foaf-project.org/

This is extremely beneficial, as it helps to eventually integrate the data from various sources. However, this is not possible in our approach. This results in lexical differences between manually constructed ontologies and those generated by using our approach, which affects precision and recall.

## 5 RDF Data Extraction

In this section we briefly describe extraction of RDF triples from Web interfaces for which we have learned corresponding ontologies. Since the transformation rules for our ontology learning approach are directly mapped from the output of our wrapper induction algorithm in [10], we can directly extract RDF data by applying our wrappers and populating the corresponding ontology. Since we wish to merely demonstrate the feasibility of extracting data from Web pages, we select three sources of varying size, namely KEGG, ChEBI and MSDChem, and execute corresponding wrappers on the result pages. We utilize the lists of identifiers provided by each source on its Web site to probe the Web forms and generate the result pages. The triples thus generated are stored in a local Sesame[9] repository. In all, approximately 300MB, 200MB and 25MB of data are extracted for KEGG, ChEBI and MSDChem respectively. The running times for applying the wrappers for these sources were approximately 7, 5 and 1 hours respectively, using a 6 Mbps internet connection.

## 6 Related Work

Although there are many approaches to ontology learning from natural language text, we are only aware of a handful of approaches which target Deep Web sites containing semi-structured data as a source for ontology learning. In [15], the authors describe a system which extracts attributes of Web search forms belonging to a certain domain, such as tourism and e-commerce. WordNet is used to find hyponyms for these attributes iteratively until a taxonomy of concepts is generated using only the IS-A relation. OntoBuilder [16] also constructs a taxonomy of concepts by parsing Web search forms, although the authors do not describe the algorithm or the resulting taxonomy in detail. OntoMiner [17] learns a taxonomy of concepts from co-domain Web sites. It relies on HTML regularities to construct taxonomic structures in XML by utilizing a hierarchical partition algorithm. The taxonomy is iteratively expanded with sub-concepts by crawling through links corresponding to concepts in the taxonomy. Another related approach is used in Triplify [29], which facilitates linked RDF data generation for Web site publishers by mapping HTTP-URI requests onto relational queries, the results of which are transformed into RDF.

A closely related area of research is Database-to-ontology conversion, whereby transformation rules similar to ours are applied directly to a relational database schema and corresponding ontology is extracted [18,19], or mapping rules are generated between an existing ontology and the underlying relational database,

[9] http://www.openrdf.org/

such as in D2RQ [7]. However, in our setting, we do not have access to the actual database implementations. Another related research topic is that of identification of HTML tables and transforming them into, for instance, logical frames as in TARTAR [20] or relational schemes as in WebTables [21]. However, such approaches neglect data on Web pages that are not structured within HTML tables. Finally, there has been considerable research using XML documents to extract relational schema [22,23], DTDs [24], or XML Schemata [25,26].

## 7 Conclusions

We have presented an unsupervised approach for extracting ontologies and RDF data from Deep Web Life Science databases. Our approach relies on transformation rules that convert XML output from our wrapper induction algorithm into OWL ontologies. Experiments were conducted on real-world online Biochemical Web sites. Our results indicate that this approach can be used to bootstrap the process of converting legacy and freely available online data into machine-procesable data for use in Semantic Web applications. Our results support our argument that the Deep Web, with its semi-structured content, is an ideal source for learning ontologies to help overcome the bottleneck for wider adoption of Semantic Web technologies. For future work, we would like to re-use existing ontologies during the ontology learning phase in order to facilitate the eventual integration of the extracted data into the Linked Open Data cloud.

## References

1. Galperin, M.Y., Cochrane, G.R.: Nucleic Acids Research annual Database Issue and the NAR online Molecular Biology Database Collection in 2009. Nucleic Acids Res. 37(Database issue), 1–4 (2009)
2. Cheung, K.H., et al.: YeastHub: a semantic web use case for integrating data in the life sciences domain. Bioinformatics 21(suppl. 1) (June 1, 2005)
3. Baker, C.J.O., et al.: Semantic Web infrastructure for fungal enzyme biotechnologists. Journal of Web Semantics 3(4) (2006)
4. Stephens, S., LaVigna, D., Dilascio, M., Luciano, J.: Aggregation of bioinformatics data using semantic web technology. Journal of Web Semantics, 4 (2006)
5. Pasquier, C.: Biological data integration using Semantic Web technologies. Biochimie 90(4), 584–594 (2008)
6. Zhao, J., Miles, A., Klyne, G., Shotton, D.: OpenFlyData: The Way to Go for Biological Data Integration. In: Paton, N.W., Missier, P., Hedeler, C. (eds.) Data Integration in the Life Sciences. LNCS (LNBI), vol. 5647, pp. 47–54. Springer, Heidelberg (2009)
7. Bizer, C.: D2RQ - treating non-RDF databases as virtual RDF graphs. In: Proceedings of the 3rd International Semantic Web Conference ISWC 2004 (2004)
8. Belleau, F., et al.: Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. In: WWW 2007, Banff, Canada (2007)
9. Reyes-Palomares, A., et al.: Systems Biology Metabolic Modeling Assistant (SBMM): An ontology-based tool for the integration of metabolic data in kinetic modeling. Bioinformatics, doi:10.1093/bioinformatics/btp061

10. Mir, S., Staab, S., Rojas, I.: Site-Wide Wrapper Induction for Life Science Deep Web Databases. In: Paton, N.W., Missier, P., Hedeler, C. (eds.) DILS 2009. LNCS, vol. 5647, pp. 96–112. Springer, Heidelberg (2009)
11. Cimiano, P., Handschuh, S., Staab, S.: Towards the Sself-Annotating Web. In: WWW 2004: Proceedings of the 13th International Conference on World Wide Web, pp. 462–471. ACM Press, New York (2004)
12. 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)
13. Sabou, M., Wroe, C., Goble, C., Mishne, G.: Learning domain ontologies for web service descriptions: an experiment in bioinformatics. In: Proc. of WWW 2005 (2005)
14. Green, A.M.: Kappa statistics for multiple raters using categorical classifications. In: Proceedings of the Twenty-Second Annual Conference of SAS Users Group (1997)
15. An, Y.J., et al.: Automatic Generation of Ontology from the Deep Web. In: DEXA Workshops 2007, pp. 470–474 (2007)
16. Roitman, H., Gal, A.: OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources using Sequence Semantics. In: Proceedings of the International Conference on Semantics of a Networked World, ICSNW 2006 (2006)
17. Davalcu, H., Vadrevu, S., Nagarajan, S., Ramakrishnan, I.: Ontominer: bootstrapping and populating ontologies from domain-specific web sites. IEEE Intelligent Systems 18(5), 24–33 (2003)
18. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive Web Sites into the Semantic Web. In: Proc. of the 17th symposium on Proceedings of the 2002 ACM Symposium on Applied Computing, SAC 2002, Madrid Spain, pp. 1100–1107 (2002)
19. Li, M., Du, X.-Y., Wang, S.: Learning ontology from relational database. In: Proceedings of International Conference on Machine Learning and Cybernetics (2005)
20. Pivk, A., et al.: Transforming arbitrary Tables into F-Logic Frames with TARTAR. Data & Knowledge Engineering (DKE) 60(3), 567–595 (2007)
21. Cafarella, M.J., et al.: Uncovering the relational Web. In: WebDB 2008 (2008)
22. Deutsch, A., Fernandez, M., Suciu, D.: Storing Semistructured Data in Relations. In: Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats (1998)
23. Cohen, S., Kanza, Y., Sagiv, Y.: Generating Relations from XML Documents. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 282–296. Springer, Heidelberg (2002)
24. Garofalakis, M.N., et al.: DTD inference from XML documents: The xtract approach. IEEE Data Eng. Bull. 26(3), 19–25 (2003)
25. Hegewald, J., Naumann, F., Weis, M.: XStruct: Efficient Schema Extraction from Multiple and Large XML Documents. In: Data Engineering Workshop, 22nd International Conference on Data Engineering Workshops (ICDEW 2006), p. 81 (2006)
26. Bex, G.J., Neven, F., Vansummeren, S.: Inferring XML schema definitions from XML data. In: VLDB 2007, pp. 998–1009 (2007)
27. Cimiano, P., Hotho, A., Staab, S.: Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. J. Artif. Intell. Res. (JAIR) 24, 305–339 (2005)
28. Maedche, A., Staab, S.: Measuring Similarity between Ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, p. 251. Springer, Heidelberg (2002)
29. Auer, S., et al.: Triplify - Light-Weight Linked Data Publication from Relational Databases. In: 18th International World Wide Web Conference, pp. 621–621

# Leveraging Terminological Structure for Object Reconciliation

Jan Noessner, Mathias Niepert,
Christian Meilicke, and Heiner Stuckenschmidt

KR & KM Research Group
University of Mannheim, B6 26, 68159 Mannheim, Germany
{jan,mathias,christian,heiner}@informatik.uni-mannheim.de

**Abstract.** It has been argued that linked open data is the major benefit of semantic technologies for the web as it provides a huge amount of structured data that can be accessed in a more effective way than web pages. While linked open data avoids many problems connected with the use of expressive ontologies such as the knowledge acquisition bottleneck, data heterogeneity remains a challenging problem. In particular, identical objects may be referred to by different URIs in different data sets. Identifying such representations of the same object is called object reconciliation. In this paper, we propose a novel approach to object reconciliation that is based on an existing semantic similarity measure for linked data. We adapt the measure to the object reconciliation problem, present exact and approximate algorithms that efficiently implement the methods, and provide a systematic experimental evaluation based on a benchmark dataset. As our main result, we show that the use of light-weight ontologies and schema information significantly improves object reconciliation in the context of linked open data.

## 1 Introduction

There is an ongoing debate concerning the role of ontologies for the semantic web. While rich ontologies have been promoted as an integral part of every semantic web application [11], it is increasingly argued that the real value of the semantic web is based on its ability to create and maintain linked open data which provides effective access to semantically enhanced information on the web [21]. In this paper, we argue that the use of (light-weight) ontologies helps to solve one of the key problems of linked open data on the web, namely, the actual linking of data by identifying different representations of the same object. This problem has been extensively studied in the context of database systems as duplicate detection, record linkage, and object or reference reconciliation [13]. Most existing work has focused on the design of specialized measures which estimate the similarity of objects based on their lexical properties. The Silk framework [22], for instance, combines lexical similarity measures in order to create links between objects. The use of schema information in the context of formal ontologies has only recently been proposed [15,10]. In this work, we leverage schema information

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 334–348, 2010.

to exclude logically inconsistent links between objects and to improve the overall accuracy of instance alignments. In particular, we use logical reasoning and linear optimization techniques to compute the overlap of derivable types of objects. This information is combined with the classical similarity-based approach, resulting in a novel framework for object reconciliation. Our contributions are the following:

- We combine classical similarity measures for object reconciliation with a semantic similarity measure that takes schema information into account;
- We show that the combined approach clearly outperforms methods that do not consider schema information;
- We present efficient ways of computing the combined similarity measures based on a formulation as an integer linear programming problem; and
- We show that the method can be efficiently implemented using an approximate algorithm with only a modest loss of precision and recall.

The paper is organized as follows. In Section 2 we discuss the object reconciliation problem in more detail and refer to existing work in this area. Section 3 extends and adapts the similarity measure proposed in [19] to the problem of object reconciliation. In Section 4, we show that computing the maximal similarity between objects in two datasets can be formulated as an optimization problem. In particular, we show that there exists a transformation to a linear integer programming problem, the solution of which corresponds to the alignment that maximizes the semantic similarity between the datasets. In addition, we apply an existing approximative graph matching algorithm to the problem. Finally, in Section 5, we show that both the optimal and the approximate algorithms result in high-quality alignments both in terms of precision and recall.

## 2 Problem Statement and Related Work

The problem of object reconciliation has been a topic of research for more than 50 years. It is also known as the problem of record linkage [8], entity resolution [1], and instance matching [9]. While the majority of the existing methods were developed for the task of matching database records, modern approaches focus mostly on graph-based data representations extended by additional schema information. We discuss the problem of object reconciliation using the notion of instance matching. This allows us to describe it within the well-established ontology matching framework [7]. Ontology matching is the process of detecting links between entities in different ontologies. These links are annotated by a confidence value and a label describing the type of link. Such a link is referred to as a *correspondence* and a set of such correspondences is referred to as an *alignment*.

**Definition 1 (Correspondence and Alignment).** *Given ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, let $q$ be a function that defines sets of matchable entities $q(\mathcal{O}_1)$ and $q(\mathcal{O}_1)$. A correspondence between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a four tuple $\langle e_1, e_2, r, n\rangle$ such that $e_1 \in$*

*$q(\mathcal{O}_1)$ and $e_2 \in q(\mathcal{O}_2)$, $r$ is a semantic relation and $n$ is a confidence value. An alignment $\mathcal{M}$ between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a set of correspondences between $\mathcal{O}_1$ and $\mathcal{O}_2$.*

The generic form of Definition 1 captures a wide range of correspondences by varying what is admissible as matchable element, semantic relation, and confidence value. A fundamental distinction between different matching tasks is determined by the restriction $q$ on the set of matchable entities. On the one hand we might be interested in links between terminological entities (concepts and properties) and on the other hand we might want to find links between instances. In the following we refer to an alignment that contains correspondences of the former type as *terminological alignment* and to an alignment that contains correspondences of the latter type as *instance alignment*. Terminological alignments relate the T-Boxes of $\mathcal{O}_1$ and $\mathcal{O}_2$ by providing equivalence or subsumption links between concepts and properties. Since instance matching is the task of detecting pairs of instances that refer to the same real world object [9], the semantic relation expressed by an instance correspondence is that of identity. The confidence value of a correspondence quantifies the degree of trust in the correctness of the statement. If a correspondence is automatically generated by a matching system this value will be computed by aggregating scores from different sources of evidence. The commonly applied methods for object reconciliation include structure-based strategies as well as strategies to compute and aggregate value similarities. Under the notion of instance matching, similarities between instance labels and datatype properties are mostly used to compute confidence values for instance correspondences. Examples of this are realized in the systems RiMOM [23] and OKKAM [18]. Additional refinements are related to a distinction between different types of properties. The developers of RiMOM manually distinguish between *necessary* and *sufficient* datatype properties. The FBEM algorithm of the OKKAM project assigns higher weights to certain properties like names and IDs. In both cases, the employed methods focus on appropriate techniques to interpret and aggregate similarity scores based on a comparison of datatype property values. Another important source of evidence is the knowledge encoded in the T-Box. RiMOM, for example, first generates a terminological alignment between the T-Boxes $\mathcal{T}_1$ and $\mathcal{T}_2$ describing the A-Boxes $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. This alignment is then used as a filter and only correspondences that link instances of equivalent concepts are considered valid [23].

In this paper we are concerned with the scenario where both A-Boxes are described in terms of the same T-Box. An object reconciliation method applicable to this setting is also proposed in [15] where the authors combine logical with numerical methods. For logical reasons it is in some cases possible to preclude that two instances refer to the same object while in other cases the acceptance of one correspondence directly entails the acceptance of another. The authors extend this approach by modeling some of these dependencies into a similarity propagation framework. However, their approach requires a rich schema and assumes that properties are defined to be functional and/or inverse functional. Hence, the approach cannot be used effectively to exploit type information based on a concept hierarchy and is therefore not applicable in many web of data

scenarios. In contrast, our approach does not rely on specific types of axioms or a set of predefined rules but on a well defined semantic similarity measure. A number of different approaches to quantify the degree of similarity between concept descriptions and ontologies have been proposed [2]. In particular, our approach is based on the measure proposed by Stuckenschmidt [19]. This measure has originally been designed to quantify the similarity between two ontologies that describe the same set of objects. We apply a modified variant of this measure to evaluate the similarity of two A-Boxes described in terms of the same T-Box. Furthermore, our method factors in a-priori confidence values that quantify the degree of trust one has in the correctness of the object correspondences based on lexical properties. The resulting similarity measure is used to determine an instance alignment that induces the highest agreement of object assertions in $\mathcal{A}_1$ and $\mathcal{A}_2$ with respect to $\mathcal{T}$.

## 3 A Similarity Measure for Instance Matching

In [19] Stuckenschmidt introduces a measure that quantifies the similarity of two A-Boxes described in terms of the same T-Box. A brief description is given in Section 3.1. In Section 3.2 we propose a modification of this measure that factors in a-priori confidence values. We argue that the underlying idea of the measure can be used to appropriately incorporate T-Box information during the matching process. Additionally, we explain and motivate our approach by means of an example. In the following, we will use $\langle a, b\rangle$ to refer to an instance correspondence $\langle a, b, =, n\rangle$ and the a-priori similarity $\sigma(a, b)$ to refer to the confidence value $n$.

### 3.1 Measuring A-Box Similarity

Stuckenschmidt's similarity measure is based on the notion of a *valid* instance alignment. Given an instance alignment $\mathcal{M}$ between $\mathcal{A}_1$ and $\mathcal{A}_2$, suppose that we merge both $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{T}$, and $\mathcal{M}$ into a single ontology $\mathcal{O}$. Due to some mismatches in $\mathcal{M}$ it might happen that $\mathcal{O}$ becomes inconsistent. Obviously, we want to avoid alignments that lead to inconsistencies. The following definition formally introduces the notion of a valid alignment.

**Definition 2 (Valid Alignment).** *Let $\mathcal{M}$ be an instance alignment between A-Boxes $\mathcal{A}_1$ and $\mathcal{A}_2$ both described in terms of T-Box $\mathcal{T}$. $\mathcal{M}$ is valid with respect to $\mathcal{T}$ if and only if for all concepts $C$ and all properties $P$ defined in $\mathcal{T}$ as well as for all correspondences $\langle a, b\rangle, \langle a', b'\rangle \in \mathcal{M}$ we have*

$$\begin{aligned}
\mathcal{T} \cup \mathcal{A}_1 \models C(a) &\Rightarrow \mathcal{T} \cup \mathcal{A}_2 \not\models \neg C(b) \\
\mathcal{T} \cup \mathcal{A}_2 \models C(b) &\Rightarrow \mathcal{T} \cup \mathcal{A}_1 \not\models \neg C(a) \\
\mathcal{T} \cup \mathcal{A}_1 \models P(a, a') &\Rightarrow \mathcal{T} \cup \mathcal{A}_2 \not\models \neg P(b, b') \\
\mathcal{T} \cup \mathcal{A}_2 \models P(b, b') &\Rightarrow \mathcal{T} \cup \mathcal{A}_1 \not\models \neg P(a, a')
\end{aligned}$$

Under the assumption that two different URI references in the same A-Box denote two distinct instances, a *valid* alignment will not lead to inconsistencies in the merged ontology. We now introduce the notion of a *functional one-to-one* alignment between A-Boxes. $\mathcal{M}$ is a functional one-to-one alignment if and only if for all pairs of correspondences $\langle a, b\rangle \neq \langle a', b'\rangle \in \mathcal{M}$ we have $a \neq a'$ and $b \neq b'$. Based on the notion of a valid functional one-to-one alignment one can count, for each possible alignment $\mathcal{M}$, the number of assertions identical in $\mathcal{A}_1$ and $\mathcal{A}_2$, where instance equivalence is determined by the alignment $\mathcal{M}$. We will call this value the *overlap* of two A-Boxes $\mathcal{A}_1$ and $\mathcal{A}_2$ induced by $\mathcal{M}$.

**Definition 3 (Overlap).** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be A-Boxes described in terms of T-Box $\mathcal{T}$. Furthermore, let $\mathcal{M}$ be a functional one-to-one*[1] *instance alignment between $\mathcal{A}_1$ and $\mathcal{A}_2$ that is valid with respect to $\mathcal{T}$. The overlap of $\mathcal{A}_1$ and $\mathcal{A}_2$ induced by $\mathcal{M}$ with respect to $\mathcal{T}$ is defined as*

$$
\begin{aligned}
&overlap_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{M}) := \\
&\big|\{C(a) \quad | \mathcal{T} \cup \mathcal{A}_1 \models C(a) \ \wedge\ \mathcal{T} \cup \mathcal{A}_2 \models C(b) \ \wedge\ \langle a, b\rangle \in \mathcal{M}\} \cup \\
&\ \{\neg C(a) \quad | \mathcal{T} \cup \mathcal{A}_1 \models \neg C(a) \ \wedge\ \mathcal{T} \cup \mathcal{A}_2 \models \neg C(b) \ \wedge\ \langle a, b\rangle \in \mathcal{M}\} \cup \\
&\ \{P(a, a') \ | \mathcal{T} \cup \mathcal{A}_1 \models P(a, a') \ \wedge\ \mathcal{T} \cup \mathcal{A}_2 \models P(b, b') \ \wedge\ \langle a, b\rangle, \langle a', b'\rangle \in \mathcal{M}\} \cup \\
&\ \{\neg P(a, a') \mid \mathcal{T} \cup \mathcal{A}_1 \models \neg P(a, a') \ \wedge\ \mathcal{T} \cup \mathcal{A}_2 \models \neg P(b, b') \ \wedge\ \langle a, b\rangle, \langle a', b'\rangle \in \mathcal{M}\}\big|
\end{aligned}
$$

Based on this, it is possible to define the A-Box similarity between $\mathcal{A}_1$ and $\mathcal{A}_2$ as the maximal possible overlap of $\mathcal{A}_1$ and $\mathcal{A}_2$. In order to find this value, we have to consider the set of all possible valid functional one-to-one alignments $\mathbb{M}$ between $\mathcal{A}_1$ and $\mathcal{A}_2$. Notice that the overlap is not only determined by $\mathcal{M}$ but also by the size of $\mathcal{A}_1$, $\mathcal{A}_2$ (number of instances), and $\mathcal{T}$ (number of concepts and properties). Thus, we have to use a normalizing denominator. The resulting similarity measure quantifies the degree of similarity as a value in the interval $[0, 1]$.

**Definition 4 (A-Box Similarity).** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be A-Boxes described in terms of T-Box $\mathcal{T}$. Furthermore, let $\mathbb{M}$ be the set of all functional one-to-one instance alignments between $\mathcal{A}_1$ and $\mathcal{A}_2$ that are valid with respect to $\mathcal{T}$. The A-Box similarity between $\mathcal{A}_1$ and $\mathcal{A}_2$ with respect to $\mathcal{T}$ is defined as*

$$
sim_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2) := \max_{\mathcal{M} \in \mathbb{M}} \frac{2 * overlap_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{M})}{overlap_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_1, \mathcal{I}_{\mathcal{A}_1}) + overlap_{\mathcal{T}}(\mathcal{A}_2, \mathcal{A}_2, \mathcal{I}_{\mathcal{A}_2})}
$$

*where $\mathcal{I}_{\mathcal{A}}$ refers to the identity alignment that maps every instance described in an A-Box $\mathcal{A}$ on itself.*

Notice that this similarity measure fulfills the properties of a conceptual similarity measure as defined by Amato et al. [5]. In particular, we have $0 \leq sim_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2) \leq 1$, $sim_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2) = sim_{\mathcal{T}}(\mathcal{A}_2, \mathcal{A}_1)$, and $sim_{\mathcal{T}}(\mathcal{A}, \mathcal{A}) = 1$.

[1] The approach is not limited to functional one-to-one alignments but can also generate m-to-n alignments. To simplify the exposition of the framework, however, we chose to describe it with respect to functional one-to-one alignments.

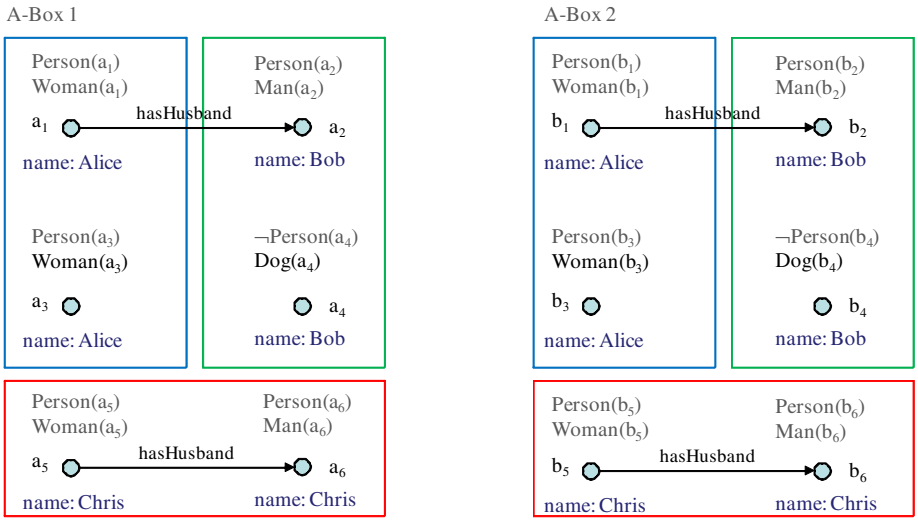



**Fig. 1.** Motivating example

### 3.2 Exploiting A-Box Similarity

In this section, we leverage the A-Box similarity from Definition 4 for the task of object reconciliation. Furthermore, we demonstrate the advantage of our method over those approaches using only lexical confidence values. Therefore, we introduce a small motivating example. Suppose that the shared T-Box $\mathcal{T}$ is defined as follows.

$$\begin{aligned} \exists hasHusband &\sqsubseteq Woman \\ \exists hasHusband^{-} &\sqsubseteq Man \\ Dog &\sqsubseteq \neg Person \\ Person &\equiv Woman \sqcup Man \end{aligned}$$

Let us assume we have six individuals $a_1, ..., a_6$ and $b_1, ..., b_6$ in each A-Box. Furthermore, let us assume that the following concept and object property assertions are explicitly specified in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

$$\begin{aligned} hasHusband(a_1, a_2) \quad & hasHusband(b_1, b_2) \\ hasHusband(a_5, a_6) \quad & hasHusband(b_5, b_6) \\ Woman(a_3) \quad & Woman(b_3) \\ Dog(a_4) \quad & Dog(b_4) \end{aligned}$$

Figure 1 provides an illustration of this example. Stated assertions are depicted in black, while gray-colored assertions can be inferred from the given ones with respect to the T-Box $\mathcal{T}$. To simplify the notation, we base the a-priori confidence value $\sigma(a, b)$ of a correspondence $\langle a, b \rangle$ on the identity of the *name* datatype-property value by setting the lexical similarity to 1 if the strings of the *name*

attribute are identical and to 0 otherwise. Now, if we computed the individual alignment between $\mathcal{A}_1$ and $\mathcal{A}_2$ by maximizing only the given lexical similarity, we would not be able to differentiate between the pairs of individuals with name *Alice* (blue squares including $a_1$, $a_3$ and $b_1$, $b_3$ in Figure 1), with name *Bob* (green squares including $a_2$, $a_4$ and $b_2$, $b_4$ in Figure 1), and with name *Chris* (red squares including $a_5$, $a_6$ and $b_5$, $b_6$ in Figure 1), respectively. Consequently, there would only be a probability of $\frac{1}{8}$ of choosing the correct alignment. In light of this, we introduce Definition 5 which extends the notion of A-Box overlap by incorporating the (a-priori) lexical confidence values as coefficients.

**Definition 5 (Weighted Overlap).** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be A-Boxes both described in terms of T-Box $\mathcal{T}$. Furthermore, let $\mathcal{M}$ be a functional one-to-one instance alignment between $\mathcal{A}_1$ and $\mathcal{A}_2$ that is valid with respect to $\mathcal{T}$ and with a-priori confidence values given by $\sigma$. The weighted overlap between $\mathcal{A}_1$ and $\mathcal{A}_2$ induced by $\mathcal{M}$ with respect to $\mathcal{T}$ is defined as*

$$overlap^{w}_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{M}) :=$$

$$\sum_{\langle a,b\rangle \in \mathcal{M}} \sum_{\substack{C \in \mathcal{T}: \\ \mathcal{T} \cup \mathcal{A}_1 \models C(a) \ \wedge \\ \mathcal{T} \cup \mathcal{A}_2 \models C(b)}} \sigma(a,b) + \sum_{\langle a,b\rangle, \langle a',b'\rangle \in \mathcal{M}} \sum_{\substack{P \in \mathcal{T}: \\ \mathcal{T} \cup \mathcal{A}_1 \models P(a,a') \ \wedge \\ \mathcal{T} \cup \mathcal{A}_2 \models P(b,b')}} \frac{\sigma(a,b) + \sigma(a',b')}{2} +$$

$$\sum_{\langle a,b\rangle \in \mathcal{M}} \sum_{\substack{C \in \mathcal{T}: \\ \mathcal{T} \cup \mathcal{A}_1 \models \neg C(a) \ \wedge \\ \mathcal{T} \cup \mathcal{A}_2 \models \neg C(b)}} \sigma(a,b) + \sum_{\langle a,b\rangle, \langle a',b'\rangle \in \mathcal{M}} \sum_{\substack{P \in \mathcal{T}: \\ \mathcal{T} \cup \mathcal{A}_1 \models \neg P(a,a') \ \wedge \\ \mathcal{T} \cup \mathcal{A}_2 \models \neg P(b,b')}} \frac{\sigma(a,b) + \sigma(a',b')}{2}.$$

The main difference between Definition 5 and Definition 3 is the weighing of the overlap of every (negated) concept and object property assertion with the a-priori similarity $\sigma$. We revisit our example to verify the ability of Definition 5 to leverage positive and negative concept and object property assertions and to improve the quality of the alignment. In order to show the improvements, we compare the weighted overlap score of the different possibilities to align the individuals named *Bob*, *Alice*, and *Chris*. With respect to the individuals named *Chris* there are two possible alignments $\{\langle a_5, b_5\rangle, \langle a_6, b_6\rangle\}$ and $\{\langle a_5, b_6\rangle, \langle a_6, b_5\rangle\}$. Both alternatives link individuals that belong to the same concept *Person* and, therefore, both add a score of two to the weighted overlap. In addition, the partial alignment $\{\langle a_5, b_5\rangle, \langle a_6, b_6\rangle\}$ links the instances having the concepts $Woman$ and $Man$ in common. Consequently, this combination adds an additional score of two to the weighted A-Box similarity. As a result, our approach will make the partial alignment $\{\langle a_5, b_5\rangle, \langle a_6, b_6\rangle\}$ part of the optimal valid one-to-one alignment due to the greater overlap of concept-assertions.

In case of the individuals named *Alice* the two possible partial alignments $\{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}$ and $\{\langle a_1, b_3\rangle, \langle a_3, b_1\rangle\}$ exist. All individuals named *Alice* belong to the concepts $Woman$ and *Person*. This means that concept assertions are not sufficient to distinguish between these alignments. However, the existing object property assertions $hasHusband(a_1, a_2)$ and $hasHusband(b_1, b_2)$ increase

the weighted similarity only for the alignment containing $\{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}$. Accordingly, our method will make the partial alignment $\{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}$ part of the optimal valid one-to-one alignment because the approach also takes object property assertions into account.

Finally, for the individuals named *Bob* the partial alignments under consideration are $\{\langle a_2, b_2\rangle, \langle a_4, b_4\rangle\}$ and $\{\langle a_2, b_4\rangle, \langle a_4, b_2\rangle\}$. Due to the disjointness axiom specified in $\mathcal{T}$ and the existing assertions one can infer the negative concept assertions $\neg Person(a_4)$ and $\neg Person(b_4)$. Hence, according to Definition 2, an alignment containing both $\langle a_2, b_4\rangle$ and $\langle a_4, b_2\rangle$ is not valid. Therefore, our method will make the partial alignment $\{\langle a_2, b_2\rangle, \langle a_4, b_4\rangle\}$ part of the optimal valid one-to-one alignment. This illustrates how our approach also factors in negative concept and object property assertions.

## 4 Optimal and Approximate Algorithms for Computing the Maximal Weighted A-Box Similarity

We now turn to the problem of devising algorithms that compute the previously defined (weighted) similarity measure between A-Boxes. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two A-Boxes both described in terms of a T-Box $\mathcal{T}$. It follows from Definition 4 and Definition 5 that, in order to compute the alignment that maximizes the weighted A-Box similarity, we have to determine

$$\underset{\mathcal{M}\in\mathbb{M}}{\operatorname{argmax}}\, overlap^{w}_{\mathcal{T}}(\mathcal{A}_1, \mathcal{A}_2, \mathcal{M})$$

with $\mathbb{M}$ the set of all functional one-to-one instance alignments that are valid with respect to $\mathcal{T}$. Notice that we can ignore the normalization denominator from Definition 4 since we are not directly interested in the maximal weighted A-Box similarity but rather the alignment that maximizes it. The problem of finding this alignment is computationally challenging due to its combinatorial complexity. It is essentially equivalent to the inexact multi-labeled graph matching problem, except that the validity requirement from Definition 2 can potentially lead to additional constraints on the set of possible alignments. As the inexact multi-labeled graph matching problem is NP-complete because it generalizes the well-known subgraph isomorphism problem [14], it can be shown that finding the alignment that maximizes the weighted A-Box similarity is also an NP-hard problem[2]. Nevertheless, we are able to provide efficient algorithms by (a) transforming the problem into an integer linear programming problem [16], and by (b) applying the approximate multi-labeled graph matching algorithm of Cour et al. [4] to the problem. We discuss the details of these two approaches in the remainder of this section.

[2] To prove the NP-hardness one can construct, for every instance of the multi-labeled graph matching problem, two A-Boxes $\mathcal{A}_1$ and $\mathcal{A}_2$ such that the alignment that maximizes the weighted A-Box similarity between $\mathcal{A}_1$ and $\mathcal{A}_2$ is also the solution to the corresponding inexact multi-labeled graph matching problem. We omit the details as the proof is beyond the scope of the paper.

### 4.1 Integer Linear Programming

Integer linear programming (ILP) can be defined as the problem of optimizing a linear objective function over a finite number of integer variables, subject to a set of linear equalities and inequalities over these variables. It is a problem that mainly occurs in the field of operations research [20]. From a mathematical perspective, it can be defined as the problem of finding a point on a polyhedron, determined by the given linear (in-)equalities, at which the linear objective function attains its minimum or maximum [16]. The problem of finding the alignment that maximizes the weighted similarity of two A-Boxes can be transformed to a integer linear programming problem as follows.

**Variables:** Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two A-Boxes described in terms of a T-Box $\mathcal{T}$ and let $a_i$, $1 \leq i \leq n$ and $b_j$, $1 \leq j \leq m$, denote the individuals in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. We will denote the set of variables of the ILP with $V$. Now, for every $1 \leq i \leq n$ and $1 \leq j \leq m$, we add the variable $x_{\langle i,j \rangle}$ to the set $V$ if there exists at least one concept[3] $C \in \mathcal{T}$ such that either $\mathcal{T} \cup \mathcal{A}_1 \models C(a_i)$ and $\mathcal{T} \cup \mathcal{A}_2 \models C(b_j)$ *or* $\mathcal{T} \cup \mathcal{A}_1 \models \neg C(a_i)$ and $\mathcal{T} \cup \mathcal{A}_2 \models \neg C(b_j)$. In addition, for every $1 \leq i,k \leq n$ and $1 \leq j,l \leq m$, we add the variables $x_{\langle i,j \rangle}$, $x_{\langle k,l \rangle}$, and $s_{\langle i,j \rangle,\langle k,l \rangle}$ to the set $V$ if there exists at least one object property $P \in \mathcal{T}$ with either $\mathcal{T} \cup \mathcal{A}_1 \models P(a_i, a_k)$ and $\mathcal{T} \cup \mathcal{A}_2 \models P(b_j, b_l)$ *or* $\mathcal{T} \cup \mathcal{A}_1 \models \neg P(a_i, a_k)$ and $\mathcal{T} \cup \mathcal{A}_2 \models \neg P(b_j, b_l)$. We will require all variables in $V$ to be binary, that is, they can take on the values 0 and 1, respectively. Note that variable $x_{\langle i,j \rangle}$ represents the correspondence $\langle a_i, b_j \rangle$, that is, $x_{\langle i,j \rangle}$ will be 1 in the solution of the ILP if and only if the correspondence $\langle a_i, b_j \rangle$ is part of the alignment that maximizes the weighted A-Box similarity. Furthermore, the variable $s_{\langle i,j \rangle,\langle k,l \rangle}$ represents the correspondences $\langle a_i, b_j \rangle$ and $\langle a_k, b_l \rangle$, that is, $s_{\langle i,j \rangle,\langle k,l \rangle}$ will be 1 in the solution of the ILP if and only if both correspondences $\langle a_i, b_j \rangle$ and $\langle a_k, b_l \rangle$ are part of the alignment that maximizes the weighted A-Box similarity.

**Objective Function:** We will now define the coefficient for each of the variables in $V$. For every $x_{\langle i,j \rangle} \in V$ we set the coefficient $c_{\langle i,j \rangle}$ to be the product of the a-priori similarity of the individuals $a_i$ and $b_j$ and the number of (negated) concepts in $\mathcal{T}$ of which both $a_i$ and $b_j$ are instances:

$$c_{\langle i,j \rangle} := \sigma(a_i, b_j) * |\{C \mid \mathcal{T} \cup \mathcal{A}_1 \models C(a_i) \wedge \mathcal{T} \cup \mathcal{A}_2 \models C(b_j)\} \cup \{C \mid \mathcal{T} \cup \mathcal{A}_1 \models \neg C(a_i) \wedge \mathcal{T} \cup \mathcal{A}_2 \models \neg C(b_j)\}|$$

Similarly, for every $s_{\langle i,j \rangle,\langle k,l \rangle} \in V$ we set the coefficient $d_{\langle i,j \rangle,\langle k,l \rangle}$ to be the product of the mean of the a-priori similarities between the individuals $a_i$, $b_j$ and $a_k$,$b_l$, respectively, and the number of (negated) object properties in $\mathcal{T}$ of which both pairs $\langle a_i, a_k \rangle$ and $\langle b_j, b_l \rangle$ are instances:

$$d_{\langle i,j \rangle,\langle k,l \rangle} := (\sigma(a_i, b_j) + \sigma(a_k, b_l))/2 * |\{P \mid \mathcal{T} \cup \mathcal{A}_1 \models P(a_i, a_k) \wedge \mathcal{T} \cup \mathcal{A}_2 \models P(b_j, b_l)\} \cup \{P \mid \mathcal{T} \cup \mathcal{A}_1 \models \neg P(a_i, a_k) \wedge \mathcal{T} \cup \mathcal{A}_2 \models \neg P(b_j, b_l)\}|$$

[3] We do *not* consider the top concept *thing* in the formulation of the ILP.

Finally, we can define the objective of the ILP as

$$\text{Maximize:} \quad \sum_{x_{\langle i,j\rangle} \in V} c_{\langle i,j\rangle} x_{\langle i,j\rangle} + \sum_{s_{\langle i,j\rangle,\langle k,l\rangle} \in V} d_{\langle i,j\rangle,\langle k,l\rangle} s_{\langle i,j\rangle,\langle k,l\rangle}$$

**Linear Constraints:** In addition to the variables and the objective function we also need to introduce several linear constraints to ensure that every feasible solution of the ILP corresponds to a valid functional one-to-one alignment between the A-Boxes $\mathcal{A}_1$ and $\mathcal{A}_2$. First, we enforce that every solution of the ILP corresponds to an alignment that is both (a) one-to-one and (b) functional by introducing the following sets of constraints:

$$\text{(a) } \forall j : \sum_{x_{\langle i,j\rangle} \in V} x_{\langle i,j\rangle} \leq 1 \text{ and (b) } \forall i : \sum_{x_{\langle i,j\rangle} \in V} x_{\langle i,j\rangle} \leq 1.$$

Furthermore, for any solution of the ILP, every variable $s_{\langle i,j\rangle,\langle k,l\rangle} \in V$ will be set to 1 if and only if the two corresponding variables $x_{\langle i,j\rangle}$ and $x_{\langle k,l\rangle}$ are also both set to 1. This can be modeled with a conjunction of the following three constraints:

$$s_{\langle i,j\rangle,\langle k,l\rangle} - x_{\langle i,j\rangle} \leq 0;\ s_{\langle i,j\rangle,\langle k,l\rangle} - x_{\langle k,l\rangle} \leq 0;\ \text{and } x_{\langle i,j\rangle} + x_{\langle k,l\rangle} - s_{\langle i,j\rangle,\langle k,l\rangle} \leq 1.$$

Finally, the validity requirement introduced in Definition 2 has to be enforced. For every variable $x_{\langle i,j\rangle} \in V$ we add the linear constraint $x_{\langle i,j\rangle} \leq 0$ if there exists at least one concept $C \in \mathcal{T}$ with $\mathcal{A}_1 \cup \mathcal{T} \models C(a_i)$ and $\mathcal{A}_2 \cup \mathcal{T} \models \neg C(b_j)$ *or* $\mathcal{A}_1 \cup \mathcal{T} \models \neg C(a_i)$ and $\mathcal{A}_2 \cup \mathcal{T} \models C(b_j)$. In addition, for every pair of variables $x_{\langle i,j\rangle} \in V$ and $x_{\langle k,l\rangle} \in V$ we add the linear constraint $x_{\langle i,j\rangle} + x_{\langle k,l\rangle} \leq 1$ to the ILP if there exists at least one object property $P \in \mathcal{T}$ with $\mathcal{A}_1 \cup \mathcal{T} \models P(a_i, a_k)$ and $\mathcal{A}_2 \cup \mathcal{T} \models \neg P(b_j, b_l)$ *or* $\mathcal{A}_1 \cup \mathcal{T} \models \neg P(a_i, a_k)$ and $\mathcal{A}_2 \cup \mathcal{T} \models P(b_j, b_l)$. Note that an additional advantage of the method is the possibility to add *known correct* correspondences to the formulation of the ILP.

The proof of the following theorem is omitted due to space constraints.

**Theorem 1.** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two A-Boxes described in terms of a T-Box $\mathcal{T}$. Furthermore, let ILP be the integer linear program constructed from $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{T}$ according to the previous steps. Then every set of variables comprising a solution of ILP correspond to an alignment that maximizes the weighted A-Box similarity between $\mathcal{A}_1$ and $\mathcal{A}_2$.*

### 4.2 Approximate Algorithm

In the experimental section, we will verify empirically that the transformation to an integer linear program can be efficiently solved for small to medium sized ontologies. However, due to the inherent computational complexity of the problem, the method will not scale to ontologies with large numbers of instances. Therefore, we will additionally apply an inexact graph matching algorithm [4] to approximate the alignment that maximizes the weighted A-Box similarity. This algorithm was originally developed for graph matching problems occurring

in the areas of computer vision and machine learning. It solves a continuous relaxation of an integer quadratic programming formulation of the inexact graph matching problem and is closely related to the spectral matching formulation of [12]. The construction of the quadratic formulation is similar to the previous construction of the ILP. We refer the interested reader to these articles for a more detailed description of the algorithm.

## 5 Experimental Evaluation

Now that we have introduced our framework for instance matching we will present empirical evidence for the utility of the method on real-world object reconciliation problems. We conducted the experiments with the following questions in mind:

- To which degree can we improve standard instance matching approaches which are mostly based on lexical similarities between datatype properties?
- How efficient is our approach with respect to runtime?
- How well does the approximate graph matching algorithm perform compared to the ILP approach?

Before we present the results of the experiments, we describe the datasets we used for our experiments as well as the baseline algorithms against which we compare our methods.

### 5.1 Dataset and Experimental Set-Up

We used the IIMB benchmark dataset[4] for the experiments. The benchmark was developed by Ferrara et al. [9] and provides a set of realistic object reconciliation problems with each of the A-Boxes containing about 300 individuals. The individuals are specific movies, actors, and directors. The T-Box is that of a typical light-weight ontology with 5 concepts, 13 datatype, and 5 object properties. There is one reference dataset with the original T-Box and A-Box and 70 different transformations which can be roughly divided into the following four categories:

**Values Transformations (VT):** Typographical errors are simulated and other lexical modifications like changing the word order are applied to datatype property values.

**Structural Transformations (ST):** The focus of these transformations is on the modification of the datatype properties themselves. They include value deletions, depth modifications, and value separations.

**Combination of VT and ST (VT & ST):** The combination of the previous two types of transformations.

**Logical Transformations (LT):** Instances are moved to different classes. These classes may be disjoint, explicit/implicit subclasses, or entirely new concepts in the T-Box.

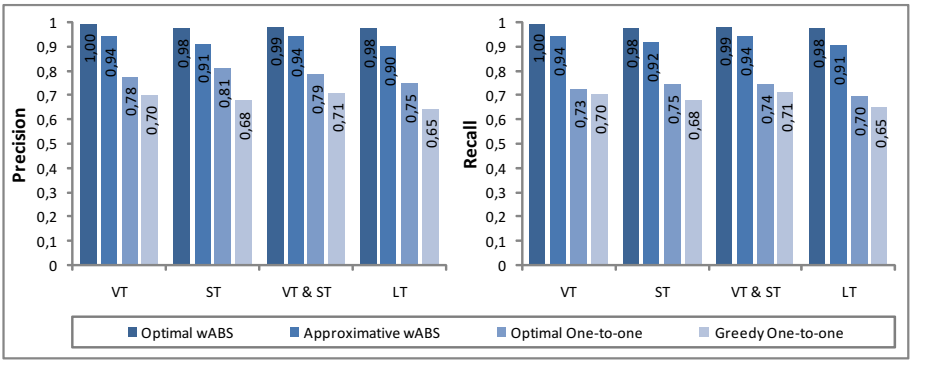



**Fig. 2.** Precision and recall for the four different methods

For our experiments we implemented a simple measure to compute the lexical a-priori similarity $\sigma$. For each pair of individuals $a$ and $b$ it considers the datatype properties $p_1, ..., p_n$ that are used to describe both $a$ and $b$. The a-priori similarity is then defined as the average similarity of these datatype property values: $\sigma(a,b) = \frac{1}{n}\sum_{i=1}^{n} sim(p_i(a), p_i(b))$. We set $\sigma(a,b)$ to zero for all pairs of individuals that have no datatype properties in common. To quantify the lexical similarity *sim* we used the SoftTFIDF string matching approach introduced by Cohen et al. [3] without modifications. Note again that the a-priori similarity $\sigma$ can be replaced by any other measure that estimates the lexical similarity of individuals. For a survey on existing methods in the context of record linkage we refer the reader to Elmagarmid [6]. Once an appropriate similarity measure $\sigma$ is chosen, most state-of-the-art approaches use one of the following two methods to generate functional one-to-one alignments. The first method selects, from the set of possible correspondences, the one correspondence $\langle a, b\rangle$ with highest confidence $\sigma(a,b)$ and removes all correspondences containing either $a$ or $b$ from the set of possible correspondences. This procedure is repeated until a functional one-to-one alignment is generated. We refer to this method as *greedy one-to-one*. The second method (denoted as *optimal one-to-one*) computes the functional one-to-one alignment that maximizes the sum of the a-priori confidence values. We implemented both methods and used the results obtained as baselines in our experiments.

We denote the optimal algorithm as *optimal wABS* (weighted A-Box Similarity) and the approximate graph matching algorithm as *approximate wABS*[5].

We used the mixed integer programming algorithm SCIP[6] to solve the ILP formulation of the *optimal wABS* algorithm. According to standard benchmarks for mixed integer linear programming algorithms[7], SCIP is one of the fastest

[4] `http://islab.dico.unimi.it/content/iimb2009/`
[5] The Matlab implementation is available at `http://www.seas.upenn.edu/~timothee/software/graph_matching/graph_matching.html`
[6] `http://scip.zib.de`
[7] `http://plato.asu.edu/ftp/milpf.html`

**Table 1.** Average execution times (in seconds) for the two different methods

| | Optimal wABS | | | Approximate wABS | | |
|---|---|---|---|---|---|---|
| | **overall** | **load and reason** | **execute algorithm** | **overall** | **load and reason** | **execute algorithm** |
| Mean | 4774.7 | 119,5 | 4728,4 | 146.2 | 121,5 | 22,5 |
| Median | 3221.5 | 123,2 | 3061,5 | 148.0 | 122,7 | 24,8 |
| St. Dev. | 4979.3 | 13,7 | 5158,1 | 19.1 | 10,6 | 5,2 |

non-commercial solvers. However, there are commercial solvers available which are several times faster than SCIP according to the benchmarks. Also, since the ILP formulation is independent of the particular solving method, progress in mixed integer linear programming will directly translate to shorter runtimes of the *optimal wABS* algorithm. The logical reasoning necessary to prepare the input for the *optimal and approximate wABS* algorithms was carried out using the reasoner Pellet [17]. All experiments were run on a desktop PC with an AMD Athlon dual core 6000+ 3.01 GHz processor and 3 GB RAM.

### 5.2 Results

We first evaluated the performance of the two baseline algorithms by comparing them to existing OAEI 2009 results of state-of-the-art matching systems. The *greedy* and *optimal one-to-one* algorithms based on the rather simple average lexical similarity achieved higher precision and recall values than 2 of the 6 state-of-the-art matchers. Hence, our baseline algorithms are comparable to the performance of existing matching algorithms. We then ran all four algorithms on the different modifications included in the IIMB dataset[8]. Figure 2 depicts the average recall and average precision values for the four categories. The results show a significant increase of precision and recall for the two wABS methods compared to the two baseline one-to-one algorithms. The *approximative wABS* algorithm has a precision and recall of 0.92 and 0.93, respectively, while the *optimal wABS* algorithm reaches a precision and recall of 0.99. Comparing this to the precision and recall of the *optimal one-to-one* algorithm of 0.78 and 0.73, respectively, we have a solid improvement between 18% and 36%. These results verify that leveraging T-Box information significantly improves the accuracy of alignments. They also show the trade-off between runtime and accuracy. The *approximative wABS* algorithm has lower precision and recall than the optimal method but is about 30 times faster. Table 1 depicts the execution times (*including* reasoning and preprocessing) of these two algorithms. While the *optimal wABS* algorithm needs an average of 1.3 hours to compute the alignment, the *approximate wABS* algorithm needs only about 2 minutes. The high standard deviation of the *optimal wABS* method speaks to the computational complexity of the problem. In most cases the ILP solver finds the optimal solution relatively

[8] We had to omit 5 of the 70 variations (19, 21, 37, 39, and 40) since these cases involved different T-Boxes.

fast but due to the hardness of the problem there are naturally some hard cases which increase the average runtime of the algorithm.

Overall, the experiments demonstrate that using the weighted A-Box similarity improves the instance alignments substantially. Since instance matching is usually not time-critical the *optimal wABS* algorithm is applicable to small to medium sized ontologies. The *approximate wABS* algorithm has the potential to also scale to larger ontologies with only a modest loss of precision and recall. The complete experimental results and the implementations are available at `http://webrum.uni-mannheim.de/math/lski/matching/rec/`

## 6 Discussion and Future Work

We proposed a framework for object reconciliation based on a semantic similarity measure between A-Boxes. The framework allows one to combine lexical a-priori similarities between instances with the terminological knowledge encoded in the ontology. We argued that most state-of-the-art approaches for instance matching focus solely on ways to compute lexical similarities. These approaches are sometimes extended by a structural validation technique where class membership is used as a matching filter. However, even though useful in some scenarios, these methods are neither based on a well defined theoretical framework nor generally applicable without adjustment. Contrary to this, our approach is grounded in a coherent theory and incorporates terminological knowledge during the matching process. Our experiments show that the resulting method is flexible enough to cope with difficult matching problems for which lexical similarity alone is not sufficient to ensure high-quality instance alignments.

Currently, our approach is restricted to generate alignments between A-Boxes described in terms of the same T-Box. In some cases this requirement is unrealistic. In such a situation it might make sense to merge the two T-Boxes prior to the instance matching process. Especially in cases where we have large A-Boxes described with relatively small T-Boxes, the benefits demonstrated by our experiments legitimate the required manual effort. In addition to this, our framework can be extended to generate both instance *and* terminological alignments at the same time. This extension requires to model instance and terminological correspondences in the same way. Instead of interpreting the axioms of the shared T-Box as hard constraints, we have to interpret both types of correspondences as soft constraints. This way we benefit from an automatically generated, uncertain terminological alignment while avoiding the risk of rejecting correct instance correspondences. In this setting, both types of correspondences are in contest with each other. The solution to the corresponding optimization problem leads to both an instance alignment and a terminological alignment. Further research will show whether this approach will provide a general framework for ontology matching that unifies instance and schema matching in an appropriate way.

**Acknowledgement.** We thank Alfino Ferrara for providing us the IIMB benchmark and for the initiative at `http://www.instancematching.org/`

## References


1. Bhattacharya, I., Getoor, L.: Entity resolution in graphs. In: Bhattacharya, I., Getoor, L. (eds.) Mining Graph Data. Wiley & Sons, Chichester (2006)
2. Borgida, A., Walsh, T.J., Hirsh, H.: Towards measuring similarity in description logics. In: Proceedings of DL (2005)
3. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of the IJCAI 2003 Workshop on Information Integration on the Web (2003)
4. Cour, T., Srinivasan, P., Shi, J.: Balanced graph matching. In: Advances in Neural Information Processing Systems 19 (2007)
5. D'Amato, C., Staab, S., Fanizzi, N.: On the influence of description logics ontologies on conceptual similarity. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 48–63. Springer, Heidelberg (2008)
6. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1 (2007)
7. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
8. Fellegi, I., Sunter, A.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
9. Ferrara, A., Lorusso, D., Montanelli, S., Varese, G.: Towards a Benchmark for Instance Matching. In: The 7th International Semantic Web Conference (2008)
10. Hassanzadeh, O., Lim, L., Kementsietsidis, A., Wang, M.: A declarative framework for semantic link discovery over relational data. In: Proceedings of the 18th International Conference on World Wide Web, pp. 1101–1102. ACM, New York (2009)
11. Horrocks, I.: Ontologies and the semantic web. CACM 51(11), 58–67 (2008)
12. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: International Conference of Computer Vision (ICCV), pp. 1482–1489 (2005)
13. Newcombe, H., Kennedy, J., Axford, S., James, A.: Automatic linkage of vital records. Science 130(3381), 954–959 (1959)
14. Papadimitriou, C.H.: Computational complexity. Addison-Wesley, Reading (1994)
15. Saïs, F., Pernelle, N., Rousset, M.-C.: Combining a logical and a numerical method for data reconciliation. Journal on Data Semantics 12, 66–94 (2009)
16. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Chichester (1998)
17. 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)
18. Stoermer, H., Rassadko, N.: Results of OKKAM feature based entity matching algorithm for instance matching contest of OAEI 2009. In: Proceedings of the ISWC 2009 Workshop on Ontology Matching (2009)
19. Stuckenschmidt, H.: A Semantic Similarity Measure for Ontology-Based Information. In: Proceedings of the 8th International Conference on Flexible Query Answering Systems (2009)
20. Taha, H.A.: Operations research: an introduction. Prentice-Hall, New York (2002)
21. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
22. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk–a link discovery framework for the web of data. In: 2nd Linked Data on the Web Workshop (2009)
23. Zhang, X., Zhong, Q., Shi, F., Li, J., Tang, J.: RiMOM results for OAEI 2009. In: Proceedings of the ISWC 2009 Workshop on Ontology Matching (2009)

# Usability of Keyword-Driven Schema-Agnostic Search

## A Comparative Study of Keyword Search, Faceted Search, Query Completion and Result Completion

Thanh Tran[1], Tobias Mathäß[2], and Peter Haase[2]

[1] Institute AIFB, KIT, Karlsruhe, Germany
ducthanh.tran@kit.edu
[2] fluid Operations, Walldorf, Germany
{tobias.mathaess,peter.haase}@fluidops.com

**Abstract.** The increasing amount of data on the Web bears potential for addressing complex information needs more effectively. Instead of keyword search and browsing along links between results, users can specify the needs in terms of complex queries and obtain precise answers right away. However, users might not always know the query language and more importantly, the schema underlying the data. Motivated by the burden facing the data Web search users in specifying complex information needs, we identify a particular class of search approaches that follow a paradigm that we refer to as *schema-agnostic*. Common to these search approaches is that no knowledge about the schema is required to specify complex information needs. We have conducted a systematic study of four popular approaches: (1) simple keyword search, (2) faceted search, (3) result completion, which is based on computing complex answers as candidate results for user provided keywords, and (4) query completion, which is based on computing structured queries as candidate interpretations of user provided keywords. We study these approaches from a process-oriented view to derive the main conceptual steps required for addressing complex information needs. Then, we perform an experimental study based on established conduct of a task-based evaluation to assess the effectiveness, efficiency and usability.

## 1 Introduction

**Motivation.** Most of the information needs are nowadays addressed by using Web search engines. Commonly, the type of search supported by these search engines is based on the *keyword search* paradigm. Using keywords, the user finds the resource of interest (informational search) or often, obtains some results as starting points (navigational search) first, from which further information is then discovered and retrieved via additional browsing. This paradigm has proven to be intuitive as well as effective in addressing simple information needs. More complex needs however require users to try out different keywords, and to browse and navigate along the complex Web space.

Recently, large amounts of structured data have been made publicly available. Two examples are data that is associated with Web pages in the form of RDFa[1] and Linked Open Data[2] (LOD), a large collection of linked datasets available in RDF that ranges

[1] A W3C Recommendation: http://w3.org/TR/xhtml-rdfa-primer/
[2] http://linkeddata.org/

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 349–364, 2010.

over many different domains. Web data like this is in the order of tens of billions RDF triples and also, it is rapidly growing. This development opens new ways for addressing complex information needs. We illustrate this with the following scenario:

*Example 1.* Mary is a novice computer science student at KIT. She is eager to learn more about this vast research field and decided to find *information about research work of researchers at AIFB*.

With traditional Web search, Mary searches and browses to find researchers at AIFB first. Then, another round of search and browsing is needed to find information about their work. Browsing is performed purely based on traversal along hyperlinks and keyword-based navigational search. The Web space is complex, containing large amounts of hyperlinks and Web pages that are relevant to a navigational query. Solving this task is thus time consuming. Using the Web of data, this complex information need can be addressed using one single query. For this, Mary specifies the need using a structured query language such as SPARQL and obtains right away the precise answers.

The example illustrates that potentially complex information needs can be addressed more efficiently using Web data. However, it also makes clear that users face the burden of specifying complex needs. Constructing structured queries requires the user to know the language and more importantly, the underlying schema. This cannot be assumed for the Web of data, which contains large amounts of evolving datasets.

**Schema-agnostic Search.** Usable query interfaces have long been an active field of research. In this work, we identify and study search approaches, which do not require users to know the schema underlying the data. Clearly, *keyword search* is a popular search interface that does not require prior knowledge about the schema. As discussed, this is a simple and intuitive paradigm that is very effective for navigational search, i.e. to obtain some starting points. From these points, users however have to do further navigation and exploration to address complex information needs. More complex needs can be expressed in terms of natural language (NL) questions. This type of interaction is supported by *NL query interfaces* [1]. Besides NL interfaces, the database community has also been investigating the use of keywords for search [2,3,4,5,6,7]. Unlike keyword search used on the Web, which focuses on simple needs, the keyword search elaborated here is used to obtain more complex results. Instead of a single set of resources, the goal is to compute complex sets of resources and their relations. We refer to this work as *result completion*, as it can be implemented to suggest completions in the form of candidate results, given the user provided keywords [8]. As an alternative to computing candidate results, candidate interpretations in the form of structured queries can be computed [9]. Given the provided keywords, completions in the form of queries are presented from which the user chooses the intended one to obtain the results in a subsequent step. We refer to this work as *query completion*. For specific domains and specific needs, experts predefine the types of complex queries that can be asked, and make them usable to lay users via customized forms. These *form-based query interfaces* have proven themselves effective for repetitive queries. They address common needs but fail on ad-hoc information needs. Another search paradigm that gains momentum is *faceted search* [10,11]. In fact, it can be seen as a special kind of form-based search. As opposed to a typical form-based interface, a faceted search interface is not restricted to the fields chosen by programmers but might contain any facets. Typically,

facets are computed on-line for the given result set. Based on these facets, the user explores and iteratively refines the result sets. Finally, less popular but also worth mentioning are graphical query interfaces [12]. The main contributions of this paper are as follows:

- Motivated by the burden facing the data Web search users in specifying complex information needs, we identify a particular class of search paradigms that we refer to as schema-agnostic approaches.
- We conduct a systematic study of four widely used approaches representing the class of keyword-driven schema-agnostic search. In particular, we focus on keyword search, query completion, result completion and faceted search, which all make use of an initial set of user keywords. We take a process-oriented view to investigate what conceptual steps these approaches require to address complex information needs.
- In the experiment, we carry out a task-based evaluation based on a large dataset and 19 participants. Based on the results, we derived the conclusions that keyword search and faceted search are effective and efficient for tasks which involve simple information needs. For complex needs, users proved to be more efficient with completion-based approaches and also, preferred them in terms of usability.

The paper is structured as follows: In Section 2 we explain the data and query model we assume for our work and define the problem of specifying queries in this context. In Section 3, we introduce a generic search process, based on which we analyze the different schema-agnostic search paradigms. We describe our experimental study in Section 4 and conclude in Section 5.

## 2 Problem Definition – Specifying Complex Information Needs

In this section we present the data and query model, in terms of which we describe the problem of specifying complex information needs.

We consider data represented using a general graph-structured data model. We distinguish two types of vertices in the graph: vertices representing entities and vertices representing data values. Edges may represent relations (connecting entities) or attributes (connecting entities with data values). We consider a predefined edge $type$, which denotes the class membership of an entity. Note that the data model is similar to that of RDF[3], but general enough to capture also web documents, XML as well as relational data.

Additionally, a data graph can be described with a schema that defines the semantics of classes and their relations. The schema itself can be represented as a graph, with classes, data types, relations and attributes as vertices. Special edges, including $subclassof$, $domain$, $range$, are used to represent class hierarchies, the domains and ranges of relations and attributes, etc.

The information needs of users we are dealing with in this work can be represented as conjunctive queries. A query of this type is basically a conjunction of query atoms,

[3] The intuitive mapping from RDF to this is, resources correspond to entities, properties to either relations or attributes and literals to data values.

which are constructed using binary predicate symbols, constants and variables. The predicate symbols are drawn from labels of relations and attributes specified in the data schema. Conjunctive queries represent a relevant fragment of first-order queries that is capable of expressing a large portion of relational queries (relational algebra). The vast majority of languages in practice falls into this fragment, including large parts of SQL, i.e. the select-from-where pattern, and SPARQL, i.e. the basic graph matching pattern. We now discuss common information needs that can be expressed with this model:

- **Entity Search.** In the IR community, this is also commonly known as navigational search. It is typically used as an entry point to the system, which is an entity such as a product, a Web page or a document in the collection. The user already knows the existence of the entity and uses the search function as a shortcut to this. Since the result is more or less known, the information need is expressed rather precisely, often with constants to refer to the particular entity, e.g. to use $name(x, AIFB)$ to search for the entity with the name $AIFB$.
- **Fact Search.** This is used in situations where the user is interested in a certain fact, like a phone number of a friend or the current temperature in San Francisco. While entity search involves one or several entities as results, this kind of search produces facts in the form of specific attribute values. Also, it is different to entity search in that it is not navigational known-item search, but rather, the purpose is to find unknown information. Thus, it is also referred to as informational search. For instance, the fact query $name(x, AIFB) \wedge location(x, y)$ searches for the specific location of $AIFB$.
- **Relation Search.** This is another type of informational search, where the goal is to gather not only information about a specific entity, but to find a complex set of entities, and especially, how they are related. In terms of conjunctive queries, this type of search allows to retrieve n-ary tuple sets as results. The running example is of the type relation search which can be expressed as $type(y, researcher) \wedge worksAt(y, x) \wedge name(x, AIFB) \wedge researchWork(y, z)$.

The examples demonstrate that different types of information needs can be represented. However, particularly for complex relation search, query construction is a difficult task. Users are required to know not only the query language but also the underlying schema, i.e. to know which predicate symbols to be used for query construction. In the context of querying Web data, this is too large a burden for the users. The data as well as the schema are evolving, making it difficult for users to keep track of the changes and to be always knowledgeable. Moreover, search on the Web often involves unknown resources. In this case, the assumption should rather be that users have no schema knowledge.

## 3 Keyword-Driven Schema-Agnostic Search

In this section, we identify and study the category of keyword-driven schema-agnostic search paradigms. As discussed, common to schema-agnostic approaches is that no knowledge about the schema is required to specify complex information needs. Work extensively studied and also applied in practice that falls into this category includes

keyword search, NL search, form-based search, faceted search, graphical query interfaces, and query as well as result completion. The last two paradigms can be regarded as extensions of auto-completion. However, auto-completion known from standard search interfaces operates at the level of words. Basically, a dictionary of terms is used to suggest candidate word completions as the user types. The completion approaches investigated here operate at a higher level, i.e. at the level of queries and results. In this paper, we focus on the study of schema-agnostic paradigms that are based on the use of keywords, i.e. keyword-driven schema-agnostic search. In particular, we will discuss keyword search, query completion, result completion as well as faceted search. We will show that, from a process-oriented point of view, also faceted search relies on the use of keywords to obtain an initial result set and thus, falls into this category.

### 3.1 Process of Specifying Complex Information Needs

We will firstly investigate the search process. The goal is to identify the main conceptual steps each approach requires to address complex information needs. In Figs. 1 and 2, we illustrate these steps for the approaches under investigation. Common to the approaches are the phases (1) specifying needs using keywords, (2) inspecting initial results and (3) further browsing, analysis and retrieval of the final results. With keyword search, there is not much help from the system to retrieve complex results. The user needs to analyze and browse at the level of resources to collect the items of interest. Faceted search supports users in inspecting, analyzing and browsing at the level of facets. With completion-based approaches, users might retrieve complex answers right away – in the form of tuples that contain resources and their relations. We will now discuss state of the art techniques and the individual steps involved in each approach.

### 3.2 Keyword Search and Resource-Based Browsing

**State of the art.** Keyword search is a paradigm commonly used by Web search engines to enable the retrieval of documents. Recently, a number of Semantic Web search engines such as Hermes [13], FalconS [14] and Sindice [15] have been developed, which also primarily rely on keyword search. Instead of documents, semantic entities (i.e. RDF resources) are returned. All these search systems are built upon the same basic concepts: (1) term-based representation of resources and queries (also called bag-of-words) and

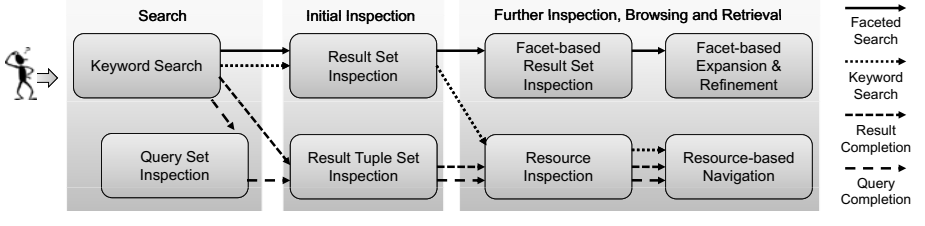



**Fig. 1.** Process of Addressing Complex Information Needs

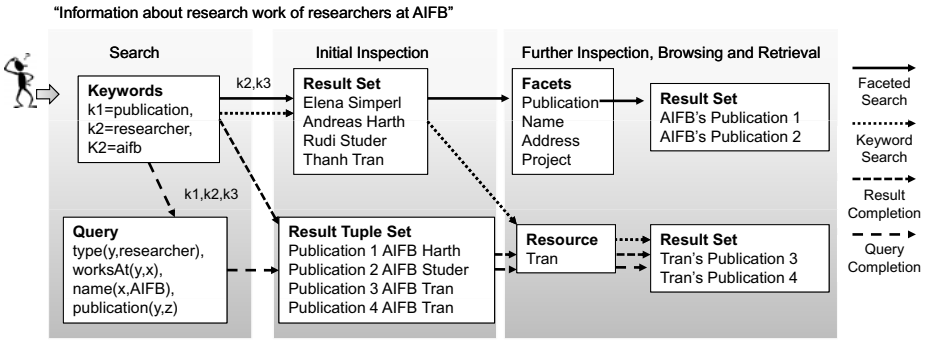



**Fig. 2.** Process for the Running Example

(2) term-based matching of queries against resources. Different matching and ranking techniques are employed. The ones frequently used are based on (an adopted version of) extensively studied information retrieval (IR) models such as the vector space model [16] and the probabilistic models [17]. The ranking schemes employed by these models leverage the discriminative quality of terms such as TFIDF and authority of resources derived via PageRank [18]. Besides, also structure information might be used for ranking. In BM25F [19] for instance, different weights are used to associate fields of documents (or generally speaking: properties of resources) with varying degrees of importance. This is to implement the intuition that certain properties are more discriminative and thus, more important than others, e.g. terms matching $name$ shall yield higher scores than terms matching $comment$. Besides the class of probabilistic models such as Okapi BM25 and its extensions, current state of the art IR approaches have embraced the use of language modelling [20]. This approach assumes that resources and expressions of information needs are objects of the same type, and assesses their match by adopting techniques of language modeling from speech and natural language processing.

In the actual implementation, the bag-of-words representation of documents are stored in an inverted index (along with scores). While commercial solutions rely on their own infrastructures, Lucene has been widely used for the implementation of Semantic Web search engines, Sindice, Hermes and FalconS in particular.

**Process.** Using a keyword search system, the user starts by entering a list of keywords. The system matches this keyword query against resources to return a ranked list of results. This might contain the immediate items of interest, which is often the case with entity and fact search. However, most keyword search performed on the Web is navigational, i.e. to obtain initial results that are then used as the starting point for further browsing. In particular for relation search (i.e. for retrieving complex information), it is necessary that the user starts with navigational search. Then, the user inspects the result set, chooses the relevant resource(s) from which further navigation is performed to collect all the items of interest. The navigation is resource-based, i.e. is performed by following links between resources.

*Example 2.* Mary enters $researchers\ AIFB$ to obtain the corresponding list of researchers. For every researcher, Mary follows the links to research work such as $publications$.

### 3.3 Keyword Search and Facet-Based Browsing/Search

**State of the art.** Browsing through the complex space of Web resources (e.g. documents, data) is a difficult task. The user might lose orientation along the way. Faceted search has become a popular paradigm as it helps to address this problem by providing facets for the user to inspect and navigate through the resources. Basically, facets correspond to properties, i.e. attributes of the underlying resources or relations between them. Faceted search is widely used in commercial systems such as on-line shops. In that setting, faceted search is conceptually similar to form-based search. The programmers predefine a fixed set of facets based on which the user can define (refine) the items of interest in terms of specific facets and facet values. Recently, advanced faceted search systems have been developed to search generic sets of resources, such as Freebase Parallax[4] for dealing with domain-independent datasets. Based on the resource schema, the system automatically computes facets for a given set of resources. Faceted search of this kind has been proposed for searching documents [21,10], for databases [22,11], as well as for RDF data [23,24].

Currently, research in this area is concerned with the aspects of efficiency and effectiveness. For instance, an efficient implementation of faceted search has been proposed based on the inverted index [25]. The Lucene index is extended to store not only terms but also the facets to which they belong. Since the number of facets that can be derived for a result set might be large, the ranking of facets has attracted interest. Widely used in faceted search systems is frequency-based ranking, which is based on the count of values that are associated with a facet [22]. Based on the similar idea, set-cover ranking has been suggested to maximize the number of distinct objects that are accessible from the top-k ranked facets [22]. A more elaborated metric is proposed by Dash et al. to incorporate the notion of interestingness [10]. Basically, it ranks those facets high which lead to surprising items, given a certain expectation. A different direction is to minimize the cost for the user finding some items of interest, where cost is derived from the facet hierarchy [11], or more specifically, the operations the user performed on it [22].

**Process.** In order to retrieve complex result sets, the user browses the set of resources obtained from keyword search. However, instead of operating at the level of resources, the user firstly obtains an overview of the resources in the form of facets. The user inspects the facets and then, navigates along these high-level resource descriptions. By adding or removing facets, the user refines and expands the current result set. This is performed until finding and collecting all relevant items of interest. Since facet search should be regarded as an additional feature, the navigation at the resource level is still possible, and might be required in some cases.

*Example 3.* After obtaining the list of resources for $researchers\ AIFB$, the user inspects the facets describing these results. This includes $name$, $address$, $affiliation$ as well as $research\ work$. The last facet can be further decomposed into $publication$ and

[4] www.freebase.com/labs/parallax/

$project$. The user navigates along these facets, and chooses to add the facet $publication$. The resulting list of resources now contains all publications from researchers at AIFB.

### 3.4 Keyword Search and Result Completion

**State of the art.** With databases, the standard use case is to obtain complex results in the form of tuples instead of documents. Also here, keyword search is recognized as an intuitive paradigm that is more assessable to lay users. The underlying technique to support retrieval of tuples using keywords leverages some concepts widely used in the IR community. In fact, IR-style ranking schemes based on TFIDF as well as an adopted version of PageRank have been applied [4,5,9]. Relations are represented using the bag-of-words model and in some systems, are also stored in an inverted index [9]. However, keywords are not matched against documents but against values (i.e. treated as terms) located in columns that are part of some database relations. More importantly, keywords of a single query might match values of a number of tuples that might be located in several relations. Thus, the matching technique employed here goes beyond simple IR-style matching. It involves additional processing to obtain sets of connected tuples, which contain all the query keywords. In particular, tuples containing some of the query keywords have to be joined such that the task amounts to finding join graphs of tuples that contain all the keywords. Each join graph is a complex tuple (that might combine results from different relations), representing the answer to the keyword query. State of the art techniques in this area can be distinguished into two classes, i.e. (1) schema-based approaches and (2) schema-agnostic approaches that directly operate at the data level. Example systems of the first type are mostly implemented as database extensions, such as DBXplorer [7] and Discover [6]. These systems translate keywords to candidate networks, which are essentially join expressions constructed using information given in the schema. These networks are then used to instantiate a number of SQL queries, which are finally executed using the underlying query engines. The results are returned as answers to the keyword query. The schema-agnostic approaches use customized indexes to match query terms against data tuples. As opposed to the schema-based approaches, the data that has to be explored for join graphs of tuples is potentially large. Thus, building the appropriate indexes to leverage materialized paths is one direction of addressing the efficiency issue [2]. Another main direction of research is to investigate efficient procedures for top-k join graph [9] (or Steiner-trees [3]) computation. Instead of returning all the results after the user finished typing the entire query, Li et al. [8] suggest to return top ranked candidate answers as completions for the keywords the user has finished typing. Thus, we refer to this kind of approaches as keyword-driven result completion.

**Process.** Instead of a list of resources, the user obtains complex results for the entered keywords in the form of tuples. In particular, the user obtains candidate results upon entering every keyword of the query. The user inspects the result tuples. When the system correctly interprets the information need as expressed in terms of keywords, the process finishes after the user has entered the entire query. The need has been satisfied right away such that no additional effort is needed for browsing and collecting results. Otherwise, the user might choose some resources in the result set, and continues with resource-based navigation and retrieval of additional results, just like with standard keyword search.

*Example 4.* Upon the user entering every keyword of the query $publication\ researchers\ AIFB$, the system suggests different candidate results, i.e. top ranked list of publications after the first keyword $publication$ has been entered, then all publications and associated researchers after the user finished adding $researchers$ to the query and finally, after the user added $AIFB$, top ranked tuples containing publications of researchers from AIFB are presented. The process stops here as the need has been satisfied.

### 3.5 Keyword Search and Query Completion

**State of the art.** Recognizing that returning queries instead of answers might improve the type of expressive keyword search discussed previously, an approach we refer to as keyword-driven query completion has been proposed [9]. The queries computed represent possible interpretations of the keywords. The idea is to present a list of candidate interpretations, from which the user can choose the intended one. The main advantages of this approach over result completion are that (1) queries can be seen as descriptions that might facilitate users in inspecting and understanding the results, (2) they enable more effective refinement (compared to operating at the level of results) and also, (3) irrelevant results can be reduced because the system computes only answers to the intended query. The technique behind this approach is similar to the one used by schema-based result completion [7,6] discussed previously. Schema information is used to search for join expressions, which can meaningfully connect elements that match the keywords. These join expressions are finally mapped to queries, representing possible interpretations of the keywords.

**Process.** Upon the user entering a keyword, the system suggests queries that represent interpretations of the keywords entered so far. The user continues typing keywords and inspecting queries, until finding a query completion that matches the intended meaning. The user poses this query against the engine to obtain complex result tuples. In cases the computed interpretations do not perfectly match the intended meaning, the user might choose to inspect specific resources and to navigate along their links, just like with standard keyword search.

*Example 5.* After entering $research\ work$, the system suggests (among others), a query that retrieves all research work. As the user finishes typing $researchers$, the system suggests a query that retrieves all research work along with the associated researchers. Finally, after the entire query $research\ work\ researchers\ AIFB$ has been finished, one query returned by the system represents the intended meaning, i.e. to retrieve all research work from researchers at AIFB. In this example, the process also stops here.

## 4 Experimental Study

In this section we present a comparison of the different search paradigms presented before using an experimental study. Because of the type of complex information needs we are dealing with (relation search in particular), the standard evaluation based on the metrics of precision and recall is too limited. Given complex needs, we aim to assess whether users are able to address them and how much effort they have to invest.

**Table 1.** Tasks Group A

| | |
|---|---|
| Task 1 | Find Tom Hanks (the actor) |
| Task 2 | Find The Beatles (the band) |
| Task 3 | Find Boston (the place) |
| Task 4 | Find the occupation of Barack Obama |
| Task 5 | Find the hometown of Metallica |
| Task 6 | Find the birthplace of Jesus |
| Task 7 | Find people with birthplace Karlsruhe |
| Task 8 | Find the names of the spouses of all directors of Rambo movies |
| Task 9 | Find all people whose birthplace is Albany, together with their deathplace |

Thus, we conduct a task-based evaluation [26] which has gained acceptance in the IR community – especially for dealing with search solutions that go beyond the standard document- and keyword-centric IR paradigm. We compare the paradigms in terms of effectiveness and efficiency, and finally, we discuss some initial usability results.

### 4.1 Description of the Experiment

**Tasks.** Every participant had to use every one of the search paradigms to solve nine tasks of varying complexity. The nine tasks involved the types of search discussed before, i.e. three of them rely on entity search, another three rely on fact search, and the remaining three involve relation search. Table 1 shows the tasks for one of the groups of the user study. Every one of the tasks was in principle solvable with either one of the search paradigms. The process of task execution was monitored by an expert. We considered a task as correctly solved as soon as the answer to the task was displayed on the screen, and the participant was able to identify this answer. For every task, the participant had the possibility to decline it as not solvable with an arguable effort. If a user spent more than three minutes trying to solve a task, we aborted this task and considered it not solved.

**Participants.** We conducted the experiment with 19 volunteers at the age of 21 to 37 years. Nine of them were software developers, two were computer science researchers and the rest of the participants were non-technical users. Fig. 3 shows statistics about the self-assessment of the participants. Most of them had experience with query languages like SQL or XQuery. The experience with standard Semantic Web Technologies like RDF or SPARQL was almost equally distributed from very much experience to no experience at all. We divided the participants in two groups (A and B) who had to solve different tasks that are of the same types.

**Systems.** We implemented the search paradigms and integrated them as separate search modules into a demonstrator system of the Information Workbench[5] that has been developed as a showcase for interaction with the Web of data. In particular, keyword search is implemented according to the design and technologies employed by standard Semantic Web search engines. Like Sindice and FalconS, we use an inverted index to

[5] http://iwb.fluidops.com/main.jsp

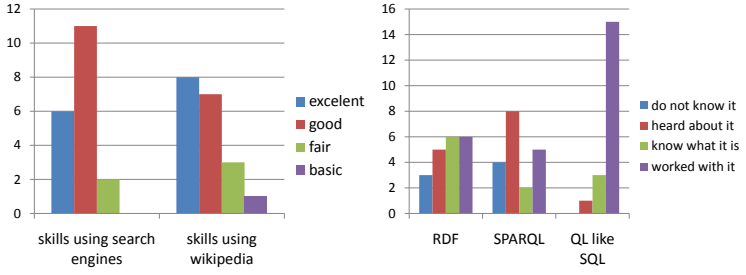



**Fig. 3.** a) Skills in using search engines and b) experience with different technologies

store and retrieve RDF resources based on terms. Also using the inverted index, faceted search is implemented based on the techniques discussed in [25]. Result completion is based on recent work discussed for the TASTIER system [8]. For computing join graphs, we use the top-k procedure elaborated in [9]. This technique is also used for computing top-k interpretations, i.e. to support query completion. We choose to display the top-6 queries and the top-25 results respectively.

**Dataset.** For the evaluation, we used DBpedia, a dataset covering a large amount of broad-ranging knowledge [27] together with YAGO [28]. It allowed us to design evaluation tasks that are not domain specific.

The experiments were conducted using commodity PCs (i.e. Intel Pentium Core2Duo with 2 x 2Ghz CPU and 3 GB RAM). All actions taken by the participants and the system responses were recorded using a screencast software. The experimental study was based on both the analysis of these screencasts as well as a questionnaire every participant had to answer after the experiment. Additional information such as this questionnaire and the handouts with a list of the tasks provided to participants can be found in the report [29].

### 4.2 Evaluation Results

In this section we firstly present the effectiveness and efficiency results. Then, we discuss usability issues based on results of the user questionnaire.

**Effectiveness.** In our context, we define the effectiveness as the fraction of the tasks which were solved correctly. The percentage of unsolved tasks grouped according to the classification introduced in Section 2 is shown in Fig. 4. We will now discuss the results with respect to the types of search used to solve the tasks.

The results show that the paradigms do not differ for simple entity search. Every participant was able to solve every entity search task with every paradigm. This is not very surprising, since entity search can mostly be performed by simply typing the name of the entity. Then, the user conducts keyword search directly or obtains a translated query and evaluates this query. Either way, it was straightforward for participants to accomplish all tasks.

For fact search, we noticed slight differences between the paradigms. About 2 percent of the fact search tasks were not solved with keyword search and faceted search.

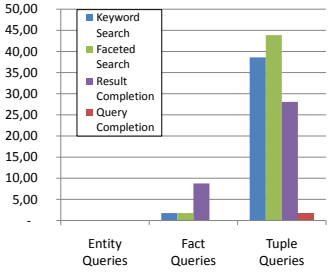



**Fig. 4.** Percentage of unsolved tasks

This means that one of the participants did not accomplish one task. We observed more failures with result completion: 9 percent of the fact queries were not solved with this paradigm. A problematic task was for instance *Find the birthplace of Jesus*. In this case, the result set is very large such that the item of interest (i.e. the entity Jesus) is not part of the top-25 results. However, more tasks were solved with query completion. More of the item of interests were covered by the top queries than the top results.

The differences in effectiveness between the paradigms became obvious for relation search: while only one candidate (about 5%) had problems obtaining the answers for relation search using query completion, we noted the percentage of tasks not solved using result completion to be 25 percent. The tasks that were difficult are for instance *Find people whose birthplace is Washington, together with their death place*. Since this involves a very special information need it was necessary to specify very precise keywords. Otherwise, the number of results computed as candidate answers is high – and in this case, does not contain the relevant answer. We found that with relation search, the percentage of unsolved talks is even higher for keyword search. For instance, only one candidate was able to accomplish searching for people born in Washington and their death place. To do this, users have to find the town (i.e. Washington) first. This is easily done by submitting the keyword *Washington*. Then, users have to inspect the information contained in the resource page. In particular, they have to find all the people born in this city. This is also relatively straightforward because in this case, this information was simply given in a list. However, much effort is required in the final step, where users have to visit the pages of every person in order to find out their death place. Along the way, most participants recognized this problem and did not complete or refused to complete the task because the given time of three minutes is too limited.

**Efficiency.** For assessing the efficiency, we identified all basic user interactions with the systems. These interactions correspond to the steps as discussed previously for the search process. We consider the operations of entering a keyword, executing the keyword query (time needed for the keyword search step in keyword search and faceted search), completing the keyword query (time needed for the keyword search step in query completion and result completion), visiting a resource (resource inspection and

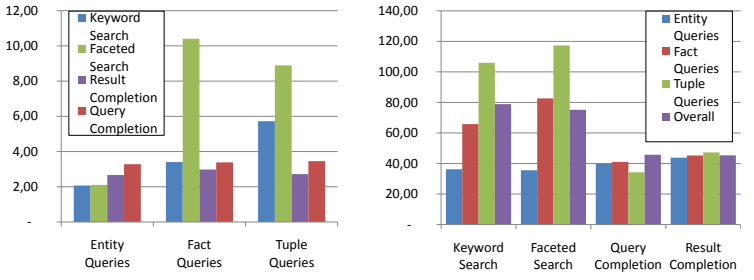



**Fig. 5.** a) Number of basic operations, b) average time users needed to solve a task (in seconds)

resource-based navigation), inspecting complex results (query set inspection, result tuple set inspection), facet-based result inspection and the operations for facet-based expansion and refinement. We measured the average times to conduct every one of these interactions. We took only the times for correctly solved tasks into consideration.

The efficiency results for the different paradigms are shown in Fig. 5. The results suggest the time needed for keyword search and faceted search increases rapidly with the complexity of the information needs. This is clearly illustrated in Fig. 5(b). While keyword search and faceted search were the most efficient paradigms for entity queries, the time almost doubled for fact queries. This trend continues as we turn to the even more complex conjunctive queries. We observed another large increase of 40 to 60 percent. The increase in time invested by the user is in fact larger for faceted search than for keyword search. Fig. 5(a) delivers an explanation for this: The number of basic operations performed by participants using faceted search is by far the highest. Although each can be performed efficiently, the overall high number of operations lead to a large amount of time in total.

In contrast to these two paradigms, we did not observe a significant increase of time for the completion-based paradigms. In fact, the time invested by participants is almost the same for all types of information needs, as illustrated in Fig. 5(b). Fig. 5(a) also shows that the number of basic operations is independent of the complexity of the information needs. The search process consists of the same steps for every query: enter a list of keywords, translate those keywords, and, when using query completion, choose the correct query, and finally, find the answer in the result set. In the cases users did not choose the keywords precisely enough, they had to repeat the entire process.

Under the aspects of efficiency, we observed only minimal differences between result and query completion. Result completion seems to be faster in some cases.

**User Questionnaire.** After the experiment, we conducted an interview with all of the candidates to assess the usability of the search paradigms. Fig. 6(a) shows that 12 out of 19 candidates (65%) liked the query completion most. The preferences of the remaining candidates were equally distributed among the other three paradigms. Further details on the comparison of the two paradigms with best evaluation results so far systems are shown in Fig. 6(b+c). It seems that most users did not face any difficulties with choosing the correct results or the correct queries while using the completion-based paradigms. Query completion seems to be slightly more usable than result completion.

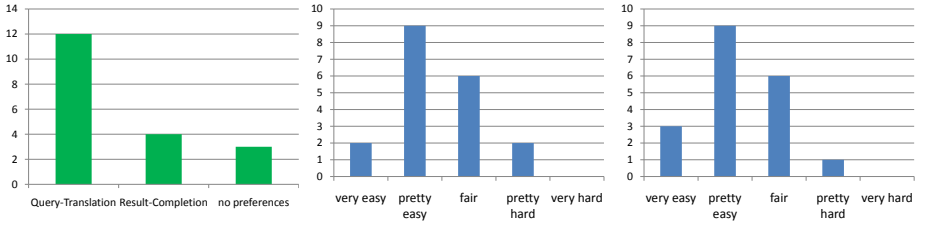



**Fig. 6.** a) most popular paradigm, b) ease of choosing correct result (result completion), c) ease of choosing correct query (query completion)

## 5 Conclusions

The increasing availability of structured data on the Web bears potential for addressing complex information needs more effectively. A primary challenge lies in enabling the users to specify complex queries without the user knowing details of the internal data and query model, and more importantly, the underlying schema of the data. In this paper, we identified and analyzed schema-agnostic search paradigms which address this challenge. We have conducted a systematic study of four popular keyword-driven approaches that rely on the use of keyword queries: (1) simple keyword search, (2) faceted search, (3) result completion, and (4) query completion. We have studied these approaches from a process-oriented view and then preformed a controlled user study to compare them.

From our experimental study we can draw the following conclusions: Not surprisingly, simple keyword search turned out to be sufficient for simple information needs. In fact, the results show that the effectiveness of the paradigms does not differ for simple entity search. The advantages of more advanced search paradigms were apparent for more complex information needs. For the most complex information needs under consideration (relation search), it turned out that only with query completion the users were effectively able to answer most queries.

Directions for future work thus include the question of how the paradigms can be combined in a useful way such that the user can easily resort to the most adequate paradigm for a given information need. In this context, the personalization of search interfaces is relevant, as preferences for one or the other paradigm might be subjective.

**Acknowledgements.** Research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen project (grant 01IA08005K).

## References

1. Androutsopoulos, L.: Natural language interfaces to databases - an introduction. Journal of Natural Language Engineering 1, 29–81 (1995)
2. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: SIGMOD Conference, pp. 305–316 (2007)

3. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB, pp. 505–516 (2005)
4. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient ir-style keyword search over relational databases. In: VLDB, pp. 850–861 (2003)
5. Liu, F., Yu, C.T., Meng, W., Chowdhury, A.: Effective keyword search in relational databases. In: SIGMOD Conference, pp. 563–574 (2006)
6. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword search in relational databases. In: VLDB, pp. 670–681 (2002)
7. Agrawal, S., Chaudhuri, S., Das, G.: Dbxplorer: enabling keyword search over relational databases. In: SIGMOD Conference, p. 627 (2002)
8. Li, G., Ji, S., Li, C., Feng, J.: Efficient type-ahead search on relational data: a tastier approach. In: SIGMOD Conference, pp. 695–706 (2009)
9. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: ICDE, pp. 405–416 (2009)
10. Dash, D., Rao, J., Megiddo, N., Ailamaki, A., Lohman, G.M.: Dynamic faceted search for discovery-driven analysis. In: CIKM, pp. 3–12 (2008)
11. Roy, S.B., Wang, H., Das, G., Nambiar, U., Mohania, M.K.: Minimum-effort driven dynamic faceted search in structured databases. In: CIKM, pp. 13–22 (2008)
12. Fogg, D.: Lessons from a “living in a database" graphical query interface. In: SIGMOD Conference, pp. 100–106 (1984)
13. Tran, T., Wang, H., Haase, P.: Hermes: Data web search on a pay-as-you-go integration infrastructure. J. Web Sem. 7(3), 189–203 (2009)
14. Cheng, G., Qu, Y.: Searching linked objects with falcons: Approach, implementation and evaluation. Int. J. Semantic Web Inf. Syst. 5(3), 49–70 (2009)
15. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
16. Wong, S.K.M., Ziarko, W., Wong, P.C.N.: Generalized vector space model in information retrieval. In: SIGIR, pp. 18–25 (1985)
17. Robertson, S.E., Maron, M.E., Cooper, W.S.: The unified probabilistic model for ir. In: SIGIR, pp. 108–117 (1982)
18. Richardson, M., Domingos, P.: The intelligent surfer: Probabilistic combination of link and content information in pagerank. In: NIPS, pp. 1441–1448 (2001)
19. Robertson, S.E., Zaragoza, H., Taylor, M.J.: Simple bm25 extension to multiple weighted fields. In: CIKM, pp. 42–49 (2004)
20. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR, pp. 275–281 (1998)
21. Hearst, M., Swearingen, K., Li, K., Yee, K.P.: Faceted metadata for image search and browsing. In: CHI 2003: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 401–408. ACM, New York (2003)
22. Dakka, W., Ipeirotis, P.G., Wood, K.R.: Automatic construction of multifaceted browsing interfaces. In: CIKM 2005: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 768–775. ACM, New York (2005)
23. Schraefel, W.M., Russell, A., Smith, D.A.: mspace: improving information access to multimedia domains with multimodal exploratory search. Commun. ACM 49(4), 47–49 (2006)
24. Hyvönen, E., Mäkelä, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula, S.: Museumfinland – finnish museums on the semantic web. Journal of Web Semantics 3(2), 25 (2005)
25. Ben-Yitzhak, O., Golbandi, N., Har’El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E.J., Sznajder, B., Yogev, S.: Beyond basic faceted search. In: WSDM, pp. 33–44 (2008)

26. Elsweiler, D., Ruthven, I.: Towards task-based personal information management evaluations. In: SIGIR, pp. 23–30 (2007)
27. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. J. Web Sem. 7(3), 154–165 (2009)
28. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A large ontology from wikipedia and wordnet. J. Web Sem. 6(3), 203–217 (2008)
29. Mathaess, T.: Semantische suchsysteme, Master's thesis, Karlsruhe Institute of Technology (2009), http://www.stud.uni-karlsruhe.de/ usauk/studium/diplomathesis/damathaess.pdf

# Collaborative Semantic Points of Interests

Max Braun, Ansgar Scherp, and Steffen Staab

University of Koblenz-Landau, Germany
{maxbraun,scherp,staab}@uni-koblenz.de
http://isweb.uni-koblenz.de

**Abstract.** The novel mobile application csxPOI (short for: collaborative, semantic, and context-aware points-of-interest) enables its users to collaboratively create, share, and modify semantic points of interest (POI). Semantic POIs describe geographic places with explicit semantic properties of a collaboratively created ontology. As the ontology includes multiple subclassifications and instantiations and as it links to DBpedia, the richness of annotation goes far beyond mere textual annotations such as tags. Users can search for POIs through the subclass hierarchy of the collaboratively created ontology. For example, a POI annotated as bakery can be found through the search string *shop* as it is a superclass of bakery. Data mining techniques are employed to cluster and thus improve the quality of the collaboratively created POIs.

## 1 Introduction

Mobile devices with permanent Internet connectivity are turning the vision of ubiquitous computing into reality [1]. Applications running on such devices are among others aware of the user's geographic location and can adapt their behavior and content accordingly. Points of interests (POIs) provide useful information about specific geographic places. Whether users are able to quickly find the POIs they are interested in, depends among others on the quality of the POIs' annotations. Unstructured textual descriptions of POIs and folksonomic tags are a good starting point. However, the relation between the POIs remains hidden in the data and is hard to extract and understand for the machine. Other (mobile) systems such as [2,3] provide a semantic representation of POIs but do not provide for a collaborative creation and modification of semantic POIs and an underlying ontology of POI categories.

In order to understand semantic POIs and their relations, we have developed the mobile application csxPOI (short for *collaborative, semantic, and context-aware points of interests*) to collaboratively create, share, and modify *semantic POIs*. While working with the POIs, the users collaboratively modify and improve an ontology of POI categories underlying to the application. Such user-contributed POIs gathered from a large group of people is likely to include duplicate POIs with similar but unequal annotations and slightly varying locations for the same physical places. Thus, a revision engine applies data mining techniques for POI clustering.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 365–369, 2010.

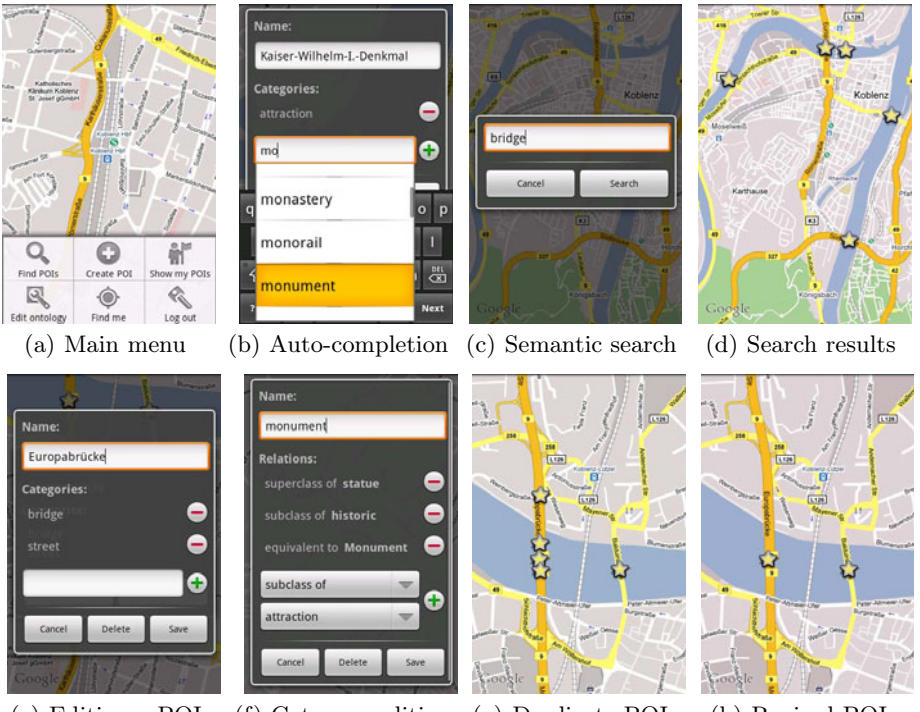
## 2 Collaborative Ontology for Points of Interests

We have created a set of ontologies for modeling the structure of POIs, content of POIs, user accounts, and most notably the history of all collaborative user activities. The *base ontology* defines the fundamental concepts such as POI, contribution, user, and their relationships. The *vocabulary ontology* defines POI categories such as monument, park, and others. It also provides the properties, relationships, and interlinks of the POIs. The vocabulary ontology is open to direct collaborative modification by users of the csxPOI application. The *POI ontology* comprises all instances of POIs and their associated metadata. As such, the ontology directly depends on the categories defined in the vocabulary ontology. Users of the csxPOI application primarily interact with this ontology during POI creation and POI retrieval. The *collaboration ontology* represents any kind of collective activity in the csxPOI application and are called *contributions*. A contribution is either the creation, modification, or deletion of a POI as well as the creation, modification, and deletion of a POI category in the vocabulary ontology. The collaboration ontology directly depends on the previous three ontologies. Finally, the *user ontology* contains all instances of users of the csxPOI application and their respective account information. As the first users of our csxPOI application have not yet created their own POI categories, there is an initial vocabulary ontology extracted automatically from LinkedGeoData [4].

## 3 The csxPOI Application

The default screen of the csxPOI application is shown in Figure 1(a). Once registered, users can *Create POIs* at the position centered on the map. Subsequently, a dialog containing a text field for the POI's name is shown. In addition, semantic categories of the POI can be entered. Figure 1(b) shows a sample POI created for the *Monument to Kaiser Wilhelm I* in Koblenz, Germany. The screenshot shows the POI with its German name *Kaiser-Wilhelm-I.-Denkmal* and a list of the two categories named *attraction* and *monument*. Categories are added by entering their name in the corresponding text field which features a semantic auto-completion function. A sample usage of the semantic auto-completion can be seen in Figure 1(b), where the categories beginning with *mo* are shown. Each entry corresponds to a category from the collaborative POI ontology introduced in Section 2 with all its relations and interlinks. The user can either select one of the suggested categories and annotate the POI by pressing the plus button or entering a new name, i.e., a category that is not included in the ontology. Entering a new category opens the ontology editor for further specification of it.

Users can *Find POIs* in the vicinity and display them on the map. Here, two options are possible: semantic search and presenting all POIs associated with the current user. A sample result for the semantic search *bridge* can be seen in Figure 1(d), where the POIs are marked with star symbols. The feature *Show my POIs* depicted in Figure 1(a) provides a list of all POIs a user created or modified. The details of a POI displayed on the map can be shown by tapping




(a) Main menu (b) Auto-completion (c) Semantic search (d) Search results

(e) Editing a POI (f) Category editing (g) Duplicate POIs (h) Revised POIs

**Fig. 1.** Screenshots of our csxPOI application showing its different features

on it. It shows the name at the top and below a list of all categories the POI belongs to. For example, a POI named *Europabrücke* (in english: Europe bridge) with the two categories *bridge* and *street* is shown in Figure 1(e). The semantic annotation allows to search along the hierarchy of concepts a POI belongs to. For example, a POI that is annotated with the concept bakery can be found through the search term *shop* as the concept shop is a superclass of bakery.

The user can *Edit* the POI, when logged in. Figure 1(e) shows the POI editor for the previous example. The POI name can be edited in a text field at the top. Categories are added by entering their name in the text field at the bottom with support of semantic auto-completion and pressing the plus button. This may open the ontology editor if a category name is unknown. Removing categories is conducted via the minus button.

Besides creating and editing semantic POIs, the collaborative POI ontology can also be edited directly through the *ontology editor*. An example is shown in Figure 1(f) for the category *monument*. Users can edit the category's name in the text field. Below is a list of relations, each representing one triple with the edited category as the subject of the triple. The predicate is defined by the relation type. It can be one of *subclass of*, *superclass of*, and *equivalent to*. The former two correspond to the property rdfs:subClassOf and its inverse, the latter to owl:sameAs. The object of the triple is the target category. It is shown on the right hand side and can be any category from the vocabulary ontology.

In addition, it can be a concept from an external resource such as DBpedia (http://dbpedia.org/). Relations are added and removed by choosing the relation type and category from the lower drop-down menus as shown in Figure 1(h) and pressing the plus button and minus button, respectively.

## 4 Revision of Collaboratively Created POIs

The csxPOI application provides a revision engine that clusters duplicate POIs with combinations of spatial, linguistic, and semantic similarity measures. The revision engine follows a two-step approach: In a first step, the spatial similarity is measured by calculating the distance between two POIs and mapping it to inverse values in $[0, 1]$ with 1 indicating the identical location and 0 indicating maximal distance. The distance is based on the WGS 84 reference ellipsoid modeling the earth's surface [5]. Linguistic similarity is modeled by the Jaro-Winkler [6] string metric between the labels of two POIs. In the csxPOI ontology, POIs are instances of categories which themselves have subclass relations among each other. This taxonomic structure is utilized to assess a semantic similarity between POIs by comparing their relative position in the hierarchy. This is computed with the asymmetric similarity measure MDSM [7]. The different metrics are integrated with a weighted linear combination of their individual normalized values. In the second step, clusters of POIs are determined employing the DBSCAN algorithm [8]. We choose the medoid as representative of the cluster. An owl:sameAs relation pointing to the medoid is then added to the members of the cluster. Figure 1(g) shows a screenshot where five semantic POIs have been created. The four POIs on the left are all variations of the actual bridge and vary in location and the name given, which is *Europabrücke* (two times), *Europa-Brücke*, and *Europabrucke*. These four POIs are categorized as bridges as they instantiate the category bridge. But some additionally belong to the street category and others to pedestrian. The bridge can be used by both vehicles and pedestrians, which lead to this ambivalent categorization. The POI on the right has the name *Balduinbrücke* and also represents a bridge. After clustering, the four POIs on the left have been merged into one POI as shown in Figure 1(h).

## 5 Related Work

IYOUIT is one of today's most sophisticated mobile context-aware applications [2]. It features social relationships, location records, and weather conditions, which are stored in formal ontologies. DBpedia [3] is an effort to extract structured information from Wikipedia and publish it in Linked Data. DBpedia Mobile [9] is a mobile client to explore that data. It uses the client's GPS sensor and semantic datasets to display nearby places on a map. DBpedia Mobile's Linked Data browser provides the user with background information about discovered places. In addition, there are several other applications that allow for mobile creation and sharing of POIs such as Wikitude (http://www.wikitude.org/). Discussing all of those applications is beyond the scope of this paper. A nice application for mobile creation and sharing of

multimedia documents is Haiku (http://haikus.tid.es) where users can leave comments and annotate the provided content. Unlike our approach, none of the existing approaches allow for the semantic annotation and collaborative creation of a network of semantic POIs and their respective categories.

## 6 Conclusions

We presented a novel mobile application csxPOI for collaboratively creating, sharing, and modifying semantic points of interest. Additionally, the vocabulary ontology comprising the POI categories used for the semantic annotation of POIs is itself a product of a collaborative process. As user-contributed POIs inherently introduce duplicate POIs, we provide a POI revision engine based on data mining techniques to improve the quality of the collaborative POI data set. To fill the cold start up situation, we investigate among others detecting events on the social media sharing platform Flickr and presenting images of those events. Further information and demo videos can be found at: http://west.uni-koblenz.de/Research/systeme/csxPOI

*Acknowledgment.* This research has been co-funded by the EU in FP7 in the WeKnowIt project (215453).

## References

1. Weiser, M.: The Computer for the 21st Century. Scientific American 265(3) (1991)
2. Boehm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, M., Wibbels, M.: Introducing IYOUIT. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 804–817. Springer, Heidelberg (2008)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: 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. Auer, S., Lehmann, J., Hellmann, S.: Linkedgeodata: Adding a spatial dimension to the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009)
5. National Imagery and Mapping Agency (NIMA): Department of Defence World Geodetic System 1984. NIMA Technical Report 8350.2 (2004)
6. Winkler, W.E.: The State of Record Linkage and Current Research Problems. R99/04, Statistics of Income Division, U.S. Census Bureau (1999)
7. Rodríguez, M.A., Egenhofer, M.J.: Comparing Geospatial Entity Classes: An Asymmetric and Context-Dependent Similarity Measure. Int. Journal of Geographical Information Science 18 (2004)
8. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: 2nd Int. Conf. on Knowledge Discovery and Data Mining (1996)
9. Becker, C., Bizer, C.: DBpedia Mobile: A Location-Enabled Linked Data Browser. In: 1st Int. Workshop on Linked Data on the Web (2008)

# Publishing Math Lecture Notes as Linked Data

Catalin David[1], Michael Kohlhase[1], Christoph Lange[1], Florian Rabe[1], Nikita Zhiltsov[2], and Vyacheslav Zholudev[1]

[1] Computer Science, Jacobs University Bremen, Germany
{c.david,m.kohlhase,ch.lange,f.rabe,v.zholudev}@jacobs-university.de
[2] Mathematics, Kazan State University, Russia
nikita.zhiltsov@gmail.com

**Abstract.** We mark up a corpus of LaTeX lecture notes semantically and expose them as Linked Data in XHTML+MathML+RDFa. Our application makes the resulting documents interactively browsable for students. Our ontology helps to answer queries from students and lecturers, and paves the path towards an integration of our corpus with external sites.

## 1 Application: Computer Science Lecture Notes

Over the last seven years, the second author has accumulated a large corpus of teaching materials, comprising more than 2,000 slides, about 1,000 homework problems, and hundreds of pages of course notes, all written in LaTeX. The material covers a general first-year introduction to computer science, graduate lectures on logics, and research talks on mathematical knowledge management. This situation is typical for educators and researchers and represents the state of the art in mathematics, physics, computer science, and engineering: LaTeX has proven suitable for writing high-quality lecture notes and publishing them as PDF. However, in our educational setting, we would like to benefit from the much larger degree of interactivity that screen reading and e-books support. For example, while reading notes students want to directly look up the meaning of a symbol (e. g. $\models$) in a formula, or examples for a difficult concept (e. g. structural induction). They may want to select advanced material for self-study from the whole body of lecture notes, based on the topics covered in the lecture. They want to use a search engine to find related material in other universities' online course notes, on mathematical web sites, or Wikipedia. Lecturers want to query their repository for document parts reusable in an upcoming lecture, given the prerequisites students are expected to meet and the material that has already been covered. In a course for a special audience, e. g. mathematics for physicists, they want to draw examples from that domain even though they are less familiar with it. They also want to locate didactic gaps, such as concepts without examples, or unjustified proof steps. These services require semantic annotations in the lecture notes that are understandable for external search engines. Plain LaTeX is barely usable for anything *beyond* on-screen reading and printing. Even simple semantic annotations are uncommon, rare exceptions are the `\title` command making its meaning explicit or `\frac{a}{b}` focusing on functional

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 370–375, 2010.

structure instead of visual layout. This is especially problematic for symbols in formulæ, which are often overloaded with multiple definitions or presentable in different notations. $\binom{n}{k}$ can be a vector or a binomial coefficient, and a French or Russian would write the latter as $\mathcal{C}_n^k$. Therefore, we have developed a semantic representation of mathematical knowledge in LaTeX and a presentation process that preserves these semantic structures as Linked Data in the output, exposing them to mashups for interactive exploration, as well as semantic searching and querying. These are based on an ontology for mathematical knowledge so that mathematical content can be linked across different repositories.

## 2 Research Background and Related Work

LaTeX's importance in scientific authoring and its extensibility by macros have led to semantic extensions enabling modern publishing workflows. SALT (semantically annotated LaTeX [8]) marks up rhetorical structures and fine-grained citations in scientific documents. Its markup is not sufficiently fine-grained for formulæ, and its vocabulary is limited to rhetorics and citations and not extensible. Our own sTeX offers macros for introducing new mathematical symbols and using arbitrary metadata vocabularies. Some math e-learning systems, such as ActiveMath [1] or MathDox [17], use semantic representations of formulæ and higher-level structures, e. g. proof steps or course module dependencies, in the standard XML languages OpenMath [5] and OMDoc [11]. They utilize semantic structures but do not *publish* them in a standard representation like RDF, which would promote general-purpose queries beyond the built-in services and integration with other systems on the web. The Linking Open Data movement promotes best practices for publishing data on the web [9], as standalone RDF or embedded into HTML documents as RDFa [2]. Applications include Sindice, an engine that crawls and indexes Linked Data [19], and the Sparks $O_3$ Browser, a mashup that utilizes RDFa annotations in HTML for interactive browsing [20]. Our interactive documents work similarly but additionally support annotations in MathML formulæ. MathML has pioneered embedded annotations long before RDFa, albeit with a more limited scope. Its *parallel markup* interlinks the rendered appearance and the semantic structure of mathematical expressions; the meaning of mathematical symbols is usually defined in lightweight ontologies called OpenMath content dictionaries [4]. HELM (Hypertext Electronic Library of Mathematics [3]) pioneered representing structures of mathematical knowledge in RDF, e. g. what mathematical theory introduces a symbol, what of its properties have been declared or asserted, and how the latter are proved. The HELM ontology has not gained wide acceptance, though. At the time of its development, there was no RDFa-like way of embedding RDF into web documents.

## 3 Architecture and Demo

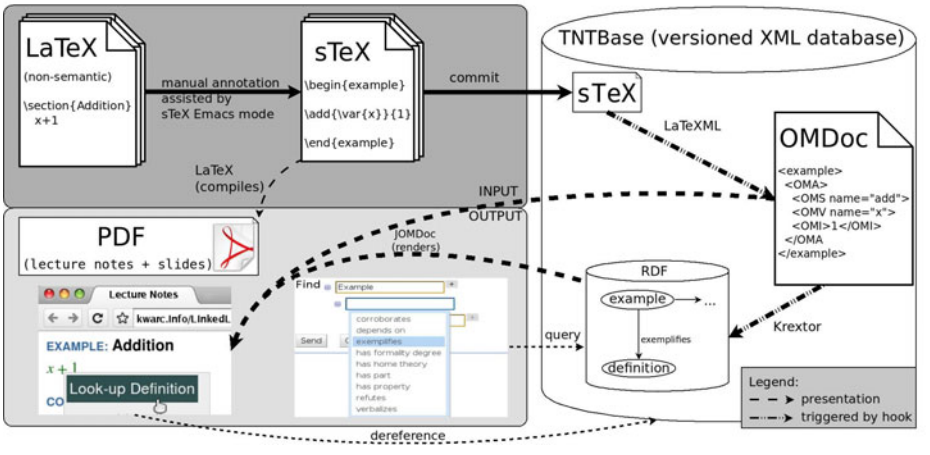
Our architecture publishes semantically enriched LaTeX lecture notes as XHTML+MathML+RDFa Linked Data. We kept LaTeX as an input language, as

it is familiar to authors and well supported by editors, and as high-quality PDF can be obtained from it. With sTEX (semantically enhanced TEX), we have introduced LATEX macros for marking up the semantic structure of formulæ and documents [12] and manually annotated our complete corpus using the sTEX plugin for the Emacs editor. One can, e. g., declare a symbol *union*, formally define it, and make its semantic representation `\union{A,B,C}` expand to `A\cup B\cup C` for human-readable rendering. There are environments for mathematical statements and theories, e. g. `\begin{example}[for=union]`. LATEXML transforms this into a semantically equivalent intermediate XML representation, using the standard XML languages OpenMath for formulæ [5] and OMDoc for higher-level structures [11]. Finally, our JOMDoc rendering library [10] generates human-readable output from this XML – an output that still contains the full semantic structure as annotations. A custom Java implementation renders formulæ as parallel markup of Presentation MathML annotated with OpenMath[1]; rendering higher-level structures as XHTML+RDFa [2] is implemented in XSLT. RDF is extracted from XML by our Krextor XML→RDF library [15], which generates URIs for all mathematical objects in a document. It uses our OMDoc ontology (cf. [14]) as a vocabulary for representing all mathematical structures (e. g. "$d$ is a definition, $e$ is an example for $d$") plus full text, inspired by HELM and designed as a more expressive counterpart of the OMDoc XML schema.

The whole transformation process is integrated into our versioned XML database TNTBase [22]; see `http://kwarc.info/LinkedLectures`. TNTBase has a Subversion-compatible interface making it suitable as a lecture notes repository. The TEX→XML and XML→RDF transformations are automatically triggered by a hook upon committing a new revision of an sTEX lecture module. If the generated OMDoc+OpenMath is not schema-valid, the commit is rejected. On the other hand, it follows Linked Data best practices and, depending on the MIME type an HTTP client requests, serves a document as OMDoc, as RDF (only a structural outline, not the full text and formulæ), or as XHTML+MathML+RDFa. The latter contains JavaScript code from our JOBAD library for interactive documents [13,7], which operationalizes the annotations – Linked Data and other – in the rendered documents. JOBAD's definition lookup determines the OpenMath annotation of the Presentation MathML symbol the user clicked on, from that obtains the URI of the symbol, and then requests XHTML from that URI (resulting in the symbol's declaration and definition), which is then displayed in a popup. The RDFa annotations are used for making parts of a document (e. g. steps of a structured proof) foldable, and for displaying the local neighborhood in the RDF graph (e. g. related examples) in popups; this is implemented using the rdfQuery library [18], relying on the Linked Data structure in the latter case. Further third-party services can be integrated in a mashup style; we have demonstrated this for a unit conversion service [13,7]. Besides

[1] A proposal for fully representing formulæ in RDF [16] has not gained wide acceptance. RDF-based reasoners are often limited to decidable first order logic subsets, which is insufficient for mathematical applications, and XML has a straightforward notion of order (e. g. of the arguments of an operator or of a set constructor).

enabling JOBAD's services, we have implemented machinery to load the extracted RDF into a triple store and query it using SPARQL. We also provide a widget for formulating queries without knowing SPARQL and the OMDoc ontology. It allows to ask some non-trivial queries, e. g. "find examples for all concepts from graph theory (about which I'm planning a lecture), assuming as prerequisites the concepts from formal languages (and their prerequisites)". This would yield the parse tree of a context-free language as an example for the concept "tree" – as operating systems were not among the prerequisites.




Our demo shows the complete pipeline in action: (i) annotating a document with our sTEX Emacs mode, (ii) committing it to TNTBase, (iii) automatic translation to OMDoc, schema validation, and RDF extraction,(iv) loading the extracted RDF data into a triple store, (v) retrieving the document in different representations, (vi) browsing the XHTML+MathML+RDFa rendering, (vii) interacting with the Linked Data in it, (viii) and querying a triple store. Additionally, we will demonstrate the generation of PDF from the sTEX sources.

## 4 Conclusion and Outlook

Our architecture makes legacy LATEX lecture notes available as Linked Data. We expose these data to external clients but have also implemented services for interactively exploring the XHTML+MathML+RDFa presentation of our data. We are also working on preserving some of the semantics in the PDF output, as SALT does. Evaluation of our enriched lecture notes by the student end users is planned for the next semester. To the best of our knowledge, we are the first provider of RDF-based Linked Data in the domain of mathematics and among the first to operationalize the Linked Data structures of formula markup. Having successfully transformed more than 300,000 normal, non-semantic LATEX documents from `arxiv.org` to XHTML+Presentation MathML [21] and working on machinery for automatically annotating them using natural language processing, we will soon be able to expose even more mathematical knowledge as Linked

Open Data; however, due to the inherent complexity of mathematical knowledge, with a less formal semantics than manually annotated documents. Our lecture notes are self-contained so far, but we are now starting to reap the benefits of Linked Data by linking them to other data sets, e. g. DBpedia [6], whose mathematical knowledge does not have a semantics as strong as ours, but which provides abundant informal background knowledge, e. g. about the originators of mathematical theories. On the other hand, hardly any well-known mathematical site (e. g. planetmath.org and mathworld.wolfram.com) currently exposes machine-understandable metadata. We promote our technology, starting with lightweight RDFa annotation using the OMDoc ontology, as a migration path towards their integration into a true mathematical Semantic Web.

## References


1. ActiveMath, http://www.activemath.org
2. RDFa in XHTML: Syntax and processing. Recommendation, W3C (2008)
3. Asperti, A., Padovani, L., Sacerdoti Coen, C., Guidi, F., Schena, I.: Mathematical knowledge management in HELM. Annals of Mathematics and Artificial Intelligence, Special Issue on Mathematical Knowledge Management 38(1-3), 27–46 (2003)
4. MathML 3.0. Candidate Recommendation, W3C (2009)
5. The Open Math standard, version 2.0. Technical report, Open Math Society (2004)
6. DBpedia, http://www.dbpedia.org
7. 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.) Calculemus 2009. LNCS (LNAI), vol. 5625, pp. 279–293. Springer, Heidelberg (2009)
8. Groza, T., Handschuh, S., Möller, K., Decker, S.: SALT – semantically annotated LaTeX for scientific publications. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 518–532. Springer, Heidelberg (2007)
9. Linked data guides, http://linkeddata.org/guides-and-tutorials
10. JOMDoc — Java library for OMDoc documents, http://jomdoc.omdoc.org
11. Kohlhase, M.: OMDoc – An open markup format for mathematical documents [Version 1.2]. LNCS (LNAI), vol. 4180. Springer, Heidelberg (2006)
12. Kohlhase, M.: Using LaTeX as a semantic markup format. Mathematics in Computer Science 2(2), 279–304 (2008)
13. Kohlhase, M., Giceva, J., Lange, C., Zholudev, V.: JOBAD – interactive mathematical documents. In: AI Mashup Challenge (2009)
14. 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)
15. Lange, C.: Krextor – an extensible XML→RDF extraction framework. In: Scripting and Development for the Semantic Web, SFSW 2009 (2009)
16. Marchiori, M.: The mathematical semantic web. In: Asperti, A., Buchberger, B., Davenport, J.H. (eds.) MKM 2003. LNCS, vol. 2594, pp. 216–223. Springer, Heidelberg (2003)
17. MathDox – interactive mathematics, http://www.mathdox.org

18. rdfQuery – RDF processing in browser, http://code.google.com/p/rdfquery/
19. Sindice – the semantic web index, http://sindice.com
20. Sparks $O_3$ browser: Enlighten the web, http://oak.dcs.shef.ac.uk/sparks/
21. Stamerjohanns, H., Kohlhase, M., Ginev, D., David, C., Miller, B.: Transforming large collections of scientific publications to XML. Mathematics in Computer Science (2010)
22. Zholudev, V., Kohlhase, M., Rabe, F.: A (insert xml format) database for (insert cool application). In: Proceedings of XML Prague (2010)

# GoNTogle: A Tool for Semantic Annotation and Search

Giorgos Giannopoulos[1,2], Nikos Bikakis[1,2], Theodore Dalamagas[2], and Timos Sellis[1,2]

[1] KDBSL Lab, School of ECE, Nat. Tech. Univ. of Athens, Greece
giann@dblab.ntua.gr, bikakis@dblab.ntua.gr
[2] Institute for the Management of Information Systems, "Athena" Research Center, Greece
dalamag@imis.athena-innovation.gr,
timos@imis.athena-innovation.gr

**Abstract.** This paper presents GoNTogle, a tool which provides advanced document annotation and search facilities. GoNTogle allows users to annotate several document formats, using ontology concepts. It also produces automatic annotation suggestions based on textual similarity and previous document annotations. Finally, GoNTogle combines keyword and semantic-based search, offering advanced ontology query facilities.

**Keywords:** Document Annotation, Ontology, Semantic Search.

## 1 Introduction

Semantic annotation and search tools are at the core of Semantic Web Technology. Annotations involve tagging of data with concepts (i.e., ontology classes) so that data becomes meaningful. Annotating data can help in providing better search facilities, since it helps users to search for information not only based on the traditional keyword-based search, but also using well-defined general concepts that describe the domain of their information need.

A great number of approaches on semantic annotation have been proposed in the literature [2]. Most of them are focused on annotating web resources such as html pages or plain text [3,4,5,6]. As far as popular document formats are concerned, there are approaches that differ in the annotation and search facilities they offer [7,8].

In this paper we present GoNTogle. GoNTogle supports manual and automatic annotation of several types of documents (doc, pdf, rtf, txt, odt, sxw) using ontology classes, in a fully collaborative environment. It also provides searching facilities beyond the traditional keyword-based search, using a flexible combination of keyword and semantic-based search. In contrast with other works, our aim was to implement an easy-to-use document annotation and search tool, that would fully support (a) viewing and annotating popular document types while maintaining their initial format, (b) sharing those annotations and (c) searching for documents combining keyword and semantic-based search.

The key features of our tool are the following:

- It allows users to open and view widely used document formats such as *.doc* and *.pdf*, maintaining their original format.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 376–380, 2010.

- It provides an easy and intuitive way of annotating documents (or document parts) using OWL and RDF/S ontologies.
- It provides an automatic annotation mechanism based on models trained from user annotation history, so that annotation suggestions are tailored to user behavior.
- It is based on a server-based architecture, where document annotations are stored in a central repository. Thus, we offer a collaborative environment where users can annotate and search documents.
- It combines keyword and semantic search, providing advanced search facilities for both types of search.

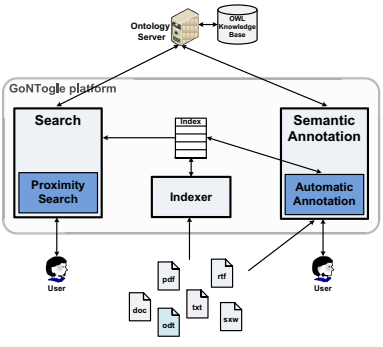



**Fig. 1.** GoNTogle architecture

## 2 System Overview

GoNTogle's architecture is presented in Figure 1. The system is divided into 4 basic components: a) *Semantic Annotation Component*, that provides facilities regarding the semantic annotation of documents. It consists of 3 modules: a Document Viewer, an Ontology Viewer and an Annotation Editor. b) *Ontology Server Component*, that stores the semantic annotations of documents in the form of OWL ontology instances. c) *Indexing Component*, that is responsible for indexing the documents using an inverted index. d) *Search Component*, that allows users to search for documents using both textual (keyword search) and semantic (ontology search) information.

### 2.1 Semantic Annotation

Semantic Annotation Component offers 2 primary functionalities: (a) annotation of whole document and (b) annotation of parts of a document. Also, a user may select between manual and automatic annotation.

Figure 2 depicts the Semantic Annotation window of our application. The user may open a document in the Document Viewer, maintaining its original format. In the particular example, the user has opened a *.doc* document. Moreover, she can load and view the hierarchy of an ontology through the Ontology Viewer.

The user can, then, select one or more ontology classes and manually annotate the whole document or part of it. The annotation is saved as an ontology instance in the

Ontology Server, along with information about the annotated document (or part). On the same time, an annotation instance is added in the Annotation Editor list. Each record of this list corresponds to an annotation stored in the Ontology Server. For example (Figure 2), the abstract of the document is annotated with class *H.2.3_Languages-Query_Languages*, while the whole document is annotated with class *H.2_DATABASE_MANAGEMENT*. The user can edit those annotation instances, adding or removing ontology classes, or completely remove them.

The user may also choose the automatic annotation functionality. In this case, the system suggests candidate ontology classes, executing our learning method based on *weighted kNN* classification [1], that exploits user annotation history to automatically suggest annotations.

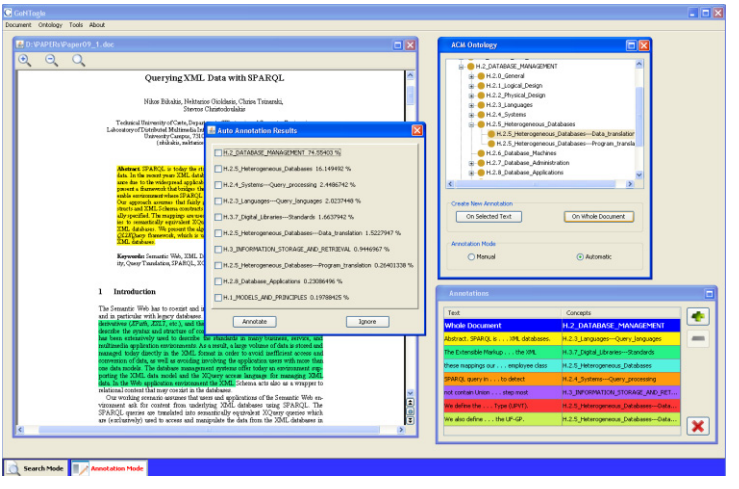



**Fig. 2.** Semantic annotation example

## 2.2 Search

Search component provides a series of search facilities which are described below. We start with the simple ones:

- *Keyword-based search*. This is the traditional searching model. The user provides keywords and the system retrieves relevant documents based on textual similarity.
- *Semantic-based search*. The user navigates through the classes of the ontology and selects one or more of them. The result list from this type of search consists of all documents that have (partially or on their whole) been annotated with one or more of the chosen classes.
- *Hybrid search*. The user may search for documents using keywords and ontology classes. She can, also, determine whether the results of her search will be the intersection or the union of the two searches.

Next we present a set of advanced searching facilities that can be used after an initial search has been completed.

- *Find related documents*. Starting from a result document *d*, the user may search for all documents which have been annotated with a class that also annotates *d*.
- *Find similar documents*. This is a variation of the previous search facility. Starting from a result document *d*, the user may search for all documents which are already in the result list and have been annotated with a class that also annotates *d*.
- *Get Next Generation*. The resulting list from an semantic-based search can be expanded by propagating the search on lower levels in the ontology (i.e., if class *c* has been used, then search is propagated only in direct subclasses of *c*). This is the case when the search topic is too general.
- *Get Previous Generation*. This offers the inverse functionality of the previous option. The resulting list from an semantic-based search can be expanded by propagating the search on higher levels in the ontology (i.e., if class *c* has been used, then search is propagated only in direct superclasses of *c*). This is the case when a search topic is too narrow.
- *Proximity Search*. This search option allows the user to search for documents not only belonging to a selected class but also belonging to the classes's subclasses. That is, if class *c* has been used, then search is propagated in all direct subclasses of *c* and their direct subclasses. The resulting documents gathered from those three levels of the ontology hierarchy are weighted properly (documents from the initial class *c* get higher score).

The resulting documents are presented in a ranked list. From the information presented in this list the user can find out whether a resulting document comes from keyword/ontology/hybrid search, and which classes it is annotated with (if any). Also, each result is accompanied by a score. This score is a weighted sum of (a) the textual similarity score given by keyword search and (b) a score that represents the extend to which each document is annotated with the classes selected in ontology search.

## 3 Implementation and Evaluation

In what follows we provide technical information about the implementation of our system and we present a preliminary experiment we performed.

**Implementation.** To compose our system, we utilized several open source tools and libraries. For indexing and keyword searching we used Lucene search engine library[1].

We used Protégé[2] as Ontology Server, so that document annotations are stored as OWL class instances.

OpenOffice API[3] was essential in incorporating in our system a viewer that could maintain the exact format of *.doc* documents, which is a very common filetype. The same applies for Multivalent[4], a generalized document viewer that was integrated in our system so that *.pdf* files could also maintain their format when being viewed and annotated.

[1] http://lucene.apache.org/java/docs/
[2] http://protege.stanford.edu/
[3] http://api.openoffice.org/
[4] http://multivalent.sourceforge.net/

All annotation and search facilities presented in this paper have been implemented in a Java prototype. Detailed information and application screenshots, as well as the application itself and installation instructions can be found in http://web.imis.athena-innovation.gr/~dalamag/gontogle.

**Evaluation.** We have performed a preliminary evaluation of the semantic annotation tool. More specifically, we tested the precision and recall of the automatic annotation process.

We turned the ACM Computing Classification into an OWL ontology. The ontology produced is a 4-level structure with 1463 nodes.

We used 500 papers, pre-categorized according to the ACM System, as sample documents for the ontology classes. Then, we used the tool to automatically annotate 66 papers from the publication database maintained in our lab. Figure 4 presents precision and recall diagrams for simple and weighted *kNN*. Best values were observed for *k=7*.

We observed better results when using weighted *kNN* compaired to simple *kNN*, so we adopted the former in our final system implementation.

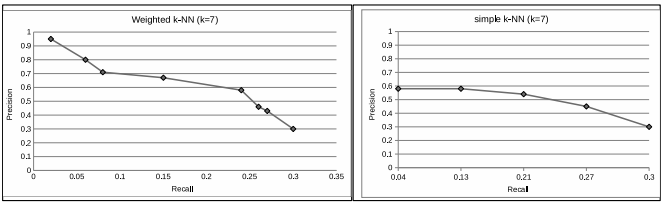



**Fig. 3.** Evaluation results for automatic annotation based on *kNN*

## References

1. Mitchell, T.M.: Machine Learning. WCB/McGraw-Hill (1997)
2. Uren, V.S., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Journal of Web Semantics 4 (2006)
3. Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., et al.: MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, p. 379. Springer, Heidelberg (2002)
4. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proc. of the ACL 2002 (2002)
5. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. Journal of Web Semantics 2(1) (2004)
6. Chakravarthy, A., Lanfranchi, V., Ciravegna, F.: Cross-media document annotation and enrichment. In: 1st Semantic Authoring and Annotation Workshop 2006 (2006)
7. Eriksson, H.: An annotation tool for semantic documents. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 759–768. Springer, Heidelberg (2007)
8. Tallis, M.: SemanticWord processing for content authors. In: Proc. of the Knowledge Markup and Semantic Annotation Workshop 2003 (2003)

# A Software Tool for Visualizing, Managing and Eliciting SWRL Rules

Saeed Hassanpour, Martin J. O'Connor, and Amar K. Das

Stanford Center for Biomedical Informatics Research
MSOB X215, 251 Campus Drive, Stanford, California, USA 94305
{saeedhp,martin.oconnor,amar.das}@stanford.edu

**Abstract.** SWRL rule are increasingly being used to represent knowledge on the Semantic Web. As these SWRL rule bases grows larger, managing the resulting complexity can become a challenge. Developers and end-users need rule management tools to tackle this complexity. We developed a rule management tool called Axiomé that aims to address this challenge. Axiomé support the paraphrasing of SWRL into simple English, the visualization of the structure both of individual rules and of rule bases, and supports the categorization of rules based on an analysis of their syntactic structure. It also supports the automatic generation of rule acquisition templates to facilitate rule elicitation. Axiomé is available as a plugin to the Protégé-OWL ontology development environment.

**Keywords:** Rule Management, Rule Elicitation, Rule Visualization, Rule Paraphrasing, Rule Categorization, OWL, SWRL.

## 1 Introduction

SWRL rules [4] are increasingly being used to represent knowledge on the Semantic Web. As the size of rule bases increases, users can face problems in understanding the content of these rule bases and managing the complexity of resulting knowledge. To support this rule management task, and to assist in the rapid exploration and understanding of rule bases, we have developed a tool that assists in the interpretations of rule bases through the generation of high-level abstractions of rule base structure. In particular, this software uses rule paraphrasing and rule visualization to help non-specialists understand and manage potentially complex rule bases. It also supports the automatic detection of common patterns in rule bases through the categorization of rules into related groups. These categorizations are used as a basis of rule elicitation functionality that assists non-specialists in developing new rules.

This tool is called Axiomé [3] and was developed as a plug-in to the popular Protégé-OWL ontology development environment. It provides an array of tools for managing SWRL rule bases in OWL ontologies. It supports visual rule base exploration, automated rule categorization, rule paraphrasing, and rule elicitation functionality with the goal of facilitating the management of large SWRL rule bases.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 381–385, 2010.

## 2 Axiomé: Core Features

Axiomé has five main functional areas, which are available as sub-tabs within the plug-in: (1) a Rule Graph tab that provides a graph structure to browse and explore the SWRL rule base; (2) a Rule Visualization tab to visualize individual rules; (3) a Rule Paraphrasing tab that displays an English-like text explanation for each rule; (4) a Rule Categorization tab to automatically categorize rules in a rule base; and (5) a Rule Elicitation tab that provides a graphical templates to acquire new rules based on analysis of existing rules in a rule base. A Rule Browser component is permanently displayed to show a tree-table representation of the SWRL rules in an ontology. This tree-table enables users to explore the rule base and launch any of five sub-tabs for the rule or group being explored.

### 2.1 Rule Graph

The Rule Graph tab provides a graphical representation of a SWRL rule base (Figure 1). Each rule in the rule base is shown as a node; edges between two nodes indicate the SWRL atoms shared by the rules and the dependency direction. A number of graphical layouts are supported. The JUNG visualization framework [2] was used to generate graph layouts. Search functionality is also provided. Rules matching a search term can be visually highlighted in the graph. Rule groups and dependencies between rules can also be indicated visually. Different types of dependency groups can be filtered and indicated separately in the dependency graph. Also, cyclical dependencies between rules can be found and visually highlighted.

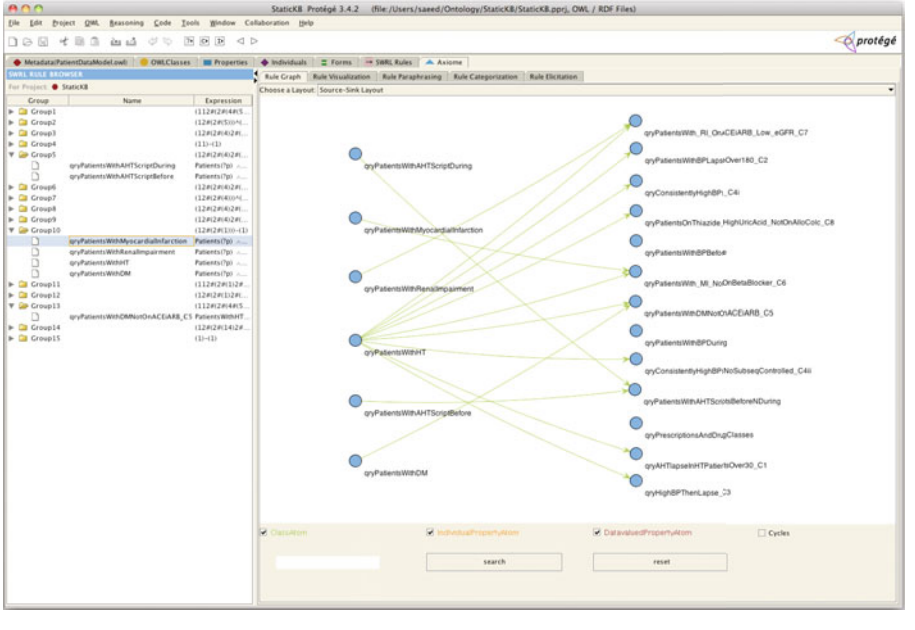



**Fig. 1.** An example of a rule base visualization in Axiome's Rule Graph tab. This visualization shows the dependencies between rules in a clinical care auditing rule base.

## 2.2 Rule Visualization

The Rule Visualization tab (Figure 2) allows the arrangement of atoms in a rule to be visualized as a tree structure. These trees are generated by performing a depth-first search for each variable chain in the rule. A number of heuristics are employed to ensure that the most significant clauses in a rule are given more prominence [1].

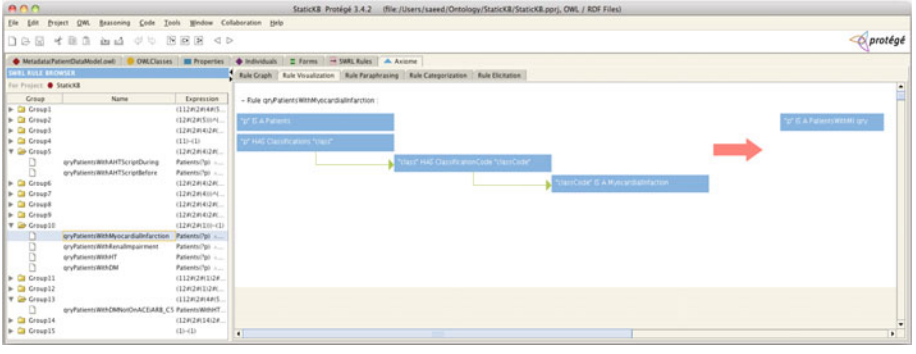


**Fig. 2.** An example of the visualization of a simple rule in the Rule Visualization tab

## 2.3 Rule Paraphrasing

The Rule Paraphrasing tab (Figure 3) uses a similar approach to build a tree structure for each rule and then uses additional heuristics to generate understandable English paraphrases of that rule [1].

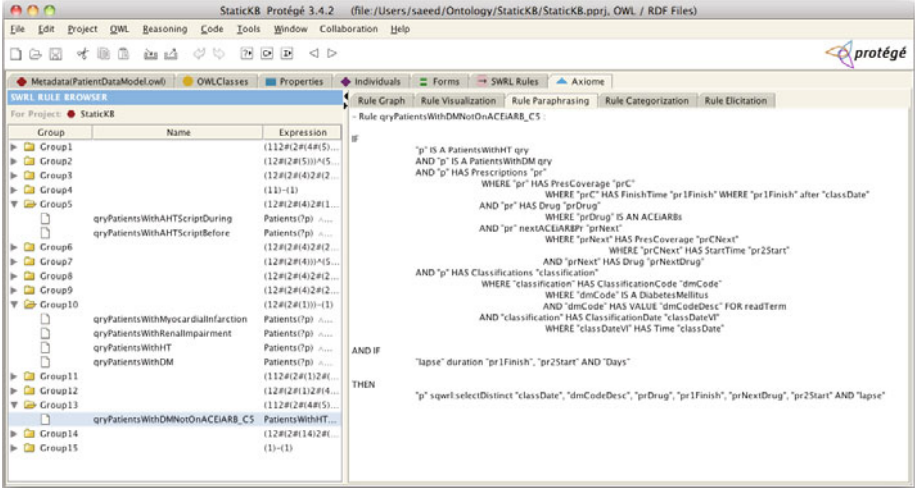



**Fig. 3.** An example of rule paraphrase generated by Axiome's Rue Paraphrasing tab. The rule is selected from a clinical care auditing rule base.

### 2.4 Rule Categorization

The Rule Categorization tab uses the data structure that is generated for the rule visualization and paraphrasing tabs to automatically group rules with a similar syntactic structure [1]. It then graphically displays the results of this grouping. These groupings can then be used in the Rule Graph tab when exploring the rule base.

### 2.5 Rule Elicitation

The Rule Elicitation tab (Figure 4) provides graphical rule templates to facilitate acquisition of rules. It generates these templates using the syntactic structures generated for rule grouping. Users can select an appropriate rule group and then generate a graphical acquisition template to enter rules with the structure of other rules in that group.

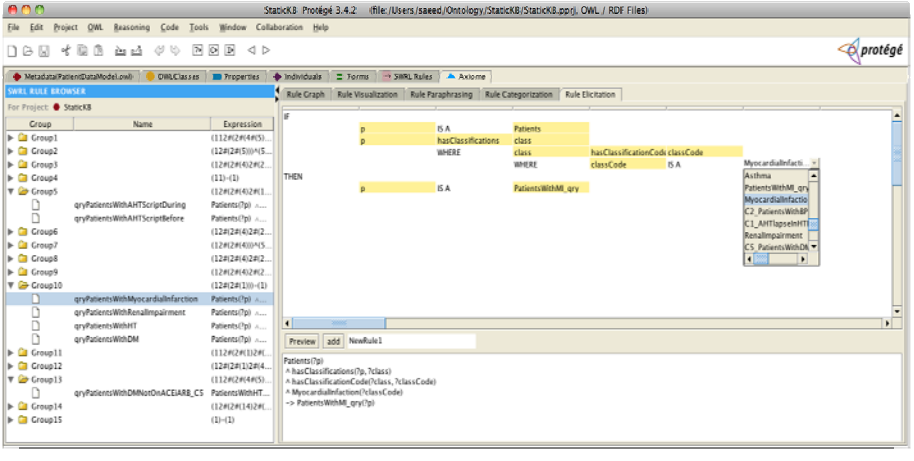



**Fig. 4.** Screen shot of Axiome's Rule Elicitation tab. This tab can automatically generate a rule acquisition template from an existing rule group.

## 3 Summary

We have described Axiomé, a free, open-source Protégé-OWL plug-in to support SWRL rule management. We have shown how the various features of the tool can facilitate understanding of SWRL rule bases through rule base graphing, rule visualization, and rule paraphrasing. The tool also supports the extension of rule bases through the elicitation of new rules using rule templates. It is included as one the default plug-ins in the latest Protégé-OWL 3.4 release [3]. We are currently testing its utility in the development and management of biomedical rule bases. We are also implementing a web-based version, which will to be available as a plugin to Web Protégé.

## References

1. Hassanpour, S., O'Connor, M.J., Das, A.K.: Exploration of SWRL Rule Bases through Visualization, Paraphrasing, and Categorization of Rules. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 246–261. Springer, Heidelberg (2009)
2. Java Universal Network/Graph Framework, http://jung.sourceforge.net
3. Axiomé, http://protegewiki.stanford.edu/index.php/Axiomé
4. SWRL Submission, http://www.w3.org/Submission/SWRL
5. Clinical-care auditing ontology, http://www.cs.auckland.ac.nz/~thusitha/aiim09/

# A Knowledge Infrastructure for the Dutch Immigration Office

Ronald Heller, Freek van Teeseling, and Menno Gülpers

Be Informed, Linie 620, 7325 DZ Apeldoorn, Netherlands
{r.heller,f.vanteeseling,m.gulpers}@beinformed.nl

**Abstract.** The Dutch Immigration Office is replacing its existing paper based case system with a fully electronic system with integrated decision support based on ontologies. The new award winning architecture (Dutch Architecture award 2009) is based on the principle of separation of concerns: data, knowledge and process. The architecture of the application, but especially the architecture of the knowledge models for decision and process support, is explained and shown in the demonstration.

**Keywords:** Knowledge models, paper based, case system, decision and process support, Be Informed, Run – Model – Analyze principle, knowledge as a service, separate know from flow.

## 1 Background and Application Context

The Dutch Immigration Office [4] decided to completely replace its existing (mostly) paper based case system and some form of decision support (decision trees). The new application had to be a completely electronic case system with integrated decision and process support, including a directly connected front office [5]. The case system is mainly targeted at the internal knowledge worker who handles the incoming requests. The front office is for clients (foreigners) who apply for a permit to stay in the Netherlands. This application replaces the current operational cluster of systems, that do not provide the flexibility and agility that is needed today. Management costs are way too high and changes take far too long to be implemented.

Ordina and Accenture together with Be Informed came up with the award winning architecture (Dutch Architecture Award 2009) [6] that is all about separation of concerns: data, knowledge and process. This enables and ensures the flexibility and agility that is so needed. When these concerns are not interconnected, the changes can be implemented, reviewed and validated in isolation, which shortens time needed for implementations as a whole, and thus enable greater agility. The semantic based knowledge support consists of a large set of models which serves both the front office (website [5]) and the back office (using SOA [2]). This implies that the same models are used for different audiences. The website offers functionality to decide what application to fill in, including tailor-made online forms, and the back office provides a range of services to support the case management system with e.g. process flows, decision support and generation of documents. The webportal is fully operational,

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 386–390, 2010.

whereas the back office components are currently being deployed in the organization. A similar architecture has been used in several other recent projects.

## 2 Technology

Be Informed is a software product to support complex and knowledge-intensive business processes. Using Be Informed software, organizations improve their service to customers and partners, streamline their working processes and achieve substantial gains in efficiency by delivering the appropriate knowledge in a direct and context-specific manner to business users and customers. Be Informed gives organizations the ability to quickly adapt to changes in legislation or their surroundings.

Be Informed is an integrated platform for all required services, processes and tasks. This enables business partners to implement knowledge infrastructures in (large) enterprises, that can be used to build multi-product, multi-label and multi-lingual knowledge intensive applications, to address the challenges of a modern organization.

Be informed uses its own strongly typed syntax, mainly because the existing web standards [1] like OWL [7] and RDF(S) [8] have at least 2 major drawbacks in a deployed application like the one subject of this demonstration. First, there is a strong need to link textual and visual content to the models, to be displayed in certain parts of the application. Secondly, decisions and classifications can be modeled much more easily using a "Closed World" assumption rather than the usual "Open World" assumption.

In Be Informed Modeling, all models are based on a metamodel: the knowledge architecture[9]. It defines conceptual areas in the domain(s) and specifies their dependencies. It also defines the context of knowledge in surrounding IT systems, the organization's generic process and case systems, which are also part of Be informed.

Based on ontology design patterns [10], we divided the semantic models for the Dutch Immigration Office in 4 main areas:

- Core Taxonomies. Basic vocabulary of the organization (permits, actions, purpose of stay, criteria etc) which is referenced by many knowledge models.
- Regulations. For each set of regulations a semantic model is built to represent the applicable laws and regulations, and act as decision support in the operational process.
- Online front office. This is the portal for the customer (foreigner) to explore the information, get personalized advice and apply for a permit.
- Catalog. This is a documentation model serving as an explanatory model "about the other models" to help developers, modelers and business users to find their way in modeling.

Note that there is a loosely coupled 5$^{th}$ area, that of a cross-governmental portal, leading foreigners coming to the Netherlands to all relevant governmental organizations, including the IND: "New to Holland" [3].

In the models we use strongly typed relations (instance, subclass, requires, depends on, etc.) connecting different concept types. In the metamodel concept types are defined for Control data, Actions, Conditions for actions, States, Roles and Generic Properties.

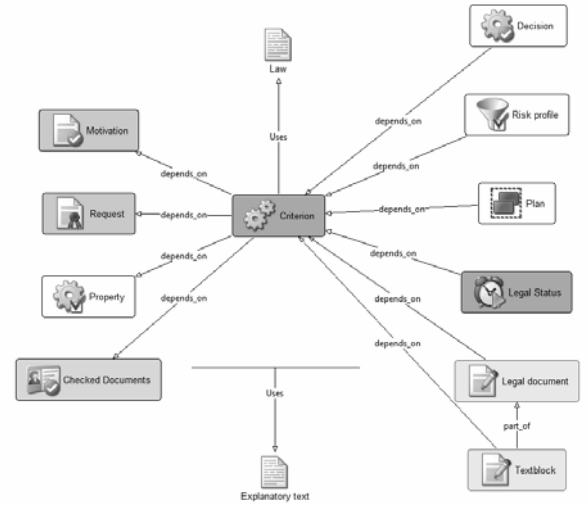



**Fig. 1.** Excerpt of the Metamodel of the IND Knowledge Base

The different models not only use the same metamodel, but are also built in roughly the same way. We use conditions or norms (criteria) that tell whether a specific request is permitted or not. To establish this decision one or more characteristics of the foreigner (properties) are compared with legal grounds.

### 2.1 Overall Architecture

The architecture of the case systems with decision support is all about the separation of the knowledge from the process (separate the know from the flow). The case system is responsible for the data on files and persons, whereas the decision support system is responsible for the decisions taken based on (the correct) laws and legislation. The two components interact via web services with XML SOAP messages through an Enterprise Service Bus which only handles the transfer of messages.

The Case system uses web services to interact with the decision support system (knowledge as a service) for complex questions like: “Which activities need to be performed”, “What is the risk concerning this applicant”, “What decision should be made (automatically or supporting)”, ”Which publications should be made” (determine and assemble), calculate fees, etc.

The case system follows the generic reference process shown in figure 2, that states that all processes of the Dutch Immigration Office are on some basic level the same. They all handle incoming requests, decide on the actions to be taken, process these actions and publish the results. Although the process is the same, the handling of a request never is. Each request has its own specific data, but the handling differs because of the knowledge applied. The relevant law and legislation relies on the data in the request, which for the Dutch Immigration Office is basically the purpose of stay.

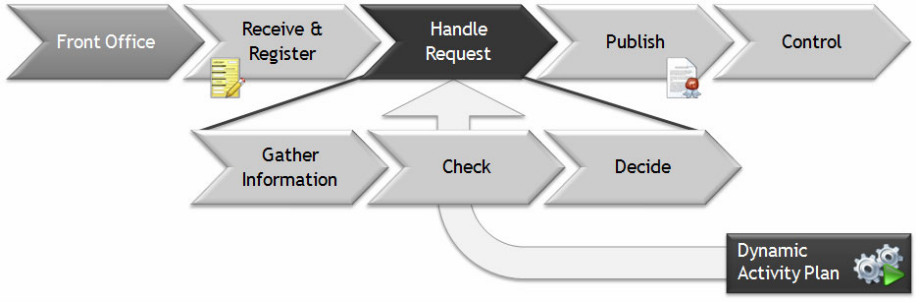



**Fig. 2.** Generic Reference Process to be filled by knowledge

The front office is built with SharePoint as the portal software and the decision support systems as the content supplier. This means (background) information, personalized advices, forms that enable submitted requests for permits and so on. This front office uses the same architectural knowledge backbone as the operational process. This means that they share the core taxonomies (the organizational vocabulary) mentioned before. This enables a seamless integration of the front office with the back office.

### 2.2 Semantic Models as the Brain of the Infrastructure

The knowledge workers request support on several domains in the operational process. This brought some key issues that the knowledge system had to solve:

1) Multilinguality; the application (especially the front office) needed to support multiple languages. Because Be Informed separates knowledge from presentation language specific labels are connected to the language independent models.

2) "Time travelling"; the knowledge models are implementations of laws and legislation, which are very dependent on time. We used timelines on (parts of) knowledge models to specify when they were relevant. Because all possible timelines are available in runtime this enables support for all cases, past, present and future, in the same operational process.

3) Skill based support; the knowledge workers involved in the process have different skills and authorization. Using typed textual references on models together with configuring user group related instruments, enabled us to show different users different (explanatory) texts.

4) Traceability & compliance; each decision made by the decision support system needs to be traceable and compliant to laws and legislations. Partly this is covered by the software, but an important factor here is the Be Informed Methodology for Knowledge Maintenance[9]. Key in this methodology is that all models are built according to a metamodel. An overview of part of the IND metamodel is shown in figure 1.

The decision support system uses the Model-Run-Analyze principle, meaning that the models (**Model**) are directly used in the operational process, there is no coding in between. This makes the model directly used in the day to day work (**Run**), which in its turn results into a direct feedback loop (**Analyze**).

## 3 Demonstration

The audience will learn the importance of the separation of concerns and what this means in an operational process, and basically why this has won the Architecture Award 2009 in the Netherlands. The demonstration will give an overview of the architecture, but mainly focuses on the way decision support is integrated in the case system. The various levels of support for the knowledge workers, depending on authorization and skill level and the way decision support is used to guide the knowledge worker in the whole operational process.

We will show the metamodel including some examples of operational models, instruments and web services. This will be done using the model based catalog which is all about knowledge management and always in sync with the operational knowledge, because it uses the same knowledge models. Besides the architectural choices and implementation examples we will also demonstrate how we handled the mentioned knowledge related issues. Naturally, this will all be shown in the live application.

## References

1. Antoniou, G.: A Semantic Web Primer, 2nd edn. The MIT Press, Cambridge (2008), ISBN-13: 978-026201242
2. Erl, T.: Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs (2005), ISBN-13: 978-0131858589
3. Heller, R., van Teeseling, F.: Knowledge Applications for Life Events. In: The Semantic Web: Research and Applications: 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4 (2009)
4. IND, Dutch Immigration Office, http://www.ind.nl
5. IND Verblijfwijzer, http://www.verblijfwijzer.nl
6. Nederlands Architectuur Forum, http://www.naf.nl/nl/naf-prijzen/
7. OWL specification, http://www.w3.org/TR/owl-semantics/
8. RDF Specification, http://www.w3.org/TR/rdf-concepts/
9. Van Teeseling, F., Heller, R.: Patterns in Metamodelling. In: Be Informed White paper 2008 (2008)
10. Van Teeseling, F., Heller, R.: Business Patterns in Ontology Design. In: Business Information Systems: 12th International Conference, BIS 2009, Poznan, Poland (2009)
11. Verbeek, J., Rensen, G.: A Citizen Centric Government Enabled by Semantic Technologies. In: Semantic Technology Conference, San Jose, CA, May 18-22 (2008)

# Verifying and Validating Multi-layered Models with OWL FA Toolkit

Nophadol Jekjantuk[1], Jeff Z. Pan[1], and Gerd Gröner[2]

[1] University of Aberdeen, United Kingdom
[2] University of Koblenz-Landau, Germany

**Abstract.** This paper details the use of OWL FA Toolkit for verifying and validating multi-layered (meta-) modelling using ontologies described in OWL FA. We will show how OWL FA and its reasoner (OWL FA Toolkit) could benefit the software modeller on leveraging the software development life cycle through a practical use case.

## 1 Introduction

Metamodelling, i.e. modelling across multiple modelling layers, dealing with concepts and meta-concepts, is a key issue in model management and especially in model-driven software development (MDSD). Metamodels appear in application areas such as UML [14], Model Driven Architecture [4] and E-Commerce.

Ontologies and the Web Ontology Language (OWL) are well established for model descriptions and model management tasks. However, the standard OWL Web Ontology Language does not support modelling and reasoning over a layered metamodelling architecture. OWL 2 provides simple metamodelling which corresponds to the contextual semantics defined in [9]. However, this modelling technique is mainly based on punning. It has been shown in [12] that this can lead to non-intuitive results, since the interpretation function is different based on the context.

Indeed, for many applications, a validation without a metamodel is not adequate. There are various works which consider validation of UML models with OCL constraints like in [3,5,7,13,8]. However, none of these approaches account for a validation across multiple layers, i.e. validate models with respect to their metamodels. Although they validate models with model constraints and instances of the models, they do not account for metamodels in their validation. We intent to show how OWL FA Toolkit could be used to help close the gap.

## 2 Motivating Example

This section gives an example to demonstrate the need for metamodelling enabled ontologies. Models are depicted in UML notations. Metamodels are more than a syntactic language description of a modelling language; a metamodel is a description of the concepts of a modelling language specifying the structure and the kind of information that can be handled [10].

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 391–395, 2010.

Models and metamodels are commonly used in model-driven software engineering (MDSE). In order to improve software development processes, new technologies which provide reasoning support like consistency checking of models and metamodels are beneficial. In MDSE, each model layer can contain both class and object definitions. However, this leads to undecidability problems in model validation w.r.t the complexity of the model. In ontology engineering, ontologies for metamodelling like OWL FA separates class and object into different layers in order to maintain the decidability of the language.

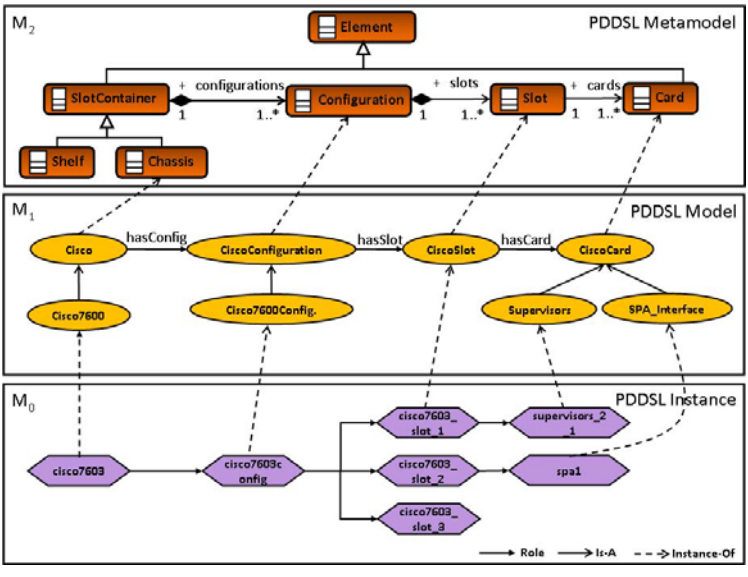



**Fig. 1.** Layered Architecture for a Physical Device Model

In Fig. 1, a layered modelling architecture is demonstrated for physical device modelling (an application of configuration management). The figure depicts three layers $\mathsf{M}_0$, $\mathsf{M}_1$ and $\mathsf{M}_2$. $\mathsf{M}_3$ is a Meta-metamodel layer which is not included in the example. The arrows between the layers demonstrate instance relationships, the arrows within a layer are concept relations (like object properties and subclass hierarchies). Moreover, the modeller requires that the concepts Chassis and Shelf in $\mathsf{M}_3$ have to be disjoint from each other.

A crucial task in model-driven engineering is the validation of models and metamodels [3,5,7,13,8]. A valid model refers to its metamodel and satisfies all the restrictions and constraints. However, the validation of multiple layers may lead to inconsistency even if the consistency is satisfied between all adjacent layers, i.e. model and instance layer. For instance, in the previous described scenario, if one would like to add Avaya is an instance of Shelf in $\mathsf{M}_2$ and Avaya, considered as a concept in $\mathsf{M}_1$, is equivalent to the concept Cisco in $\mathsf{M}_1$. Combined with the constraints on the model layer $\mathsf{M}_2$ which requires the disjointness of

class Shelf and Chassis leads to a contradiction and therefore to an inconsistent ontology. Without capturing multiple layers, this inconsistency is not detected since Avaya is either considered as a concept on layer $\mathsf{M}_1$ or as an instance. However, modelling with more than two layers has to account for concepts on layer $\mathsf{M}_i$ that are at the same time instances of concepts of the layer $\mathsf{M}_{i+1}$.

## 3 Technical Background

OWL FA [12] enables metamodelling. It is an extension of OWL DL, which refers to the description logic $\mathcal{SHOIN}(\mathcal{D})$. Ontologies in OWL FA are represented in a layered architecture. This architecture is mainly based on the architecture of RDFS(FA) [11]. OWL FA specifies a layer number in class constructors and axioms to indicate the layer they belong to. For example, `SubClassOf(Annotation (Layer "2"^^xsd:int) Chassis SlotContainer)` belongs to layer 2. Since OWL FA is based on OWL DL all language features from OWL DL are also available in OWL FA in contrast of RDFS(FA) like transitive `TransitiveProperty` and inverse `owl:inverseOf` property restrictions.

The syntax is adopted from OWL DL. The semantics of two layers which can be considered as TBox and ABox are same as in OWL DL. The idea of OWL FA is that the interpretation depends on the layer but is still an OWL DL interpretation. Therefore in each layer, or from one layer to the next layer standard OWL DL reasoning capabilities can be used. For more details about the reasoning in OWL FA refer to [6].

## 4 OWL FA Toolkit

In this section, we introduce the OWL FA Toolkit, a simple graphic user interface for modeller to create an OWL FA ontology and perform reasoning over it. The OWL FA Toolkit contains features as following:

- Editor - for checking the OWL FA ontology before perform the reasoning. In this version it supports only functional syntax.
- Ontology Consistency - for checking whether a given multi-layered models is valid. This enables modeller to validate models with their metamodels.
- Concept Satisfiability - for verifying whether a concept $A$ is a non-empty set in a given OWL FA ontology $\mathcal{O}$. A modeller can verify a particular concept that might leads to contradiction of the model.
- Query Answering - for accessing information form a given multi-layered models by using SPARQL query.
- Reasoning with OWL DL ontologies - for reasoning with a multi-layered models that described in OWL DL syntax. A modeller can easily create model ontology and metamodel ontology separately.

Figure 2 shows screen capture of OWL FA Toolkit loaded OWL FA ontology for validating multi-layered models. More details about the OWL FA Toolkit are described in [6].

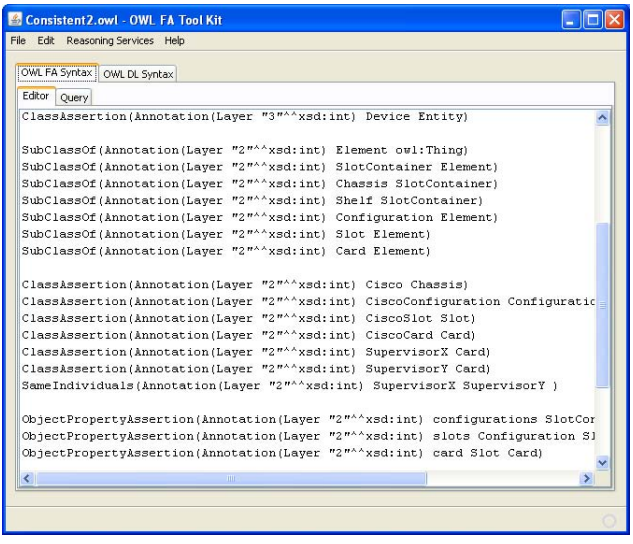



**Fig. 2.** OWL FA Toolkit

## 5 Related Work

In [15] spanning objects are used in order to have different interpretations for objects that are instances and classes simultaneously. Compared to OWL FA one spanning object refers to one ontology $\mathcal{O}_i$.

A UML diagram with OCL constraints is transformed to a constraint satisfaction problem. In [13] UML/OCL models are modelled with the constraint language Z in order to validate class diagrams. The language Alloy is used in [1] as a representation of UML/OCL models. The Alloy Analyser verifies the model properties. Constraint logic programming (CLP) is applied in [8] to validate UML models and model constraints. Also the metamodels are translated to CLP and validated based on defined metamodel specifications. However, properties of a layer are not considered in the next layer.

Berardi et al. [3] apply DL Reasoning to UML class diagrams. The expressiveness of UML diagrams and constraints are restricted to the expressiveness of the DL $\mathcal{ALC}^-$. Basic conceptual modelling including model constraints is demonstrated for UML diagrams in OWL The consistency check of a UML class diagram is then reduced to concept satisfiability in $\mathcal{ALC}^-$. However, the verification is only performed on the conceptual level, without accounting for a metamodelling architecture.

First-order logic (FOL) is used in [7] for consistency checks of UML class diagrams. The main contribution are different algorithms to perform the consistency check and the analysis of inconsistency triggers. The transformation from UML class diagrams with OCL constraints to FOL is also described in [2] in order to enable consistency check.

## 6 Conclusion

This paper focuses on showing how to use OWL FA Toolkit for verifying and validating multi-layered models. We also compare our tool with OWL 2 modelling and reasoning, since OWL 2 is the only OWL language that supports metamodelling and has tool support. For demonstration purposes, we use some case studies from the MOST project (http://www.most-project.eu), in particular, the physical device modelling (an application of configuration management). We will show how OWL FA and its reasoner (OWL FA Toolkit) could benefit software modellers on leveraging the software development life cycle.

## References


1. Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: UML2Alloy: A challenging model transformation. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 436–450. Springer, Heidelberg (2007)
2. Beckert, B., Keller, U., Schmitt, P.H., et al.: Translating the Object Constraint Language into first-order predicate logic. In: Proceedings, VERIFY, Workshop at Federated Logic Conferences (FLoC), Copenhagen, Denmark, Citeseer (2002)
3. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
4. Brown, A.: An introduction to Model Driven Architecture. IBM Technical Report (2004), http://www-128.ibm.com/developerworks/rational/library/3100.html
5. Cabot, J., Clariso, R., Riera, D.: Verification of UML/OCL Class Diagrams using Constraint Programming. In: Software Testing Verification and Validation Workshop, pp. 73–80 (2008)
6. Jekjantuk, N., Gröner, G., Pan, J.Z.: Reasoning in Metamodeling Enabled Ontologies. In: Proceeding of the International Workshop on OWL: Experience and Directions, OWL-ED 2009 (2009)
7. Kaneiwa, K., Satoh, K.: Consistency checking algorithms for restricted UML class diagrams. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 219–239. Springer, Heidelberg (2006)
8. Malgouyres, H., Motet, G.: A UML Model Consistency Verification Approach based on Meta-Modeling Formalization. In: SAC 2006: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 1804–1809. ACM, New York (2006)
9. Motik, B.: On the properties of metamodeling in owl. J. Log. Comput. 17(4), 617–637 (2007)
10. Ober, I., Prinz, A.: What do we need metamodels for?
11. Pan, J.Z., Horrocks, I.: RDFS(FA) and RDF MT: Two Semantics for RDFS. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 30–46. Springer, Heidelberg (2003)
12. Pan, J.Z., Horrocks, I., Schreiber, G.: OWL FA: A Metamodeling Extension of OWL DL. In: Proceeding of the International Workshop on OWL: Experience and Directions, OWL-ED 2005 (2005)
13. Roe, D., Broda, K., Russo, A.: Department of Computing. Mapping UML models incorporating OCL constraints into Object-Z. In: Imperial College of Science, Technology and Medicine, Department of Computing (2003)
14. UML. Unified Modeling Language, http://www.uml.org/
15. Welty, C.A., Ferrucci, D.A.: What's in an instance? Technical report, RPI Computer Science (1994)

# What's New in WSMX?

Srdjan Komazec and Federico Michele Facca

Semantic Technology Institute (STI) - Innsbruck,
ICT Technologiepark, Technikerstrasse 21a, 6020 Innsbruck, Austria
firstname.lastname@sti2.at

**Abstract.** The Web Service Execution Environment (WSMX) is the most complete implementation of a Semantic Execution Environment to support the automation of the Web service life-cycle. WSMX is a constantly evolving project. The demo will provide insight and justify the value of the newly introduced features: the Complex Event Processing engine, the Notification Broker engine and the Orchestration engine.

## 1 Introduction

A Semantic Execution Environment (SEE) represents a class of middleware solutions that support the common service-related life-cycle tasks through the exhaustive use of machine-processable service descriptions. The most comprehensive existing SEE implementation to date is the Web Service Execution Environment (WSMX)[1]. WSMX is a reference implementation of Web Service Modeling Ontology (WSMO) [1]. WSMX is adopted as one of the reference architectures of OASIS Semantic Execution Environment (SEE) Technical Committee[2]. In the past edition of the demo [2] at ESWC'09 we demonstrated a new WSMX version implementing some new features: Ranking engine, Grounding engine and Monitoring facilities. In this demo we will focus on the latest WSMX advancements comprised of:

- Complex Event Processing engine that, coupled with the existing WSMX monitoring facilities, enables the next level of the environment agility;
- Notification Broker engine that enables the asynchronous notification of discovery results;
- Orchestration engine that enables the orchestration of Semantic Web services; and
- A set of smaller improvements such as a new format for Web Service discovery results, caching of the discovery results and security facilities.

The paper is structured as follows: Section 2 updates the WSMX architecture; in Section 3 we present the advancements in SEE monitoring with special emphasis on the integrated complex event processing engine; Section 4 presents the asynchronous notification of discovery results; in Section 5 we describe the

[1] http://www.wsmx.org
[2] http://www.oasis-open.org/committees/semantic-ex

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 396–400, 2010.

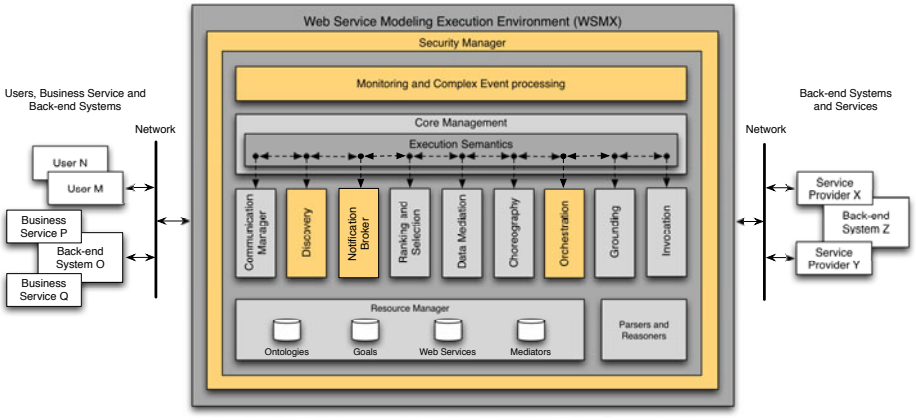



**Fig. 1.** The advanced WSMX architecture

orchestration engine; Section 6 gives a brief overview of other smaller improvements; finally in Section 7 we draw some conclusions and describe the intended demonstration plan.

## 2 Updated WSMX Architecture

WSMX adopts a typical multi-layered service-oriented architecture with the goal of minimizing cross-cutting concerns and encapsulate functional responsibilities, as shown in Figure 1. The top layer is devoted to the Core Management facilities, which orchestrate the lower-level broker services by providing data and control flow capabilities in order to enable execution of useful processes such as: Web service discovery, Web service invocation and a combination of the two. The middle layer is represented by the set of broker services offering the basic functionalities (e.g., discovery, ranking, choreography, etc). The lower-level provides a set of fundamental logistical services such as management of WSMO-related artifacts as well as parsing and reasoning functionality. More details about the WSMX broker services can be found in [3]. In contrast to the architecture shown in the previous demo [2], the following differences can be identified: First, the integration of the Complex Event Processing engine as a part of the monitoring facilities of the platform that enables timely responses to complex situations of particular interest (described more in Section 3). Second, the addition of the Notification Broker engine, which enables asynchronous discovery of Web services fulfilling the submitted goals and notification of the goal owners by relying on publish/subscribe mechanisms (described more in Section 4). Third, the addition of the Orchestration engine, which enables the execution of complex Semantic Web service compositions (explained in a bit more detail in Section 5). The discovery and ranking processes have also undergone some smaller improvements. Additionally, a component to manage user authentication and authorization has been introduced. These advancements are described briefly in Section 6.

## 3 Complex Event Processing

A SEE implementation generates a rich source of events, including events emitted at the level of the overall platform (e.g. at the start of the invokeWebService process), events at the level of single broker services (e.g. tracking of the functionality provided by the Discovery engine) and events at the level of the external services (e.g. a fault during a Web service invocation). Since the events have ontological representations, the complete event history is preserved in an RDF repository for convenience purposes (support for massive storage, inferencing and querying). Before reaching the repository, the generated events must be adequately processed in order to detect and react in a timely manner to the complex situation of interest (e.g. three failed attempts to invoke a particular Web service may trigger another round of Web service discovery to select the second best solution if such exists).

WSMX now integrates an RDF-based Complex Event Processing engine (RDF-CEP) built on top of the existing WSMX monitoring facilities described in [4] and demonstrated in [2]. The primary objective of RDF-CEP is to detect occurrences of RDF triples satisfying expressive RDF patterns. When a complex event is detected, the engine executes the appropriate actions associated with the detected event (e.g. update of the aggregated statistical measures, notification of the administrator about successive failed attempts to access WSMX facilities). The engine can be also used outside of the WSMX environment as a general purpose tool for filtering and processing RDF data streams.

## 4 Notification Broker

Asynchronous communication within WSMX is enabled through the newly introduced broker service called the 'Notification engine'. This service is responsible for registering and storing user subscriptions and related results (in order to avoid duplicate notifications) and sending notifications of results related to user subscriptions. For now, WSMX supports subscriptions to goal-based discovery, but the mechanism can easily be extended to other execution semantics.

To support the full process, a new execution semantics was created that is executed as soon as a user subscribes to a goal or a new Web service is registered in the platform. This execution semantics checks for existing (and not expired) user subscriptions, and executes for each subscribed goal the discovery process. Once new Web services have been matched w.r.t. previous notifications, they are notified to the users. The notification occurs according to the mechanism selected by the user. Currently we support email or Web service invocation. In the later case a subscriber must implement a specific WSDL definition. A wider discussion related to the solution implemented in WSMX can be found in [5].

## 5 Web Service Orchestration

Choreography, as currently used in WSMO/L/X, involves a client communicating with several components to achieve the desired result. In contrast, orchestration

involves not just client-component communication, but also communication directly between components. The Orchestration engine introduced in WSMX is conformant with the WSMO dataflow-based orchestration language as defined in [6]. The engine supports dataflow in a manner consistent with WSMO/L, and adopts the explicit concept of *performance*, and a new type of mediator to mediate between performances.

In a nutshell the engine and the underlying language solve the following issues:

- mediation in any connection, including dataflow, between heterogenous components;
- *extraction* and *aggregation* in the consumption and production of messages according to the WSML grounding approach; and
- execution of service discovery and service invocation within composed services.

## 6 Other Minor Improvements

To keep the architecture up-to-date with results from the OASIS SEE-TC, we refined the discovery and ranking engine so that it now returns, instead of a set of services, a set of Web service-to-goal mediators (wgMediator). In this way we are able to associate the service with the goal, and to attach to this tuple results coming from the discovery or ranking steps, such as type of the match, value of the non-functional properties associated with the ranking results, goal specific quotes, and so on. All this information enriches the knowledge on why and how a certain service has been associated with a certain goal, thus giving better support for the selection of services.

The results of the discovery process are now cached. This enables a reduction in the computational time associated with discovery when a goal is submitted multiple times by relying on previous results. The cache is invalidated each time a new service is (un)registered in the system.

Last but not least, WSMX now introduces a set of security features as the first line of defense against malicious use of the offered functionality. Basically, authentication, authorization, and accounting for both SOAP and Web-based WSMX endpoints has been implemented. Security is based on the developed security ontology following the Role-based Access Control model. While authentication verifies supplied security credentials against the ontology, authorization checks whether the client has sufficient privileges to consume particular WSMX functionality. Integrated with the monitoring RDF store (see Section 3) the accounting facilities provide the possibility to track the consumption of WSMX functionality and resources.

## 7 Conclusions and Demonstration Plan

After completion of the basic SEE requirements, WSMX is now on track to embrace advanced techniques in order to make it a more controllable and self-aware

environment suitable for enterprise-level use. This paper presents some of the techniques, namely the inclusion of a Complex Event Processing engine coupled with the existing Monitoring facilities, a deferred goal Notification mechanism and an Orchestration engine, together with some smaller improvements related to the Web service discovery process and security concerns.

The prospective attendees to the demo will be able to examine and simulate different situations on a live WSMX instance prepared with a number of Semantic Web Service descriptions coming from the Enterprise Interoperability and Enterprise Collaboration domain. In particular, an attendee will be able to submit goals for asynchronous execution and then examine the platform behavior when suitable subsequent Web services are registered and notifications generated. In the context of complex event processing, the attendee will be able to change and adapt the event pattern descriptions. After executing the appropriate WSMX processes, the attendee will observe the behavior of the engine. The Orchestration engine will be demonstrated with an example of a Web service composition showing all the features of the WSMO orchestration language. The platform internals will be exposed through the Web console that will graphically presents the current status of the system and its evolution.

## References

1. 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 (2006)
2. Facca, F.M., Komazec, S., Toma, I.: WSMX 1.0: A Further Step toward a Complete Semantic Execution Environment. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E.P.B. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 826–830. Springer, Heidelberg (2009)
3. Fensel, D., Kerrigan, M., Zaremba, M. (eds.): Implementing Semantic Web Services: The SESA Framework, 1st edn., May 2008. Springer, Heidelberg (2008)
4. Komazec, S., Facca, F.M.: Towards a Reactive Semantic Execution Environment. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 877–887. Springer, Heidelberg (2009)
5. Facca, F.M., Komazec, S., Zaremba, M.: Towards a Semantic Enabled Middleware for Publish/Subscribe Applications. In: ICSC, pp. 498–503. IEEE Computer Society, Los Alamitos (2008)
6. Norton, B., Haselwanter, T.: Dataflow for Orchestration in WSMO. In: Working Draft d15.1, WSMO Working Group (2007), http://www.wsmo.org/TR/d15/d15.1/v0.1/

# OntoFrame S3: Semantic Web-Based Academic Research Information Portal Service Empowered by STAR-WIN

Seungwoo Lee, Mikyoung Lee, Pyung Kim, Hanmin Jung, and Won-Kyung Sung

Korea Institute of Science and Technology Information (KISTI),
335 Gwahangno, Yuseong-gu, Daejeon, Korea 305-806
{swlee,jerryis,pyung,jhm,wksung}@kisti.re.kr

**Abstract.** We have developed a prototype of a practical knowledge-driven semantic portal, OntoFrame S3, which provides various reasoning-based analysis services on academic research information. To realize this semantic portal, we developed and applied several Semantic Web and linguistic technologies. Through this demonstration, we will show how Semantic Web technologies can be utilized for information connection and fusion in the academic research information service sector and empowered by linguistic knowledge.

**Keywords:** Academic research information portal service, semantic search, ontology, reasoning, semantic word network.

## 1 Introduction

According to the US National Science Foundation (NSF)[1], researchers are spending more than half of their total research and development hours for hunting information. Thus, to allow researchers to have more time to spend on research and development itself, it is crucial to reduce time spent on gathering information. Currently, leading search engines only provide a keyword-based matched list as the result of a search query, which is limited in terms of accuracy and efficiency of information comprehension. A new type of information service is required that can find the information desired by the researcher, and then connect, combine, and analyze it to provide as much value to the user as possible. To address this need, we have developed a prototype of a knowledge-driven semantic portal, OntoFrame S3[2], which provides various analysis services on academic research information combining OntoFrame and STAR-WIN, which are implemented by Semantic Web and linguistic technologies.

In this demonstration, we will show you various semantic analysis services focused on topics and researchers. Topic-focused services include trends, leading researchers and researcher networks on a topic and researcher-focused services include research areas, research outputs, co-working researchers, and similar researchers of a researcher. More detail description about these services is given in the next section.

[1] http://www.nsf.gov/
[2] S3 means STAR-WIN-enabled Semantic Service.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 401–405, 2010.

## 2 Related Work

There are some research information service systems similar to OntoFrame S3. BiomedExperts (www.biomedexperts.com) and Authoratory (authoratory.com) provide useful services such as domain experts, researcher and article details, statistics, researcher network and researcher map in biomedical domain mainly dependent on PubMed[3] database. ResearchGATE (researchgate.com) also provides similar network community services like domain experts, researcher details, and researcher network. It requires users to provide the profiles of their own and colleagues. These service systems have very similar features to ours in the viewpoint of service functions. However, they are typical database applications based on data mining technologies whereas OntoFrame S3 is developed with Semantic Web technologies such as ontology and reasoning, which enable to achieve a flexible and precise service in both connecting knowledge and planning services.

## 3 Technologies Used

OntoFrame S3 shows a service example that combines Semantic Web technologies, implemented on OntoFrame, and linguistic technologies, implemented in STAR-WIN.

OntoFrame is an information service platform that uses Semantic Web technologies. It includes OntoURI – a semantic knowledge management tool that creates ontology instances by referring ontology schemata and resolves co-references between ontology individuals; and OntoReasoner – a reasoning engine that stores and infers ontology-based RDF triples and answers SPARQL queries. It also involves Mariner – a commercial search engine provided by DiQuest (www.diquest.com) – to provide search functionalities. STAR-WIN (Science & Technology Assister – Word Intelligent Network) is a semantic word network of technical terms that represents semantic and conceptual relationships among the terms in science and technology domain. In the following subsections, we explain each component in detail.

### 3.1 Ontology Engineering

The goal of OntoFrame S3 is to provide connection, fusion, and analysis services on academic research information to enable researchers to effectively obtain information needed for their work. In order to achieve the goal, we have developed the KISTI Reference and Academic Ontologies[4]. They model research agents such as persons and institutions, their accomplishments such as articles, reports and patents, publications which indicate specific journal issues or proceedings, locations and topics. In contrast with other research-related ontologies, our ontology connects researchers to their affiliations of which they were members at the time they had their accomplishments. It also connects institutions to their locations such as countries, states and cities. The ontology schemata are composed of 16 classes and 89 properties, and designed using RDF, RDFS and OWL (strictly, OWL-Lite) vocabularies.

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

[4] http://isrl.kisti.re.kr/ontologies/ReferenceOntology1_0.owl,
http://isrl.kisti.re.kr/ontologies/AcademicOntology1_0.owl

Currently, OntoFrame S3 is in service as a practical prototype (http://www.ontoframe.kr/S3/english). It contains about 4 million articles in English, which have been written by 12.7 million researchers, on 400,000 topics, who work for 90,000 institutions, spanning 410,000 locations. In total, the system has a total of about 826 million RDF triples. These data were populated partly from CiteSeer OAI (Open Archive Initiative) metadata[5] and partly from NDSL (National Digital Science Links) metadata[6]. We applied an elaborate process to identify and disambiguate same-name authors and populate them into ontology instances [1].

### 3.2 Storing and Reasoning

The populated ontology instances are stored and inferred using OntoReasoner [2]. That is, OntoReasoner consists of roughly two parts: Triple store and Reasoner. The triple store gets ontology instances and stores them to back-end DBMS such as MS-SQL Server according to predefined table schema. It also can answer to a query represented in SPARQL, which is converted into appropriate SQL and executed. The reasoner performs rule-based reasoning based on RDF Semantics[7] and OWL Semantics[8] in ways of forward-chaining. It currently supports entailments of full RDFS and some of OWL vocabularies such as *owl:inverseOf*, *owl:sameAs*, and *owl:TransitiveProperty*.

### 3.3 Semantic Word Network

OntoFrame S3 is empowered by STAR-WIN to get linguistic flexibility. STAR-WIN is a semantic work network for Korean language developed by KISTI and represents semantic and conceptual relationships between words [3]. It covers technical terms[9] of science and technology domain. It also includes Korean-English translation information for technical terms. Thus, it is highly applicable to linguistic information applications such as information retrieval and machine translation.

Linguistic flexibility of OntoFrame S3, provided by STAR-WIN, includes following three points: auto-complete of search keywords, semantic word network for keyword extension, and Korean-English translation of search keyword. Auto-complete of keyword also functions as keyword recommendation. Semantic word network shows various relations such as broad and narrow terms, sibling terms, similar terms and other related terms. Keyword translation makes it possible to query in Korean although the ontology instances only have English literals.

## 4 Services

OntoFrame S3 is designed to be topic and researcher-centric (Fig. 1). This is because authors (i.e., researchers) and research topics are useful starting points for finding academic research information. When a user performs a search, a keyword-based search engine is used to find results, but URIs for the terms are also retrieved and

[5] http://citeseer.ist.psu.edu/oai.html
[6] http://www.ndsl.kr
[7] http://www.w3.org/TR/rdf-mt/
[8] http://www.w3.org/TR/owl-semantics/
[9] Currently, it contains about 350,000 technical terms.

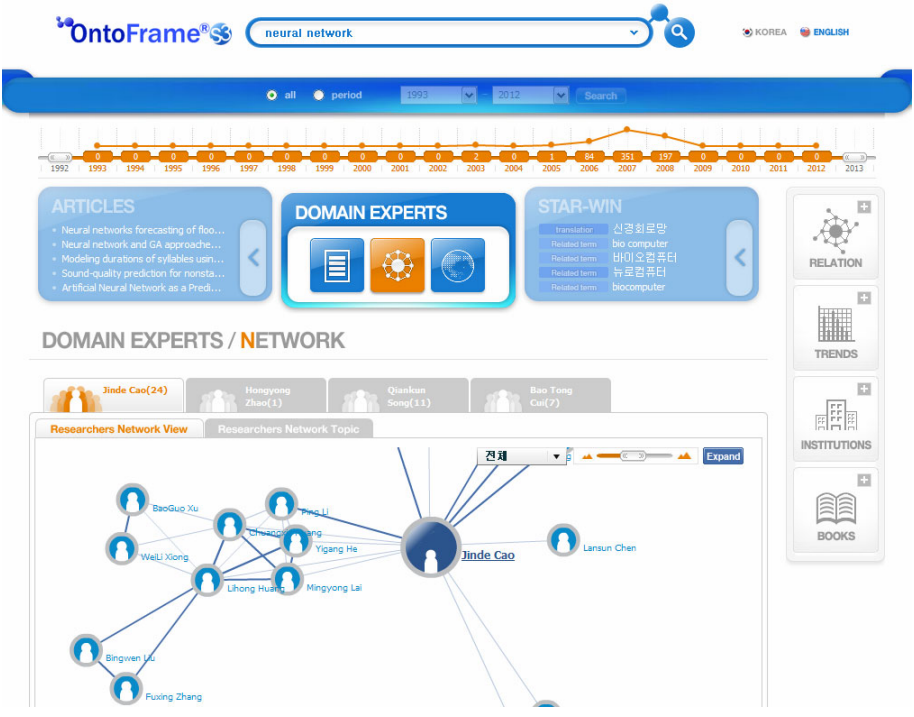



**Fig. 1.** OntoFrame S3 service snapshot

used to search the ontology instances for more information relating to the term. The results are presented to the user by appropriate visualization.

The semantic portal currently implements and provides many service components such as topic trends, identification of leading researchers or research institutes for a topic, main research topics of a researcher, researcher publication network, research trends of a publication network, recommendation of associated researchers, similar researchers, similar papers, and geographical distribution of researchers. These services provide knowledge that has been analyzed by connecting and fusing fractional information along with proper visualization.

The service prototype has been designed and developed considering user-centric service layout and usability. We selected three main services considering user preference and distinguished these services from add-on services considering service priority. Through this, each service component was properly placed at main area or right area of service user interface.

## 5 Conclusion

We have developed a prototype of a knowledge-driven semantic portal, OntoFrame S3, which provides various analysis services on academic research information combining OntoFrame and STAR-WIN. This semantic portal service shows how Semantic Web

technologies can be utilized for information connection and fusion in the academic research information service sector and empowered by linguistic knowledge.

We will continue to evaluate the services and develop new useful services to open real services to the public. Thus, we plan to measure the usefulness of service attributes by comparatively evaluating OntoFrame S3 and other similar service systems mentioned in the Related Work.

## References

1. Kang, I.-S., Na, S.-H., Lee, S., Jung, H., Kim, P., Sung, W.-K., Lee, J.-H.: On co-authorship for author disambiguation. Information Processing and Management 45, 8–97 (2009)
2. Lee, S., You, B.-J.: Memory and DBMS-based Hybrid Reasoning. In: Korea Computer Congress 2009, Korea, pp. 122–123 (2009) (in Korean)
3. Lee, S.-H., Choe, H.-S., Yoon, H.-M., You, B.-J.: KISTI-STAR: Semantic Technology based Science and Technology Knowledge Information Service. In: Service Science Society 2009 Spring Conference, Korea, pp. 1–8 (2009) (in Korean)

# Rapid Prototyping a Semantic Web Application for Cultural Heritage: The Case of MANTIC

Glauco Mantegari, Matteo Palmonari, and Giuseppe Vizzari

Department of Computer Science, Systems and Communication (DISCo),
University of Milan - Bicocca,
via Bicocca degli Arcimboldi 8, 20126 - Milan, Italy
{mantegari,palmonari,vizzari}@disco.unimib.it

**Abstract.** MANTIC[1] is a Web application that integrates heterogenous and legacy data about the archeology of Milan (Italy); the application combines semantic Web and mashup technologies. Semantic Web models and technologies supports model-driven and standard-compliant data integration on the Web; the mashup approach supports a spatial and temporal aware form of information presentation. MANTIC shows that model-driven information integration applications for cultural heritage can be fast prototyped with limited deployment effort by combining semantic and mashup technologies. Instead, higher-level modeling aspects need a deep analysis and require domain expertise.

## 1 Introduction and Motivation

Semantic Web technologies (SWT) natively provide support for model-based information integration, exchange and processing by offering Web-compliant knowledge representation languages (e.g. RDF, RDFS and OWL), query and reasoning engines. Mashup technologies (MTs) [1] provide support for simple application integration and for aggregating heterogeneous information on the Web. Combining SWTs and MTs it is possible to build data-driven Web applications with limited efforts and costs, by reusing available information sources and models, and information presentation layers. Model-driven information integration, and reuse of application services are particularly attractive in the field of cultural heritage (CH) because of a number of issues that characterize the domain.

First of all, available funds are often limited, in particular when an application is not part of a large national or international project. Second, although a ISO standard conceptual reference model for CH has been recently developed, i.e. the CIDOC CRM[2], most of data sources are still based on legacy models and non trivial ontology-based mappings are needed in order to support information integration. Furthermore the frequent and heterogeneous spatial and temporal references (e.g. multiple spatial references and different historical classification systems), require reasoning capabilities (e.g. on temporal intervals); such references are crucial to support the access to the information (e.g. by map-based result presentation and timelines rendering), which makes the

[1] http://www.lintar.disco.unimib.it:8080/mantic/
[2] http://cidoc.ics.forth.gr/

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 406–410, 2010.

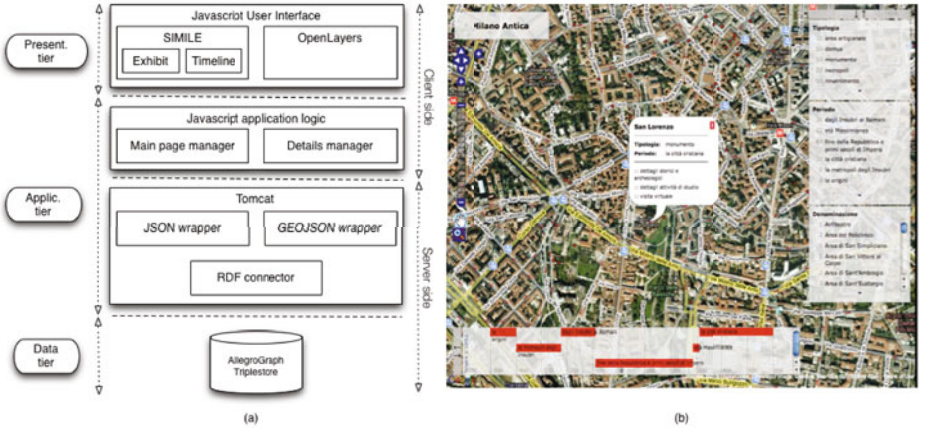



**Fig. 1.** Mantic Architecture (a) and a screenshot of the navigation page (b)

reuse external services (e.g. Google Maps) and software components (e.g. the SIMILE [3] timeline representations) particularly useful. Finally, rich and deeply nested descriptions due to the modeling choices of the CIDOC CRM (and the related OWL-DL implementation presented in [2]): the relevant information describing, for instance, a monument is represented as a very nested graph [3] and this requires reasoning engines (e.g. to retrieve data through the subclass chains down to the actual sources) as well as optimization techniques (e.g. because of expensive query answering).

MANTIC is a Web application that, considering the above issues, realizes a portal for archaeological information concerning the city of Milan, which is based on the integration of SWTs and MTs: a Web mashup approach provides browsing and navigation functionalities on a semantic repository integrating legacy information sources (a Web portal and existing relational databases). The mashup approach allowed to reuse available services and software components focusing on a set of core aspects related to the modeling activity. MANTIC shows that by combining SWTs and MTs it is possible to fast prototype model-driven information integration applications with limited deployment effort and budget (the application is based on free software and required 5 person/months of a bachelor-level computer scientist), and focusing on higher-level modeling aspects that require domain expertise (activities of existing data schemas analysis and modeling/mapping definitions required 6 person/months or a domain expert).

The application shows that effective results can be obtained also by exploiting data available from the deep Web and with very limited funds on national or international scale. Actually, most of the relevant works carried out in developing semantic Web applications in the field of CH comes from european project or equivalent national ones (e.g. eCHASE[4], Culture Sampo[5], STITCH[6], The Vbi Erat Lvpa [7], STAR [8] and Perseus-Arachne [9] projects). Conversely, our application shows the effectiveness of combining MTs and SWTs even in smaller, region-scale contexts.

[3] http://simile.mit.edu/

In this paper, we focus on two main classes of outcomes of the project: the integration of legacy and Web sources scraped from the Web, including the information extraction, modeling and mapping processes, discussed in Section 2; the system architecture and functionalities, including the SWTs used, the browsing approach, the timeline navigation, and the optimization techniques needed, discussed in Section 3. Lessons learned are also discussed at the end of the paper (Section 4).

## 2 Information Extraction, Modelling and Mapping

MANTIC integrates the following data sources:

- SIRBeC: the Regional Information System on cultural heritage is the main technological platform created by the Regione Lombardia for cataloging cultural heritage (compliant to and representative of the Italian cataloguing standards); it collects information about the items that are present in the territory or are preserved in museums, collections and other cultural institutions.
- IDRA: the Information Database on Regional Archaeological-Artistic-Architectural heritage which connects several databases related to Lombardia heritage; it collects archaeological data concerning the city of Milan (Italy); these data have been obtained by means of Web scraping techniques.
- MANTIC core: a PostgreSQL database containing synthetic information about the archaeological sites, structures and monuments of the city.

Each of the above data sources comes with a local schema and has been transformed with ETL techniques into a RDFS++ ontology. The global schema is based on the CIDOC CRM standard. In particular the OWL-DL CIDOC CRM representation defined by [2] has been used. The main challenges faced in the process concerned the following aspects:

**Event-centric global model vs Object-centric local models.** The CIDOC CRM model represented in the global schema substantially differs from the models used in the local sources. In particular, the former adopts an event-centric approach, where object descriptions are represented through descriptions of events related to these objects (e.g. the production, modification, or documentation of a given artifact). On the contrary, local sources adopt an object-centric approach, where descriptions are referred to objects themselves. A significant amount of implicit information had to be identified, analyzed, and represented to leverage this substantial ontological mismatch. The most frequent problems in the design of mappings are related to (i) different subgraphs for equivalent metadata, and (ii) identical subgraphs for different metadata. The graph representing the global schema is thus more composite than the original models; as a consequence complex graph queries are required to retrieve even relatively simple information.

**Multiple historical periods reference systems.** Different interpretation of the same periods were assumed in different data sources, and multiple classifications of historical periods are present.

**Coreference Linking.** The integration of complementary sources through a common model is intrinsically characterized by the problem of identifying possible different resource identifiers (e.g. URIs) that refer to the same real world entities. In addition, in

this context we have a problem of multiple temporal references and different classification systems for historical periods.

## 3 System Architecture and Functionalities

MANTIC architecture is shown in Figure 1. After a comparison of different triple storage we chose to use Allegro Graph[4] because of the good trade-off among the reasoning capabilities offered and efficiency. In particular, Allegro Graph natively support RDFS$^{++}$, which enrich RDFS reasoning capabilities (necessary to support type inference through subclass relationship) with support for *owl:sameas* and for inverse and transitive properties (useful respectively to support coreference, and reasoning about temporal intervals). Data obtained by querying the triple store is translated to JSON format (managed by the UI components) by a specific connector.

The main functionalities of MANTIC are:

- **Map-Based Browsing:** The system presents a view of the Milan area in which the relevant archaeological evidences (e.g. monuments and sites of interest dating between V century B.C. and VI century A.D.) are shown as points in the modern city map. The interactive mapping system is based on the OpenLayers framework[5]; it currently shows cartographic data fetched from standard Internet mapping services (i.e. Google Maps and Openstreetmap), but the integration and employment of local and more precise data from Regione Lombardia is currently being implemented.
- **Faceted Browsing and Timeline based filtering:** The system presents two types of additional UI elements that can be employed to filter the shown monuments and sites: a timeline supporting the selection of the elements according to the different phases in which the relevant time period is subdivided, and facets highlighting typological characterizations (e.g. domus, monument, necropolis) and possibly other relevant dimensions of cultural heritage description (e.g. materials, building techniques, etc.).
- **Query Optimization and Caching:** The first approaches to the definition of SPARQL queries to retrieve information from the Allegro Graph triple store were unsatisfactory: given the peculiarities of the CIDOC CRM even conceptually simple queries (e.g. retrieving basic information about monuments), required to retrieve and filter information related to several CIDOC classes, leading to moderately complex queries whose execution would not be compatible with the expected time delay for a dynamic web page generation. To tackle this issue we decided to (i) adopt caching techniques to avoid unnecessary query processing, and (ii) translated complex queries into equivalent portions of Java code employing simpler queries.

## 4 Lessons Learned and Conclusions

The combination of SWTs and MTs supports the rapid prototyping of CH Web applications for the integration and presentation of heterogeneous information from legacy

[4] http://www.franz.com/agraph/
[5] http://openlayers.org

sources with limited development efforts. In model-driven contexts such as CH, particular attention must be instead paid to modeling issues due to sensible ontological mismatches among heterogenous models. In particular, the role of domain experts with respect to the definition of mappings from heterogeneous documentation systems to the CRM becomes crucial. In the specific project, this modeling effort will simplify the integration of further information sources complying to Italian cataloguing standards. In developing such an application we experienced a trade-off between model-level qualities like richness and generality of a standard ontology (i.e. the CIDOC CRM in our case), and application-level qualities such as efficiency and simplicity. Finally, we also noticed that RDFS++ presents a good balance between reasonable scalability and useful reasoning task support for information integration and navigation.

## References

1. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing 12(5), 44–52 (2008)
2. Goerz, G., Schiemann, B., Oischinger, M.: An Implementation of the CIDOC Conceptual Reference Model (4.2.4) in OWL-DL. In: CIDOC Annual Conference, September 15-18 (2008), http://www8.informatik.uni-erlangen.de/IMMD8/Services/cidoc-crm/docu/crm_owl_cidoc2008.pdf
3. Nussbaumer, P., Haslhofer, B.: CIDOC CRM in Action – Experiences and Challenges. In: Kovács, L., Fuhr, N., Meghini, C. (eds.) ECDL 2007. LNCS, vol. 4675, pp. 532–533. Springer, Heidelberg (2007)
4. Addis, M.J., Hafeez, S., Prideaux, D., Lowe, R., Lewis, P.H., Martinez, K., Sinclair, P.A.S.: The echase system for cross-border use of european multimedia cultural heritage content in education and publishing. In: AXMEDIS 2006: 2nd International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (2006)
5. Hyvönen, E., Mäkelä, E., Kauppinen, T., Alm, O., Kurki, J., Ruotsalo, T., Seppälä, K., Takala, J., Puputti, K., Kuittinen, H., Viljanen, K., Tuominen, J., Palonen, T., Frosterus, M., Sinkkilä, R., Paakkarinen, P., Laitio, J., Nyberg, K.: CultureSampo: A National Publication System of Cultural Heritage on the Semantic Web 2.0. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 851–856. Springer, Heidelberg (2009)
6. van Gendt, M., Isaac, A., van der Meij, L., Schlobach, S.: Semantic web techniques for multiple views on heterogeneous collections: A case study. In: Gonzalo, J., Thanos, C., Verdejo, M.F., Carrasco, R.C. (eds.) ECDL 2006. LNCS, vol. 4172, pp. 426–437. Springer, Heidelberg (2006)
7. Doerr, M., Schaller, K., Theodoridou, M.: Integration of complementary archaeological sources. In: Proceedings of the 32nd Computer Applications and Quantitative Methods in Archaeology Conference (2004)
8. Binding, C., May, K., Tudhope, D.: Semantic Interoperability in Archaeological Datasets: Data Mapping and Extraction Via the CIDOC-CRM. In: Christensen-Dalsgaard, B., Castelli, D., Ammitzbøll Jurik, B., Lippincott, J. (eds.) ECDL 2008. LNCS, vol. 5173, pp. 280–290. Springer, Heidelberg (2008)
9. Babeu, A., Bamman, D., Crane, G., Kummer, R., Weaver, G.: Named Entity Identification and Cyberinfrastructure. In: Kovács, L., Fuhr, N., Meghini, C. (eds.) ECDL 2007. LNCS, vol. 4675, pp. 259–270. Springer, Heidelberg (2007)

# Hey! Ho! Let's Go! Explanatory Music Recommendations with dbrec

Alexandre Passant and Stefan Decker*

Digital Enterprise Research Institute,
National University of Ireland, Galway
firstname.lastname@deri.org

**Abstract.** In this demo paper, we present dbrec (http://dbrec.net), a music recommendation system using Linked Data, where recommendation are computed from DBpedia using an algorithm for *Linked Data Semantic Distance* (*LDSD*). We describe how the system can be used to get recommendations for approximately 40,000 artists and bands, and in particular how it provides explanatory recommendations to the end-user. In addition, we discuss the research background of dbrec, including the *LDSD* algorithm and its related ontology.

## 1 Research Background

### 1.1 Measuring Semantic Distance on Linked Data

The underlying research question behind dbrec is to understand how to define semantic distance [4] measures for resources that follow the Linked Data principles [1]. In order to define such measures, we first defined a theoretical model for Linked Data datasets, as follows.

**Definition 1.** *A dataset following the Linked Data principles is a graph $G$ such as $G = (R, L, I)$ in which $R = \{r_1, r_2, ..., r_n\}$ is a set of resources — identified by their URI —, $L = \{l_1, l_2, ..., l_n\}$ is a set of typed links — identified by their URI — and $I = \{i_1, i_2, ..., i_n\}$ is a set of instances of these links between resources, such as $i_i = \langle l_j, r_a, r_b \rangle$.*

In addition, we defined different functions that identify if and how two resources (represented by their URI, and following the Linked Data principles) are connected in such graphs.

**Definition 2.** *$C_d$ is a function that computes the number of* direct and distinct *links between resources in a graph $G$. $C_d(l_i, r_a, r_b)$ equals 1 if there is an instance of $l_i$ from resource $r_a$ to resource $r_b$, 0 if not. By extension $C_d$ can be used to compute (1) the total number of direct and distinct links from $r_a$ to $r_b$ ($C_d(n, r_a, r_b)$) as well as (2) the total number of distinct instances of the link $l_i$ from $r_a$ to any node ($C_d(l_i, r_a, n)$).*

* The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Líon-2).

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 411–415, 2010.

**Definition 3.** *$C_{io}$ and $C_{ii}$ are functions that compute the number of* indirect and distinct *links, both* outcoming and incoming*, between resources in a graph $G$. $C_{io}(l_i, r_a, r_b)$ equals 1 if there is a resource $n$ that satisfy both $\langle l_i, r_a, n\rangle$ and $\langle l_i, r_b, n\rangle$, 0 if not. $C_{ii}(l_i, r_a, r_b)$ equals 1 if there is a resource $n$ that satisfy both $\langle l_i, n, r_a\rangle$ and $\langle l_i, n, r_b\rangle$, 0 if not. By extension $C_{io}$ and $C_{ii}$ can be used to compute (1) the total number of indirect and distinct links between $r_a$ and $r_b$ ($C_{io}(n, r_a, r_b)$ and $C_{ii}(n, r_a, r_b)$, respectively outcoming and incoming) as well as (2) the total number of resources $n$ linked indirectly to $r_a$ via $l_i$ ($C_{io}(l_i, r_a, n)$ and $C_{ii}(l_i, r_a, n)$, respectively outcoming and incoming)*

Based on these definitions, we defined different formula for computing *Linked Data Semantic Distance*[1] [3], *i.e.* the distance that exists between two resources within a Linked Data dataset[2]. These formula were defined using different criteria, using both direct and indirect relationships ($C_d$, $C_{io}$, $C_{ii}$), as well as using weights to give more importance to the links that are used only a few times in the dataset. Based on these different measures and user interviews, we decided to concentrate on a particular one, that we will simply name $LDSD$, and that uses both direct and indirect links as well as the aforementioned weights. This measure is defined as follows (Fig. 1).

$$LDSD(r_a, r_b) = \frac{1}{1 + \sum_i \frac{C_d(l_i,r_a,r_b)}{1+log(C_d(l_i,r_a,n))} + \sum_i \frac{C_d(l_i,r_b,r_a)}{1+log(C_d(l_i,r_b,n))} + \sum_i \frac{C_{ii}(l_i,r_a,r_b)}{1+log(C_{ii}(l_i,r_a,n))} + \sum_i \frac{C_{io}(l_i,r_a,r_b)}{1+log(C_{io}(l_i,r_a,n))}}$$

**Fig. 1.** The $LDSD$ measure

### 1.2 The LDSD Ontology

Moreover, in order to represent the distances computed using the aforementioned $LDSD$ measure in a machine-readable way (so that they can be queried and reused by third-party applications), we designed a related ontology, available at `http://dbrec.net/ldsd/ns`. This ontology has two main goals. On the one hand, it aims at modelling the distance between resources, so that they can be queried using SPARQL (and in some cases with its `FILTER` clause to limit resources to the ones at a particular distance from the seed one). On the other hand, its goal is to store some information about the way the distance has been computed, so that these distances can be explained, as we shall see in the upcoming section.

Using this ontology, the fact that Elvis Presley is at a distance of 0.09 from Johnny Cash, and that such distance can be explained because (among other) both share the value `http://dbpedia.org/class/yago/SunRecordsArtists` for their `rdf:type` property, which is shared by only 19 artists in the original dataset, can be represented as follows (Table 1).

[1] Note that we use the term *distance* while some of these measures are not symmetric.
[2] We shall note that theoretically, nothing prevents this dataset to be distributed as *e.g.* the Linking Open Data cloud.

**Table 1.** Excerpt of the distance explanation between Johnny Cash and Elvis Presley, using the *LDSD* ontology

```
@prefix ldsd: <http://dbrec.net/ldsd/ns#> .
<http://dbrec.net/distance/774a32aa-dede-11de-84a3
    -0011251e3563> a ldsd:Distance ;
   ldsd:from <http://dbpedia.org/resource/Johnny_Cash> ;
   ldsd:to <http://dbpedia.org/resource/Elvis_Presley> ;
   ldsd:value "0.0977874534544" .
<http://dbrec.net/distance/774a32aa-dede-11de-84a3
    -0011251e3563> ldsd:explain [
  a ldsd:IndirectOut ;
  ldsd:property <http://www.w3.org/1999/02/22-rdf-syntax-
      ns#type> ;
  ldsd:node <http://dbpedia.org/class/yago/
      SunRecordsArtists> ;
  ldsd:total "19"
] .
```

In addition, the *LDSD* ontology features mappings with the MuSim ontology[3] [2]. Additional mappings with SCOVO[4] — the Statistical COre VOcabulary — might be provided in the future (since the number of instance sharing a particular link can be considered as statistical data).

## 2 dbrec: Music Recommendations Using *LDSD*

Based on the previous findings, we implemented dbrec — `http://dbrec.net` —, a system that demonstrates how *LDSD* can be used in the realm of recommender systems. In particular, dbrec has been build by computing *LDSD* for all artists and bands referenced in DBpedia. While it does not involve cross-datasets

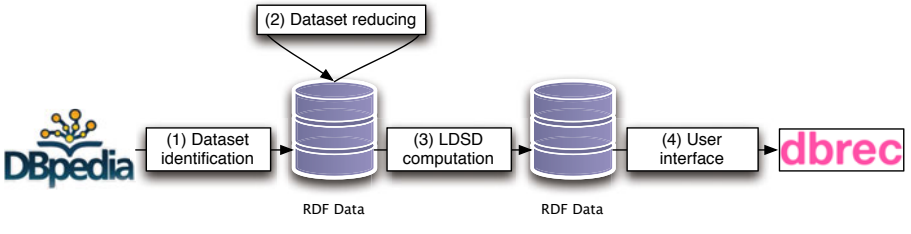



**Fig. 2.** Architecture of the dbrec framework

[3] `http://grasstunes.net/ontology/musim/musim.html`
[4] `http://sw.joanneum.at/scovo/schema.html`

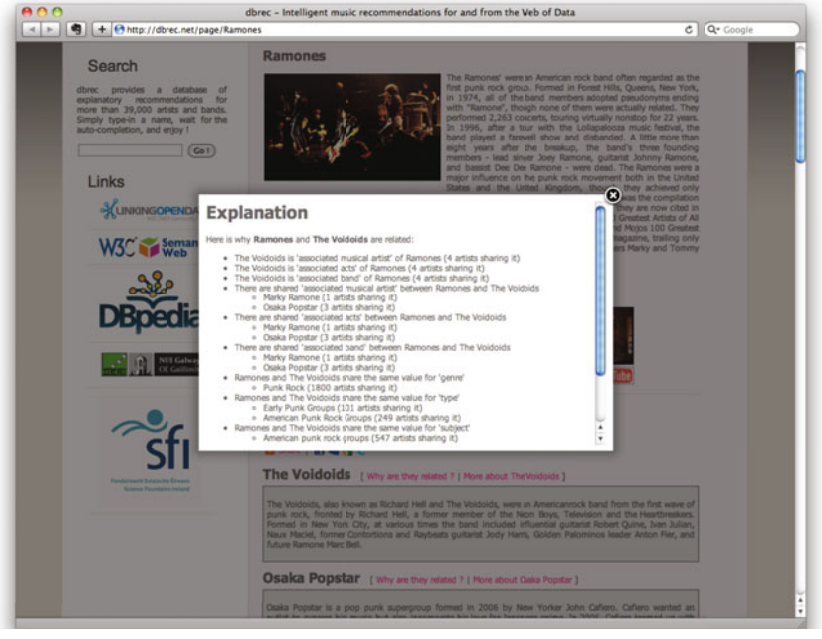



**Fig. 3.** Recommendations and explanations in dbrec

recommendations, it offers two major advantages: (i) there are more than 39,000 artists available in DBpedia for which recommendations can be build, and (ii) DBpedia also provides pictures and descriptions for most of them, that can be used when building the related user interface, as we will see in the next section. The system was implemented using the following steps (Fig. 2): (1) identify the relevant dataset from DBpedia; (2) reduce the dataset for query optimisation; (3) compute the distances using the *LDSD* algorithm and represent the results using the *LDSD* ontology; (4) build a user interface for browsing recommendations.

## 3 Purpose of the Demonstration

The purpose of the demonstration is to give a comprehensive overview of the dbrec system in use.

First, attendees will be able to check recommendations for the artists and bands of their choice. The search interface of dbrec provides auto-completion capabilities so that users simply start typing the name of a band, and the system suggests names for which recommendations are available in the system. Once validated, users have access to a live view of the recommendations, built using SPARQL queries applied to the RDF data resulting from the *LDSD* algorithm (this data being modelled with the aforementioned *LDSD* ontology). As we previously mentioned, dbrec relies on DBpedia to provide picture and description

of each artist and band, hence enhancing the user experience when browsing recommendations. It also displays related YouTube videos about the main artist and provide links to share the recommendation on various Web 2.0 services such as Twitter and Facebook.

In addition, attendees will experience the explanation feature provided by dbrec, so that they can learn why the recommendations have been made. Indeed, when browsing recommendations, a pop-up providing such explanations can be displayed for each recommended artist or band. The explanations are provided using the information recorded about each measure, using the *LDSD* ontology as we previously detailed. By using this feature, attendees will discover why dbrec recommends $X$ if they are looking for information about $Y$.

The following figure (Fig. 3) displays the recommendation page for the Ramones, as well as a pop-up explaining why the Voidoids are recommended in that case[5].

In addition, we will show how these recommendations, by being available as Linked Data (*via* RDFa markup within the pages) can be browsed independently, and how they could be combined with other data for navigation and querying purposes.

## References

1. Berners-Lee, T.: Linked Data. In: Design Issues for the World Wide Web, World Wide Web Consortium (2006), http://www.w3.org/DesignIssues/LinkedData.html
2. Jacobson, K., Raimond, Y., Sandler, M.: An Ecosystem for Transparent Music Similarity in an Open World. In: International Symposium on Music Information Retrieval (2009)
3. Passant, A.: Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations. In: Linked AI: AAAI Spring Symposium Linked Data Meets Artificial Intelligence. AIII (2010)
4. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man and Cybernetics 19, 17–30 (1989)

[5] Live view at http://dbrec.net/page/Ramones

# PossDL — A Possibilistic DL Reasoner for Uncertainty Reasoning and Inconsistency Handling

Guilin Qi[1], Qiu Ji[1], Jeff Z. Pan[2], and Jianfeng Du[3]

[1] School of Computer Science and Engineering, Southeast University, Nanjing 210096
[2] Institute AIFB, University of Karlsruhe, Karlsruhe, Germany
[3] Department of Computing Science, University of Aberdeen, Aberdeen, UK
[4] Guangdong University of Foreign Studies, Guangzhou 510006, China

**Abstract.** Uncertainty reasoning and inconsistency handling are two important problems that often occur in the applications of the Semantic Web. Possibilistic description logics provide a flexible framework for representing and reasoning with ontologies where uncertain and/or inconsistent information exists. Based on our previous work, we develop a possibilistic description logic reasoner. Our demo will illustrate functionalities of our reasoner for various reasoning tasks that possibilistic description logics can provide.

## 1 Introduction

Uncertainty reasoning and inconsistency handling are two important problems that often occur in the applications of the Semantic Web, such as the areas like medicine and biology [4]. Recently, there is an increasing interest to extend Web Ontology Language OWL to represent uncertain knowledge. Most of the work is based on Description Logics (DL) that provide important formalisms for representing and reasoning with ontologies. A DL knowledge base is then extended by attaching each axiom in it with a degree of belief. The degree of belief can have several meanings depending on the semantics of the logic. For example, in probabilistic description logics, the degree of belief can be explained as degree of overlap between two concepts and in possibilistic description logics [4], the degree of belief is explained as the necessity degree or certainty degree (see [7]). Inconsistency handling in DL is another problem that has attracted a lot of attention. Inconsistency can occur due to several reasons, such as modeling errors, migration or merging ontologies, and ontology evolution. When an ontology is inconsistent, an ontology language which has first-order features, such as a description logic, cannot be applied to infer non-trivial conclusions.

Let us consider a medical ontology modified from an example given in [4].

*Example 1.* Given an ontology $\mathcal{B}$ consisting of the following terminological axioms attached with confidence degrees:

$$ax_1 : (Heartpatient \sqsubseteq HighBloodPressure, 1)$$
$$ax_2 : (PacemakerPatient \sqsubseteq \neg HighBloodPressure, 1)$$
$$ax_3 : (HeartPatient \sqsubseteq MalePacemakerPatient, 0.4)$$
$$ax_4 : (HeartPatient \sqsubseteq \exists HasHealthInsurance.PrivateHealth, 0.9)$$
$$ax_5 : (PacemakerPatient(Tom), 0.8).$$

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 416–420, 2010.

Suppose we use possibilistic logic, then $ax_1$ means that "it is absolute certain that heart patients suffers from high blood pressure", $ax_2$ can be explained similarly, $ax_3$ says that "it is a little certain that heart patient are male pacemaker patient", $ax_4$ says "it is highly certain that heart patients have a private insurance", and finally $ax_5$ states that "it is quite certain that Tom is a pacemaker patient". Suppose we learn that Tom is a heart patient with degree 0.5 ($ax_6$: ($HeartPatient(Tom)$,0.5)), i.e., it is somewhat certain that Tom is a heart patient, and we add this axiom to the ontology, then the ontology will become inconsistent. From this updated ontology, we may want to query if Tom suffers from high blood pressure and to ask to what degree we can infer this conclusion?

Possibilistic description logics, first proposed by Hollunder in [2] and further developed by Qi and Pan in [7], are extensions of description logics with possibilistic semantics. It is well-known that possibilistic logic is a powerful logical framework for dealing with uncertainty and handling inconsistency. Possibilistic description logics inherit these two nice properties and have very promising applications in the Semantic Web. A possibilistic DL knowledge base consists of a set of weighted axioms of the form $(\phi, \alpha)$, where $\phi$ is a DL axiom such as an assertional axiom of the form $C(a)$ and $\alpha$ is an element of the semi-open real interval (0,1] or of a finite total ordered scale. A weighted axiom $(\phi, \alpha)$ encodes the constraint $N(\phi) \geq \alpha$, where $N$ is a necessity measure [1], with the intended meaning that the necessity degree of $\phi$ is at least $\alpha$.

In our previous work [7], we have provided syntax and semantics of possibilistic description logics and defined several inference services. We have also provided algorithms for implementing these inference services. Based on these algorithms, in this work, we develop a possibilistic description logic reasoner by using OWL API 3.0.0.v1310 and Pellet v2.0.0[1]. Our demo will illustrate functionalities of our reasoner for various reasoning tasks that possibilistic description logics can provide.

## 2 The PossDL Reasoner

### 2.1 Possibilistic Description Logics

We introduce the syntax of possibilistic DLs and some reasoning tasks, and refer to [7] for the semantics of possibilistic DLs. A *possibilistic axiom* is a pair $(\phi, \alpha)$ consisting of an axiom $\phi$ and a weight $\alpha \in (0, 1]$ denoting the confidence degree of $\phi$, which will be interpreted as the necessity degree of $\phi$. A *possibilistic TBox* (resp., *ABox*) is a finite set of possibilistic axioms $(\phi, \alpha)$, where $\phi$ is an TBox (resp., ABox) axoim. A possibilistic DL knowledge base $\mathcal{B} = (\mathcal{T}, \mathcal{A})$ consists of a possibilistic TBox $\mathcal{T}$ and a possibilistic ABox $\mathcal{A}$. Let $\mathcal{B}_\alpha = \{\phi_i | (\phi_i, \alpha_i) \in \mathcal{B}, \alpha_i \geq \alpha\}$. An important reasoning task in possibilistic DLs is to compute the inconsistency degree of a possibilistic DL knowledge base $\mathcal{B}$, denoted by $Inc(\mathcal{B})$, which is defined as $Inc(\mathcal{B}) = max\{\alpha_i | \ \mathcal{B}_{\geq}$ is inconsistent$\}$. Consider Example 1, suppose $\mathcal{B} = \{ax_1, ..., ax_6\}$, then $Inc(\mathcal{B}) = 0.5$.

There are three inference services in possibilistic DLs.

- A DL axiom $\phi$ is a *plausible* consequence of a possibilistic DL knowledge base $\mathcal{B}$, written $\mathcal{B} \models_P \phi$ if $\mathcal{B}_{>Inc(\mathcal{B})} \models \phi$.

[1] http://clarkparsia.com/pellet/

- A DL axiom $\phi$ is a *possibilistic* consequence of $\mathcal{B}$ to degree $\alpha$, written $\mathcal{B} \models_\pi \phi$, if the following conditions hold: (1) $\mathcal{B}_{\geq\alpha}$ is consistent, (2) $\mathcal{B}_{\geq\alpha} \models \phi$, (3) for all $\beta{>}\alpha$, $\mathcal{B}_{\geq\beta} \not\models \phi$.
- A possibilistic DL axiom $(\phi, \alpha)$ is a consequence from $\mathcal{B}$, written $\mathcal{B} \models (C(a), \alpha)$, if $\alpha > Inc(\mathcal{B})$ and $\mathcal{B}_{\geq\alpha} \models \phi$.

Note that the possibilistic consequence relation $\models_\pi$ is different from the consequence relation $\models$ because the former is to check to what degree an assertion holds whilst the latter is to check if a possibilistic assertion holds. Consider Example 1 again, we have $\mathcal{B} \models_P \neg HighBloodPressure(Tom)$, $\mathcal{B} \models_\pi (\neg HighBloodPressure(Tom), 0.8)$, and $\mathcal{B} \models (Heartpatient \sqsubseteq HighBloodPressure, 1)$. However, $HeartPatient \sqsubseteq MalePacemakerPatient$ cannot be inferred from $\mathcal{B}$ by possibilistic inference due to the notorious *drowning effect*, i.e., all axioms whose degrees are less than or equal to $Inc(\mathcal{B})$ are blocked to be used in the inference. Therefore, we also implement a drowning-free variant of possibilistic inference, called linear order inference. Let $\mathcal{B} = \{(\phi_i, \alpha_i) : i = 1, ..., n\}$ be a possibilistic DL knowledge base. Suppose $\beta_j$ $(j = 1, ..., k)$ are all distinct necessity degrees appearing in $\mathcal{B}$ such that $\beta_1 > \beta_2 > ... > \beta_k$. Let $\Sigma_\mathcal{B} = (S_1, ..., S_k)$, where $S_i = \{\phi_l : (\phi_l, \alpha_l) {\in} \mathcal{B}, \alpha_l = \beta_i\}$, and $\Sigma_{LO,\mathcal{B}} = \bigcup_{i=1}^{k} S_i^{'}$, where $S_i^{'}$ is defined by $S_i^{'} = S_i$ if $S_i \cup \bigcup_{j=1}^{i-1} S_j^{'}$ is consistent, $\emptyset$ otherwise. A DL axiom $\phi$ is said to be a linear consequence of $\mathcal{B}$, denoted by $\mathcal{B} \models_{LO} \phi$, if and only if $\Sigma_{LO,\mathcal{B}} \models \phi$. In Example 1, we have $\mathcal{B} \models_{LO} HeartPatient \sqsubseteq MalePacemakerPatient$.

### 2.2 The PossDL Reasoner

We have developed a tool, called PossDL, as a plug-in in NeOn Toolkit v2.3[2] which is a multi-platform ontology engineering environment. The PossDL reasoner provides the functionalities of computing inconsistency degree and doing instance / subsumption checking with necessity degree for a possibilistic DL knowledge base. It takes an OWL ontology and a separate file storing the necessity degrees for the axioms in the ontology as inputs. PossDL mainly consists of four parts, described as follows.

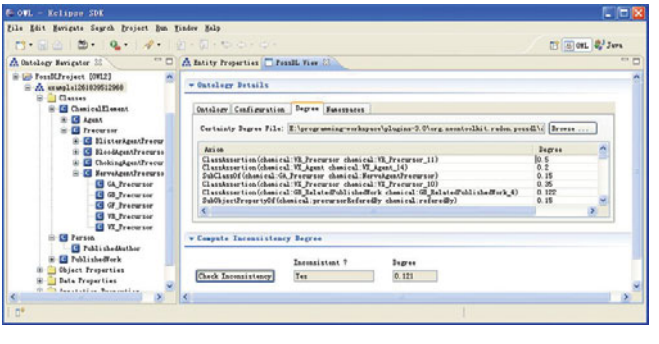


**Fig. 1.** The user interface to compute inconsistency degree

[2] http://neon-toolkit.org/wiki/Main_Page

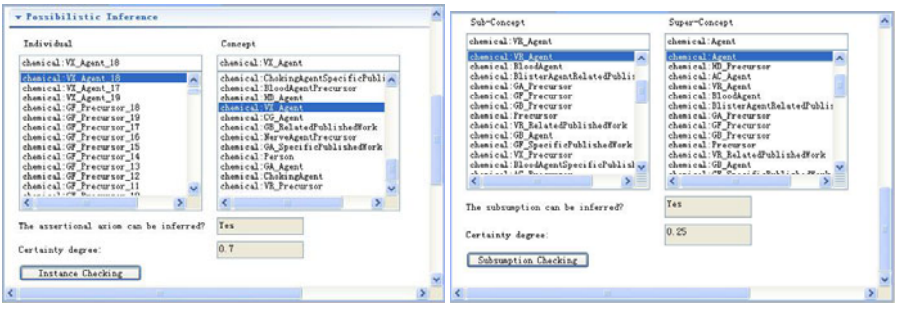



**Fig. 2.** The user interfaces for instance / subsumption checking using possibilistic inference

In the first part (see "Ontology Details" in Figure 1), we show the information about the chosen ontology $O$ (see "Ontology" tab), the necessity degrees for the axioms in $O$ (see "Degree" tab), the namespaces appearing in $O$ (see "Namespaces" tab) and the configuration (see "Configuration" tab). In the "Configuration" tab, we provide three options about the reasoning tasks: compute inconsistency degree, possibilistic inference and linear order inference. If one of them is chosen, the corresponding part or section will be shown automatically. By default, the first option (i.e. compute inconsistency degree) is chosen. It is noted that, the necessity degrees are stored in a separate file. This file is still in OWL format. Specifically, we take a string of an axiom as a concept ID and associate a necessity degree to this concept by an annotation property.

In the part of "Compute Inconsistency Degree" (see "Compute Inconsistency Degree" in Figure 1), we can compute the inconsistency degree by clicking the button. If this possiblisitic DL knowledge base is inconsistent, "yes" will be shown in the text area below the sentence of "Inconsistency?" and the inconsistency degree can be seen in the text area right below the word of "Degree". Otherwise, "no" and "–" will be shown in two text areas respectively.

In the part of "Possibilistic Inference" (see the left part in Figure 2), we can choose an individual and a concept to do instance checking with necessity degree based on the possibilistic inference by clicking the button of "Instance Checking". The result about whether the corresponding assertional axiom can be inferred will be shown in the first text area. If this axiom can be inferred, the necessity degree for this axiom will be computed and it can be seen in the second text area. Otherwise, "–" will be shown. It is similar to do subsumption checking in this part (see the right part in Figure 2). The part of "Linear Order Inference" is similar to the part of "Possibilistic Inference" but using different inference strategies.

## 3 Related Work

The relationship between possibilistic DLs and other uncertainty formalisms for DLs has been discussed in a survey paper [5]. One of the most important approaches that extend DLs with uncertainty reasoning are probabilistic DLs, such as the work presented in [4] which has a tool support [3]. Some major differences between possibilistic DLs and probabilistic DLs are given as follows. First, unlike probabilistic DLs, the confidence degree attached to an axiom in possibilistic DLs is not absolute and can be replaced by another

number as long as the ordering between two confidence degrees is not changed. Second, in possibilistic DLs, a necessity degree is attached to a DL axiom. Whist in probabilistic DLs in [4], an interval $[l, u]$ ($l, u \in [0, 1]$) is attached to a *conditional constraint* $(D|C)$, where $C$ and $D$ are DL concepts, which cannot be expressed by means of DL axioms.

Fuzzy DLs can be used to deal with uncertainty or vagueness in DLs (e.g., [9,8]) with scalable tool support [6]. The main difference between possibilistic DLs and fuzzy DLs is that, the truth value of a concept (or a role) in possibilistic DLs is still two-valued, whilst in fuzzy DLs, the truth value of a concept (or a role) is multi-valued. So the semantics of possibilistic DLs is different from that of fuzzy DLs.

## 4 What Will Be Demonstrated?

In our demonstration, we present our PossDL reasoner, which is an extension of Pellet for uncertainty reasoning and inconsistency handling. In particular, PossDL supports three different possibibilistic inference services and the linear order inference which is used to deal with the drowning effect. In the demonstration, we will illustrate our reasoner with practical examples obtained from ontology learning and ontology matching. Besides, for various needs of the users different algorithms to deal with inconsistency and/or uncertainty will be demonstrated.

## Acknowledgments

Guilin Qi is partially supported by Excellent Youth Scholars Program of Southeast University under grant 4009001011.

## References

1. Dubois, D., Lang, J., Prade, H.: Possibilistic logic. In: Handbook of Logic in Aritificial Intelligence and Logic Programming, pp. 439–513 (1994)
2. Hollunder, B.: An alternative proof method for possibilistic logic and its application to terminological logics. In: Proc. of UAI 1994, pp. 327–335 (1994)
3. Klinov, P., Parsia, B.: Optimization and evaluation of reasoning in probabilistic description logic: Towards a systematic approach. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 213–228. Springer, Heidelberg (2008)
4. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. 172(6-7), 852–883 (2008)
5. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics 6(4), 291–308 (2008)
6. Pan, J.Z., Stamou, G.B., Stoilos, G., Taylor, S., Thomas, E.: Scalable querying services over fuzzy ontologies. In: Proc. of WWW 2008, pp. 575–584 (2008)
7. Qi, G., Pan, J.Z., Ji, Q.: Extending description logics with uncertainty reasoning in possibilistic logic. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 828–839. Springer, Heidelberg (2007)
8. Stoilos, G., Stamou, G., Pan, J.Z., Tzouvaras, V., Horrocks, I.: Reasoning with very expressive fuzzy description logics. J. Artif. Intell. Res. 30, 273–320 (2007)
9. Straccia, U.: Reasoning within fuzzy description logics. J. Artif. Intell. Res. 14, 137–166 (2001)

# PoolParty: SKOS Thesaurus Management Utilizing Linked Data

Thomas Schandl and Andreas Blumauer

punkt. NetServices GmbH,
Lerchenfelder Gürtel 43, 1160 Vienna, Austria
schandl@punkt.at, blumauer@punkt.at

**Abstract.** Building and maintaining thesauri are complex and laborious tasks. PoolParty is a Thesaurus Management Tool (TMT) for the Semantic Web, which aims to support the creation and maintenance of thesauri by utilizing Linked Open Data (LOD), text-analysis and easy-to-use GUIs, so thesauri can be managed and utilized by domain experts without needing knowledge about the semantic web. Some aspects of thesaurus management, like the editing of labels, can be done via a wiki-style interface, allowing for lowest possible access barriers to contribution. PoolParty can analyse documents in order to glean new concepts for a thesaurus. Additionally a thesaurus can be enriched by retrieving relevant information from Linked Data sources and thesauri can be imported and updated via LOD URIs from external systems and also can be published as new linked data sources on the semantic web.

**Keywords: S**emantic Web, Linking Open Data, Thesaurus, SKOS, RDF, Wiki.

## 1 Introduction

Thesauri have been an important tool in Information Retrieval for decades and still are [1]. While they have the potential to greatly improve the information management of large organisations, professionally managed thesauri are rarely used in content management systems, search engines or tagging systems.

Important reasons frequently given for this are: (1) the difficulty of learning and using TMT, (2) the lacking possibilities to integrate TMTs into existing enterprise information systems, (3) it's laborious to create and maintain a thesaurus, and while TMTs often support either automatic or manual methods to maintain a thesaurus they rarely combine those two approaches, and (4) companies don't have enough knowledge about thesaurus building methodologies and/or valuable use cases utilizing semantic knowledge models like SKOS thesauri.

The TMT PoolParty[1] addresses the first three issues. The demo will show PoolParty's thesaurus management features including document analysis, its Linked Data capabilities and its Wiki interface.

[1] http://poolparty.punkt.at/PoolParty/ - A screencast is available at http://bit.ly/6OqhYZ

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 421–425, 2010.

## 2 Use Cases

PoolParty is a tool to create and maintain multilingual SKOS (Simple Knowledge Organisation System)[2] thesauri, aiming to be easy to use for people without a Semantic Web background or special technical skills. Utilizing semantic web technologies like RDF and especially SKOS allow thesauri to be represented in a standardised manner [2]. While OWL would offer greater possibilities in creating knowledge models, it is deemed too complex for the average information worker.

PoolParty was conceived to facilitate various commercial applications for thesauri. In order to achieve this, it needs to publish them and offer methods of integrating them with various applications [3]. In PoolParty this can be realized on top of its RESTful web service interface providing thesaurus management, indexing, search, tagging and linguistic analysis services.

Some of these (semantic) web applications are:

- Semantic search engines
- Recommender systems (similarity search)
- Corporate bookmarking
- Annotation- & tag recommender systems
- Autocomplete services and facetted browsing.

These use cases can be either achieved by using PoolParty stand-alone or by integrating it with existing Enterprise Search Engines and Document Management Systems.

## 3 Technologies

PoolParty is written in Java and uses the SAIL API[3], whereby it can be utilized with various triple stores, which allows for flexibility in terms of performance and scalability.

Thesaurus management itself (viewing, creating and editing SKOS concepts and their relationships) can be done in an AJAX Frontend based on Yahoo User Interface (YUI). Editing of labels can alternatively be done in a Wiki style HTML frontend.

For key-phrase extraction from documents PoolParty uses a modified version of the KEA[4] 5 API, which is extended for the use of controlled vocabularies stored in a SAIL Repository (this module is available under GNU GPL). The analysed documents are locally stored and indexed in Lucene[5] along with extracted concepts and related concepts.

## 4 Thesaurus Management with PoolParty

The main thesaurus management GUI of PoolParty (see Fig. 1) is entirely web-based and utilizes AJAX to e.g. enable the quick merging of two concepts either via drag & drop or autocompletion of concept labels by the user. An overview over the thesaurus can be gained with a tree or a graph view of the concepts.

[2] http://www.w3.org/2004/02/skos
[3] http://www.openrdf.org/doc/sesame2/system/ch05.html
[4] http://www.nzdl.org/Kea/index.html
[5] http://lucene.apache.org/

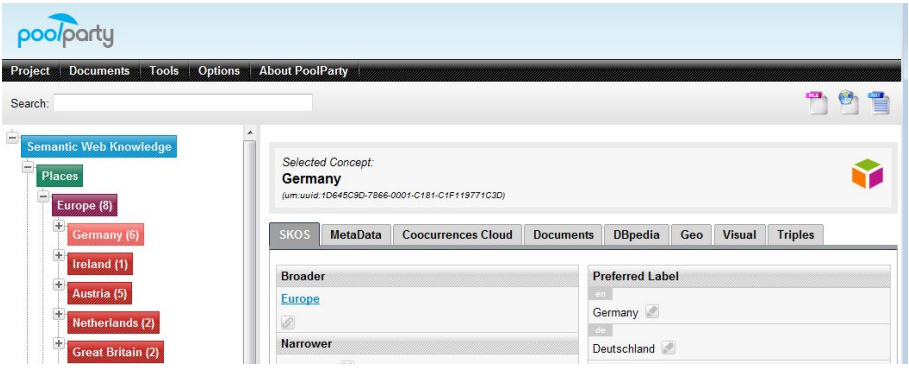



**Fig. 1.** PoolParty's main GUI with concept tree and SKOS view of selected concept

Consistent with PoolParty's goal of relieving the user of burdensome tasks while managing thesauri doesn't end with a comfortable user interface: PoolParty helps to semi-automatically expand a thesaurus as the user can use it to analyse documents (e.g. web pages or PDF files) relevant to her domain in order to glean candidate terms for her thesaurus. This is done by a key-phrase extractor which is based on KEA. The extractor not only detects concepts within a text which are already part of the thesaurus but also new canditate terms ("free concepts"). These can be approved by a thesaurus manager and can be integrated into the thesaurus, turning them into "approved concepts".

Documents can be searched in various ways – either by keyword search in the full text, by searching for their tags or by semantic search. The latter takes not only a concept's preferred label into account, but also its synonyms and the labels of its related concepts are considered in the search. The user might manually remove query terms used in semantic search. Boost values for the various relations considered in semantic search may also be adjusted. In the same way the recommendation mechanism for document similarity calculation works.

PoolParty by default also publishes an HTML Wiki version of its thesauri, which provides an alternative way to browse and edit concepts. Through this feature anyone can get read access to a thesaurus, and optionally also edit, add or delete labels of concepts. Search and autocomplete functions are available here as well.

The Wiki's HTML source is also enriched with RDFa, thereby exposing all RDF metadata associated with a concept as linked data which can be picked up by RDF search engines and crawlers.

PoolParty supports the import of thesauri in SKOS (in serializations including RDF/XML, N-Triples or Turtle) or Zthes format.

## 5 Linked Open Data Capabilities

PoolParty not only publishes its thesauri as Linked Open Data (additionally to a SPARQL endpoint)[6], but it also consumes LOD in order to expand thesauri with

[6] Example thesaurus published as Wiki with embedded RDFa to expose linked data: http://bit.ly/aM7LSL

information from LOD sources. Concepts in the thesaurus can be linked to e.g. DBpedia[7] via the DBpedia lookup service [4], which takes the label of a concept and returns possible matching candidates. The user can select the DBpedia resource that matches the concept from his thesaurus, thereby creating a owl:sameAs relation between the concept URI in PoolParty and the DBpedia URI. The same approach can be used to link to other SKOS thesauri available as Linked Data.

Other triples can also be retrieved from the target data source, e.g. the DBpedia abstract can become a skos:definition and geographical coordinates can be imported and be used to display the location of a concept on the map, where appropriate. The DBpedia category information may also be used to retrieve additional concepts of that category as siblings of the concept in focus, in order to populate the thesaurus.

PoolParty is not only capable of importing a SKOS thesaurus from a Linked Data server, it may also receive updates to thesauri imported this way. This feature has been implemented in the course of the KiWi[8] project funded by the European Commission. KiWi also contains SKOS thesauri and exposes them as LOD. Both systems can read a thesaurus via the other's LOD interfaces and may write it to their own store. This is facilitated by special Linked Data URIs that return e.g. all the top-concepts of a thesaurus, with pointers to the URIs of their narrower concepts, which allow other systems to retrieve a complete thesaurus through iterative dereferencing of concept URIs.

Additionally KiWi and PoolParty publish lists of concepts created, modified, merged or deleted within user specified time-frames. With this information the systems can learn about updates to one of their thesauri in an external system. They then can compare the versions of concepts in both stores and may write according updates to their own store.

Data transfer and communication are achieved using REST/HTTP, no other protocols or middleware are necessary. Also no rights management for each external systems is needed, which otherwise would have to be configured separately for each source.

## 6 System Demo

In the demonstration session visitors will learn how to manage a SKOS thesaurus and how PoolParty supports the user in the process of creating, editing, relating and merging of SKOS concepts using the web GUI, autocomplete and drag and drop. We will explore different views of concepts (tree, graph, triples and location on a map).

We'll take a tour of the Wiki interface and learn how to use it to edit labels and take a look at the RDFa output exposed in the Wiki.

The document analysis features will be presented, showing how new concepts can be gleaned from text and integrated into a thesaurus. The visitor will learn how to conduct a semantic search function as well as how the similarity recommendations for indexed documents tagged with concepts work.

It will be shown how to interlink local concepts from DBpedia, thereby enhancing one's thesaurus with triples from the LOD cloud. Finally the data synchronisation via

[7] http://dbpedia.org/
[8] http://kiwi-project.eu/

LOD will be shown by way of example interactions between the semantic framework KiWi and PoolParty.

## 7 Future Work

In the course of project LASSO funded by Austria's FFG[9] we will research improved methods of interlinking local thesauri with relevant entities from the LOD cloud. This will enable PoolParty to support thesaurus managers e.g. by semi-automatically populating a thesaurus by looking for related terms on external sites. Tighter interlinking with the LOD cloud can also enable PoolParty to suggest how new concepts could be classified, i.e. recommend possible parent concepts from a thesaurus for a concept in focus.

The synchronisation process via Linked Data will be improved in the ongoing KiWi project. We will implement an update and conflict resolution dialogue through which a user may decide which updates to concepts to accept and to consequently write to the system's store.

Most importantly we will work on integrating PoolParty with existing Enterprise Search Engines, Enterprise Wikis and Content Management Systems.

## References

1. Aitchison, J., Gilchrist, A., Bawden, D.: Thesaurus Construction and Use: A Practical Manual, 4th edn. Europa Publications (2000)
2. Pastor-Sanchez, J.P., Martínez Mendez, F., Rodríguez-Muñoz, J.V.: Advantages of thesaurus representation using the Simple Knowledge Organization System (SKOS) compared with proposed alternatives. Informationresarch 14(4) (December 2009), http://informationr.net/ir/14-4/paper422.html
3. Viljanen, K., Tuominen, J., Hyvönen, E.: Publishing and using ontologies as mashup services. In: Proceedings of the 4th Workshop on Scripting for the Semantic Web (SFSW 2008), 5th European Semantic Web Conference 2008 (ESWC 2008), Tenerife, Spain, June 1-5 (2008)
4. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 154–165 (2009)

[9] Austrian Research Promotion Agency - http://ffg.at/

# DSMW: Distributed Semantic MediaWiki

Hala Skaf-Molli, Gérôme Canals, and Pascal Molli

Université de Lorraine, Nancy, LORIA – INRIA Nancy-Grand Est, France
{skaf,canals,molli}@loria.fr

**Abstract.** DSMW is an extension to Semantic Mediawiki (SMW), it allows to create a network of SMW servers that share common semantic wiki pages. DSMW users can create communication channels between servers and use a publish-subscribe approach to manage the change propagation. DSMW synchronizes concurrent updates of shared semantic pages to ensure their consistency. It offers new collaboration modes to semantic wiki users and supports dataflow-oriented processes.

## 1 Research Background: Collaborative Editing

Semantic wikis allow to create and edit collaboratively semantically annotated documents. However, compared with other collaborative systems, semantic wikis do not support offline work or multi-synchronous editing [1]. In existing semantic wikis, every change in a page is immediately visible for both end users and semantic engines. However, in many cases it is necessary to change multiple pages before to make them visible. Existing semantic wikis cannot prevent users to access, navigate or query inconsistent pages. Moreover, the lack of multi-synchronous support prevents users to work isolated [2] and also prevents semantic wikis to support dataflow oriented processes.

To overcome these limitations, we propose a distributed approach for semantic wikis. In this approach, semantic wiki pages are replicated over a network of interconnected semantic wiki servers. Changes issued on one server are local but can be published to other servers. Remote servers can subscribe to these changes, pull them and integrate them to their local pages. Changes propagation remains under the control of the users. Concurrent changes on a page issued by different servers are handled by a merge procedure.

Using this approach, users can alternate between isolated periods of work and synchronization sequences with remote servers. They can introduce changes to multiple pages before to atomically render these changes public. They can choose when to incorporate, or not, remote changes. In addition, the approach can be the basis for implementing processes in which flows of semantic wiki pages can traverse a network of semantic wiki servers. Each wiki server can implement one or several steps of a particular process for the creation and maintenance of semantic pages.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 426–430, 2010.

## 2 DSMW Approach

DSMW is an extension to Semantic MediaWiki (SMW) [3]. It allows to create a network of SMW servers that share common semantic wiki pages. DSMW manages the propagation and the integration of changes issued on one SMW server to remote servers on the network. The system ensures the consistency of the whole set of replicated pages.

DSMW users can create and edit semantically annotated wiki pages as with a regular SMW server. Then she/he can manage pages changes as a software developer does with her/his source code using a distributed version control system: she/he can work in isolation while editing pages and semantic annotation on a single server, then she/he can publish part or all of her own changes by pushing them to DSMW public feeds, and she/he can subscribe to any remote public DSMW feeds, pull changes from remote servers and integrate them to the local pages.

The DSMW extension adds two main features to SMW: an optimistic replication algorithm, and an ontology to manage changes, publication and integration of changes sets.

*Page replication in DSMW* is handled by a dedicated replication procedure. Since semantic annotations are embedded in page content in SMW, DSMW replicates only page contents and there is no need to deal with annotations. DSMW uses the Logoot algorithm to synchronize concurrent changes[4]. Logoot guarantees the consistency of the shared pages based on the CCI model (Causality, Convergence, Intentions [5], the model used also by Google Wave). The propagation technic is publish-subscribe: changes issued one one server can be published by pushing them to one or several *pushFeeds*. Remote servers can subscribe to these feeds by connecting *pullFeeds* to existing remote pushFeeds. Then, they can pull changes and integrate them to the local pages. Concurrent changes are merged by the Logoot algorithm. Hereafter a brief description of the operations related to replication.

**Save:** the SMW save operation, called when a user saves edit modifications on a page, has been extended to build and log patches. Patches represent changes to a page as a sequence of elementary Insert and Delete operations. Patches are computed by the Logoot algorithm as a diff between the current and the new version of the page being saved. Logoot uses a special index to determine absolute insertion and deletion positions of the elements in a page. Once computed, a patch is applied to the local page and logged.

**CreatePushFeed:** creates a named communication channel to publish local changes. The content of the feed is specified by a semantic query. All pages in the query result belongs to that channel, meaning that changes on these pages will be published through that feed. Note that a page can belong to several channels.

**Push:** the push operation computes the set of patches for a given pushFeed to form a ChangeSet. A changeSet is the ordered set of all patches logged for all the pages belonging to that pushFeed. Once computed, the changeSet is added to the feed and can then be pulled by remote servers.

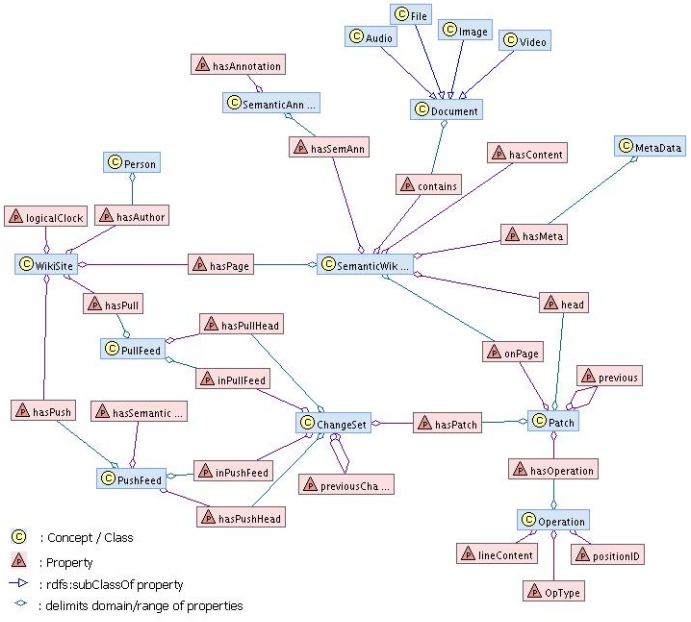



**Fig. 1.** Multi-synchronous ontology

**CreatePullFeed:** creates a named communication channel to pull remote changes. A pullFeed is connected to one single remote pushFeed, so a pull feed is defined by the URL of the remote server and the pushFeed name.

**Pull:** the push operation download all the pending changeSets published in the pushFeed connected to a given pullFeed. Patches extracted from the changeSets are then locally applied in the order they appear. To do so, Logoot uses the absolute positions computed during the patch creation to insert or delete page elements.

*The DSMW ontology* shown in the figure 1 represents all the concepts of DSMW: changes, change sets, push and pull feeds. This ontology makes possible the DSMW users to query the current state of the wiki and its complete history using SMW semantic queries. For instance, queries can extract *the list of unpublished changes* or *the list of published changes on a given channel*. This ontology is populated through the user interaction with the system: all the operations described in the previous paragraph create or delete instances of the DSMW ontology (see [6] for more details).

## 3 Use Cases and Applications

DSMW is used in ongoing French national projects: *WikiTaaable* and *CyWiki* *WikiTaable* is a distributed collaborative knowledge building systems for cooking recipes [7]. It integrates a case-based reasoning engine. WikiTaaable uses SMW

as a central module to manage all data and knowledge used in the system. DSMW supports the humans and machines collaboration by deploying several DSMW servers to implement continuous integration processes as those used during software development. WikiTaable is accessible at `http://taaable.fr`.

The *CyWiki* project uses DSMW as an infrastructure for the collaborative and assisted transformation of textual content into formal and structured knowledge. The transformation process is a decentralized process in which both human agents and automatic agents (text-mining agents, classification agents) collaborate to build knowledge units (in the form of ontology elements). This knowledge can then be used to query and make reasoning about the content. The experimental and application domain of the project is education.

## 4 System Demonstration

The demonstration scenario will focus on three main use-cases:

*The knowledge aggregation* corresponds to the use of a DSMW server to aggregate and combine wiki pages and knowledge from multiple sources. This server subscribe to these sources by creating pull feeds connected to the public push feeds at each source. By combining these sources, the system can answer new queries that could not be evaluated on a single source. The demonstration example will be the following:

- a first DSMW server holds semantic wiki pages about hotels in a city. Hotels are described with various properties (rooms, prices ...) and their location in the city relatively to well-known places (e.g. the train station, the main square),
- a second DSMW server holds semantic wiki pages about touristic information in the city. It describes sites of interest and cultural events with various properties and their location in the city, relatively to well known places.
- a third DSMW server subscribes to the public push feeds of the two previous, and regularly pull them. It then holds semantic wiki pages on both hotels and touristic information and their location in the city, and maintain these pages consistent with the original sources. This server can answer queries that cannot be evaluated on the original sources, typically to find an hotel close to a particular site of interest.

*The knowledge validation steps* corresponds to the use of one or several DSMW server to implement a validation process: prior to rendering public a set of semantic wiki pages, it can be desirable in some cases to validate their content by users or by running non-regression tests. The scenario will be based on the same hotel-tourist example. It consists in adding a fourth DSMW server that will serve as a public front-end for querying the hotel-tourist knowledge base. The aggregation server will then serve to combine the original sources and validate the new knowledge base. This validation step is done by users verifying the semantic annotations and eventually modify them and running tests by evaluating a fixed set of queries whose results are known and should not change.

Once validated, changes are propagated to the fourth server and are thus accessible to end-users. This validation step ensures the consistency and the stability of the final knowledge base. Any change to the original sources is tested, verified and eventually fixed before to be queried by end-users.

*The network construction use case* corresponds to the construction of a network of interconnected DSMW server. The demonstration will show how the push and pull feeds are created on the different servers of the hotel-tourist example, and connected to create the network.

## 5 Conclusion

In this demonstration we have presented a new collaborative tool, called DSMW, to support multi-synchronous collaboration and dataflow processes over semantic wiki pages. DSMW is developed as an extension of SMW. The first public release release of DSMW was published in October 2009. A new release DSMW 0.5 is available at `http://dsmw.org`. We continue the development and the research on DSMW. Research concerns divergence awareness in DSMW and the analysis of the social networks built by the collaborative editing.

## References

1. Dourish, P.: The parting of the ways: Divergence, data management and collaborative work. In: 4th European Conference on Computer Supported Cooperative Work (1995)
2. Molli, P., Skaf-Molli, H., Bouthier, C.: State Treemap: an Awareness Widget for Multi-Synchronous Groupware. In: Seventh International Workshop on Groupware - CRIWG. IEEE Computer Society, Los Alamitos (2001)
3. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Journal of Web Semantic 5(4), 251–261 (2007)
4. Weiss, S., Urso, P., Molli, P.: Logoot: a scalable optimistic replication algorithm for collaborative editing on p2p networks. In: International Conference on Distributed Computing Systems (ICDCS). IEEE, Los Alamitos (2009)
5. Sun, C., Jia, X., Zhang, Y., Yang, Y., Chen, D.: Achieving Convergence, Causality Preservation, and Intention Preservation in Real-Time Cooperative Editing Systems. ACM Transactions on Computer-Human Interaction 5(1) (1998)
6. Rahhal, C., Skaf-Molli, H., Molli, P., Weiss, S.: Multi-synchronous collaborative semantic wikis. In: Vossen, G., Long, D.D.E., Yu, J.X. (eds.) WISE 2009. LNCS, vol. 5802, pp. 115–129. Springer, Heidelberg (2009)
7. 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: SemWiki 2009 - 4rd Semantic Wiki Workshop at the 6th European Semantic Web Conference - ESWC 2009, Grèce Heraklion, May 16 (2009)

# TrOWL: Tractable OWL 2 Reasoning Infrastructure

Edward Thomas, Jeff Z. Pan, and Yuan Ren

Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK

**Abstract.** The Semantic Web movement has led to the publication of thousands of ontologies online. These ontologies present and mediate information and knowledge on the Semantic Web. Tools exist to reason over these ontologies and to answer queries over them, but there are no large scale infrastructures for storing, reasoning, and querying ontologies on a scale that would be useful for a large enterprise or research institution. We present the TrOWL infrastructure for transforming, reasoning, and querying OWL2 ontologies which uses novel techniques such as Quality Guaranteed Approximations and Forgetting to achieve this goal.

## 1 Introduction

Ontologies play a key role in the Semantic Web [3], where the W3C recommendation OWL [9] and its successor OWL2 [6] have become the de facto standards for publishing and sharing ontologies online. Increasingly these ontologies are being used by a variety of organisations, covering the definitions of a very wide range of subjects. While the number and variety of ontologies increases, the question of how to use these ontologies at an organisational level remains unresolved.

The reason why this is not a trivial problem is that OWL-DL language has a worst-case computational complexity of NExpTime, and 2NExpTime for OWL2-DL. This means that increasingly large ontologies may, in the worst case, require exponentially increasing computing resources to reason. Because of this, OWL2 also includes a number of tractable profiles which have combined complexity of PTIME-complete or better; however, these profiles all greatly restrict the expressive power of the language. As tool support for these profiles is still limited, it is also very easy for an ontology developer to accidentally exceed the complexity of their target profile by using a construct which is beyond the capability of that language fragment.

The approach of TrOWL is to offer support for all the expressive power of OWL2-DL, while maintaining tractability, by using language transformations. In particular, we utilise a Semantic Approximation [7] to transform OWL2-DL ontologies into OWL2-QL for conjunctive query answering, and a syntactic approximation from OWL2 to OWL2-EL for TBox reasoning. In addition, TrOWL contains a profile checker to detect which profile an ontology may already fit into, and it has support for heavyweight reasoning using a plug-in reasoner such as Fact++, Pellet, Hermit, or Racer.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 431–435, 2010.

## 2 Applications

The TrOWL reasoner was developed to support work on the MOST project[1] as well as provide reasoning support for large ontology-based knowledge bases.

**Validating Process Refinements.** During software development, processes are modelled in the standard language, Business Process Modelling Notation (BPMN); these process models are then refined to produce progressively more detailed models. Several metrics exist to validate these refinements as being consistent with the earlier models, but no tools are available which can validate these refinements automatically across multiple refinements on large models. Our approach has been to translate this process model into an ontology and use ontology reasoning services to validate the model. With this approach, we can validate that the refinements are valid, or highlight the processes which are causing the problem.

The process refinement case study generates ontologies with general concept inclusions (GCIs) of particular patterns. At the time of developing REL, mainstream reasoners such as Pellet and FaCT++ failed to efficiently provide complete classification results on the generated ontologies. Via the syntactic approximation of TrOWL, the GCIs in these ontologies can be efficiently resolved and the reasoning results can be proved complete.

**Software Engineering Guidance Ontology.** The physical device configuration case study uses ontologies to validate the consistency of the configuration of a network device. These devices are configured with several cards, and this configuration must be validated against a model which describes correct configurations.

The case study generates ontologies describing the configuration of network devices. These ontologies can sometimes be inconsistent, reflecting an invalid configuration of a physical device. To understand how this is manifested in the physical device and provide guidance on how it may be resolved, it is necessary to find justifications for the inconsistency, and isolate each axiom set which may be causing the inconsistency. Traditional tableaux reasoners usually terminate when an inconsistency is detected, making it difficult obtain all justifications. In this case, TrOWL can provide a more efficient and reliable service when used as a reasoning backend.

**Linked Open Data.** We have also investigated using TrOWL for linked open data repositories. We used the RDF-DL reasoning component in the Billion Triple Challenge in ISWC 2009. We managed to successfully load and reason over the billion triple RDF data set, with full RDFS reasoning over class and property subsumption. The benefit of using TrOWL for linked open data is that it supports reasoning in all profiles of OWL, as well as using RDF-DL reasoning over RDFS data. Since conjunctive query answering is always reduced to OWL-QL query answering, this allows queries to be run over large heterogeneous ontologies with its characteristic $AC_0$ data complexity.

[1] http://www.most-project.eu

## 3 Technology

TrOWL is based around two primary technologies. Language transformations, and lightweight reasoners. The most important of these are outlined briefly here.

### 3.1 Language Transformations

TrOWL is the common interface to a number of reasoners. Quill provides reasoning services over RDF-DL and OWL-QL; REL provides reasoning over OWL-EL; and TrOWL can support full DL reasoning using a plug-in reasoner such as Pellet or Fact++. These reasoners and the languages which they support are optimised for certain applications, for example, OWL-QL has excellent ABox query answering performance but it lacks many constructors present in the more expressive flavours of OWL2.

The transformation from OWL2 to OWL-QL is based around Semantic Approximation from OWL-DL to DL-Lite which is described in [7]. Semantic Approximation uses a heavyweight reasoner to guarantee that every axiom in the approximated ontology is valid with respect to the source ontology. Because the semantics of OWL-QL are a subset of, and are hence compatible with, the direct semantics OWL2, this means that for all reasoning results against the approximated ontology are sound. In fact, it has been shown in that for conjunctive query answering, which is the strength of the QL language, results against the semantic approximation are also complete for a very large class of queries (those with no non-distinguished variables, or with non-distinguished variables in leaf nodes of the query).

The transformation from OWL2 to OWL-EL is based on the soundness preserving approximate reasoning approach presented in [8]. This is achieved by representing non-OWL-EL concept expressions with fresh named concepts, and maintaining non-OWL EL information, such as complementary relations, in separate data structures. In the reasoning stage, additional completion rules are plugged into the inference engine to restore the semantics of these information. The approximation is syntactic-based and can be performed in linear time. The additional completion rules retain the tractability of OWL2-EL. Thus the overall complexity for OWL2-DL ontologies can be reduced to PTime. Although known to be incomplete, our evaluation shows that, REL can classify existing benchmarks very efficiently with high recall (over 95%) [8].

Other transformation techniques used in TrOWL include forgetting [5,11,10].

### 3.2 Lightweight Reasoners

**Quill.** The Quill reasoner has been implemented in Java using a novel and unique database schema for storing normalised representations of OWL2-QL ontologies. This allows us to rewrite any conjunctive query into a single, simple, SQL query over the underlying database, using the database itself to perform the transitive completion of class and property subsumption with an innovative

exploitation of the way database indices work. To support this we have developed new algorithms to replace those proposed in [4]. for query rewriting, and ontology normalisation. Initial testing across large knowledge bases with deep concept hierarchies, such as the DBPedia dataset and the Yago ontology, shows a significant performance improvement over other DL-Lite query engines. Using the standard query rewriting algorithm PerfectRef over a deep class or property hierarchy can result in a set of hundreds or thousands of conjunctive queries, where our method will only ever result in a single query. Quill supports all reasoning tasks for OWL2-QL, including consistency and satisfiability checking, and query answering, and by using an OWL-DL reasoner it can perform semantic approximation of more expressive ontologies.

**REL.** The REL reasoner is a java implementation of an OWL-EL reasoner, in which an optimisation of the EL+ algorithm [2] has been extended with the completion rules for OWL-EL [1]. This allows REL to provide tractable TBox reasoning for OWL-EL ontologies and make up the core component of the soundness-preserving syntactic approximation. By this way, REL can provide soundness-guaranteed tractable TBox reasoning services for OWL2-DL ontologies. In additional, REL also consists of an OWL-EL conjunctive query engine [12], which allows queries over OWL-EL ontologies been answered more efficiently without semantic approximation.

## 4 Demonstration

Our demonstration of the TrOWL reasoner will focus on two scenarios. The first part of the demo will show how TrOWL differs from traditional reasoners, and give a comparison of the performance of TrOWL for different reasoning tasks. The second part of the demonstration will showcase the case studies from the MOST project and show how TrOWL is helping to solve these.

The proposed structure of the demonstration is:

- Reasoning Demonstration
  - TBox Reasoning
  - Query Answering
  - Comparison with traditional reasoners
- MOST Project Case Studies
  - Validating process refinement
  - Physical device configuration
  - Requirements modelling using ontologies
- Deploying TrOWL
  - As an embedded reasoner
  - As a SPARQL endpoint
  - As a web service

## Acknowledgements

This research has been partially supported by the European Commission and by the Swiss Federal Office for Education and Science within the 7th Framework Programme project MOST number 216691 (cf. http://most-project.eu). We would also like to thank Stuart Taylor, Nophadol Jekjantuk and Yuting Zhao at the University of Aberdeen for their helpful discussion and other contributions.

## References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ Envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI 2005 (2005)
2. Baader, F., Lutz, C., Suntisrivaraporn, B.: Is tractable reasoning in extensions of the description logic el useful in practice? In: Proceedings of the 2005 International Workshop on Methods for Modalities, M4M 2005 (2005)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (2001)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The dl-lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
5. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Forgetting in managing rules and ontologies. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006), Hongkong, pp. 411–419. IEEE Computer Society, Los Alamitos (2006)
6. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (October 2009), http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/
7. Pan, J.Z., Thomas, E.: Approximating OWL-DL Ontologies. In: The Proc. of the 22nd National Conference on Artificial Intelligence (AAAI 2007), pp. 1434–1439 (2007)
8. Ren, Y., Gröner, G., Lemcke, J., Rahmani, T., Friesen, A., Zhao, Y., Pan, J.Z., Staab, S.: Validating process refinement with ontologies. In: Proceedings of the 22nd International Workshop on Description Logics, DL 2009 (2009)
9. Smith, M.K., Welty, C., McGuiness, D.L.: (February 2004) http://www.w3.org/TR/owl-guide/
10. Wang, K., Wang, Z., Topor, R.W., Pan, J.Z., Antoniou, G.: Concept and role forgetting in ALC ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 666–681. Springer, Heidelberg (2009)
11. Wang, Z., Wang, K., Topor, R., Pan, J.Z.: Forgetting in DL-Lite. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 245–257. Springer, Heidelberg (2008)
12. Zhao, Y., Pan, J.Z., Ren, Y.: Implementing and evaluating a rule-based approach to querying regular el+ ontologies. In: Proc. of the International Conference on Hybrid Intelligent Systems, HIS 2009 (2009)

# Making the Semantic Data Web Easily Writeable with RDFauthor

Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth

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

**Abstract.** In this demo we present RDFauthor, an approach for authoring information that adheres to the RDF data model. RDFauthor completely hides syntax as well as RDF and ontology data model difficulties from end users and allows to edit information on arbitrary RDFa-annotated web pages. RDFauthor is based on extracting RDF triples from RDFa-annoted Web pages and transforming the RDFa-annotated HTML view into an editable form by using a set of authoring widgets. As a result, every RDFa-annotated web page can be made writeable, even if information originates from different sources.

## 1 Introduction

To a large extent the overwhelming success of the World Wide Web was based on the ability of ordinary users to author content easily. In order to publish content on the WWW, users had to do little more than to annotate text files with few, easy-to-learn HTML tags. Unfortunately, on the semantic data web the situation is slightly more complicated. Users do not only have to learn a new syntax (such as N3, RDF/XML or RDFa), but also have to get acquainted with the RDF data model, ontology languages (such as RDF-S, OWL) and a growing collection of connected RDF vocabularies for different use cases (such as FOAF, SKOS and SIOC).

Previously, many applications were developed to ease the syntax side of semantic authoring [4,2]. The *RDFauthor* approach[1] is based on the idea of making arbitrary XHTML views with integrated RDFa annotations editable. *RDFa* [1] is the W3C Recommendation, which allows to combine human and machine-readable representations within a single XHTML document. RDFauthor builds on RDFa by preserving provenance information in RDFa representations following the named-graph paradigm and by establishing a mapping from RDFa view representations to authoring widgets. On configurable events (such as the clicking of a button or moving over a certain information fragment with the mouse) the widgets will be activated and allow the editing of all RDFa-annotated information on the Web page. While editing, the widgets can access background information sources on the Data Web in order to facilitate the reuse of identifiers or to encourage the interlinking of

[1] http://aksw.org/Projects/RDFauthor

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 436–440, 2010.

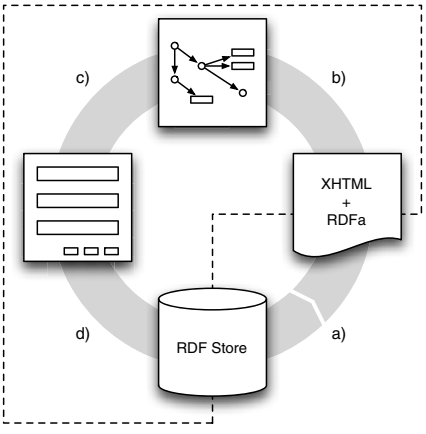



**Fig. 1.** Editing cycle for an RDFa-enhanced web page. The processes involved are a) page creation and delivery, b) client-side page processing, c) form creation and d) update propagation via SPARQL/Update. The dashed line encloses the processes carried out by RDFauthor.

resources. Our resource editing widget, for example, suggests suitable, previously defined resources derived from calls to the Sindice Semantic Web index [5]. Once editing is completed, the changes are propagated to the underlying triple stores by means of the SPARQL/Update language.

## 2 System Architecture Overview

The basic cycle of how web pages are edited with RDFauthor is depicted in figure 1. It is composed of four distinct processes, of which (b) client-side page processing, (c) widget selection and (d) form creation and update propagation via SPARQL/Update [3] are handled by RDFauthor.

In order to link RDFa annotations on the page to the respective querying/update services (i.e. SPARQL/Update endpoints), we propose the use of the `link` HTML tag with an `about`-attribute to identify the named graph, a `rel`-attribute with the value `update:updateEndpoint` and a `href`-attribute with the URL of the respective SPARQL/Update endpoint. Another option to declare graph metadata is the use of empty `span`- or `div`-elements together with the RDFa attributes inside the body of the page.

The initiation of the RDFauthor editing processes happens through element-based or page-wide trigger events, which can be e.g. the clicking of an edit button, hovering over an RDFa element or a bookmarklet. When the user finishes the editing process, all widgets involved are asked to update the page graph with their changes. The difference between the original and modified page graphs are calculated (i. e. added statements, removed statements), yielding a diff graph. The associated store to each graph is then updated with the respective diff graph by means of SPARQL/Update operations.

## 3 Use Cases

We demonstrate the benefits of RDFauthor, by showcasing the integration of the approach into two Semantic Web applications and a standalone RDFauthor bookmarklet facilitating the collection of RDF data from arbitrary RDFa-annotated websites.

### 3.1 OntoWiki

OntoWiki [2][2] is a tool for browsing and collaboratively editing RDF knowledge bases. It differs from conventional Semantic Wikis in that OntoWiki uses RDF as its natural data model instead of Wiki texts. Information in OntoWiki is always represented according to the RDF statement paradigm and can be browsed and edited by means of views, which are generated automatically by employing the ontology features, such as class hierarchies or domain and range restrictions. OntoWiki adheres to the Wiki principles by striving to make the editing of information as simple as possible and by maintaining a comprehensive revision history. It has recently been extended to incorporate a number of Linked Data features, such as exposing all information stored in OntoWiki as Linked Data as well as retrieving background information from the Linked Data Web. Apart from providing a comprehensive user interface, OntoWiki also contains a number of components for the rapid development of Semantic Web applications, such as the RDF API Erfurt, methods for authentication, access control, caching and various visualization components.

RDFauthor is used in OntoWiki both in the generic resource property view as well as in extensions which render resources in a domain-specific way (e. g. specific visualizations for SKOS concepts or FOAF persons). In order to perform the integration, we have extended OntoWiki in two ways:

1. We extended the default properties view for resources and all other views with RDFa attributes to annotate which data is presented as well as to link the graph to the internal update service. Since OntoWiki is entirely based on an RDF store, this extension was easy to implement. Likewise, all extension developers had to extend their views, e. g. for SKOS concepts.
2. We included RDFauthor by referencing it in the head of every OntoWiki page and adding JavaScript edit buttons on every page where data should be editable.

### 3.2 vCard and Publication Mashup

In order to showcase the simultaneous authoring of information from multiple sources, we integrated RDFauthor into the text-based wiki application WackoWiki[3]. WackoWiki is often used in small and medium-sized companies as well as in small organizations such as research groups.

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

[3] `http://wackowiki.org`

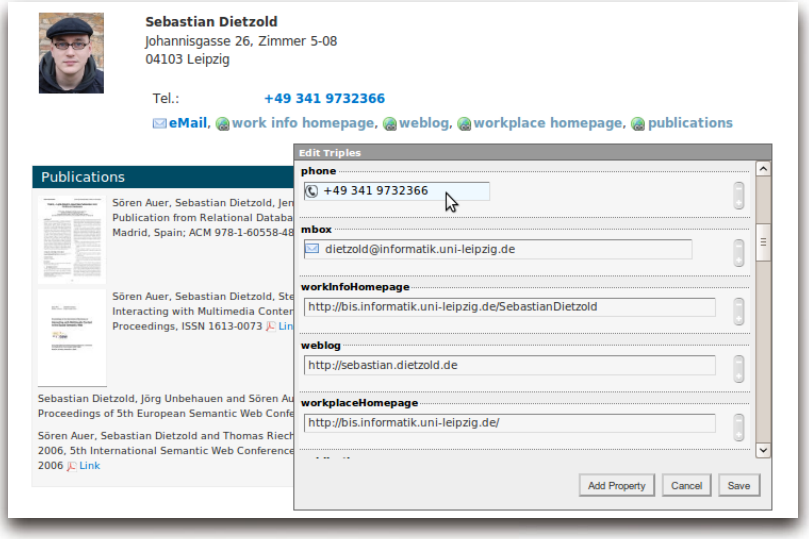



**Fig. 2.** RDFa-enhanced FOAF vCard and publications mashup with statements from different named graphs. In addition to generic literal and resource widgets, authoring widgets for the URI schemes `tel:` and `mailto:` hide the URI syntax.

The AKSW research group uses a WackoWiki installation for its entire web page (`http://aksw.org`) and integrates external data sources by means of so-called WackoWiki actions. Actions are small scripts that prepare some content and output it at the given position in the wiki page. In addition, actions are able able to fetch data from external resources, thus allowing us to use structured information on different places in the wiki, e. g. by presenting the last publications selected by author, project or topic.

While integrating and presenting this information is easy and covered by many applications and techniques, full read/write integration of such external resources is tackled by RDFauthor. By employing RDFauthor, users of our wiki are able to edit both the wiki page and the structured information in one place and avoid using different web applications for one edit task and with different data. The RDFauthor GUI on top of a Wacko Wiki page is depicted in figure 2.

### 3.3 Data Collection from RDFa Websites

Another interesting usage scenario, which is more concerned with collecting data instead of editing, is described in this section. Most of the RDFa-enabled pages on the web do not yet contain provenance and update information. However, RDFauthor also allows to use an arbitrary update endpoint, which does not necessarily have to match the originating endpoint.

Since a SPARQL/Update-capable RDF store and a target graph is all the information required for using RDFauthor, it is easy to embed these into a

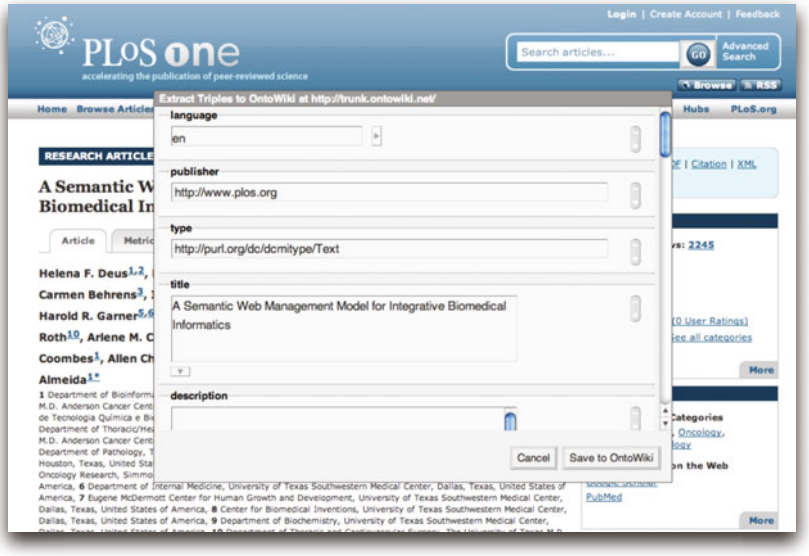



**Fig. 3.** RDFauthor overlay with widgets for triples extracted from a PLoS web page

bookmarklet used to initialize the editing process. In this case, the number of possible SPARQL/Update endpoints is limited to those under one's control. RDFauthor extracts the data from any page visited and displays the edit form. The data can be revised and unwanted statements can be removed from the view. Saving works, however, differently: instead of propagating the changed data back to the original source, it is sent to one's own RDF store and saved into the previously set-up graph.

## References

1. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and Processing. In: Recommendation, World Wide Web Consortium (W3C) (October 2008), http://www.w3.org/TR/rdfa-syntax/
2. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
3. Seaborne, A., Manjunath, G.: SPARQL/Update: A language for updating RDF graphs. Technical Report Version 5: 2008-04-29, Hewlett-Packard (2008)
4. Tudorache, T., Noy, N.F., Tu, S., Musen, M.A.: Supporting Collaborative Ontology Development in Protégé. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 17–32. Springer, Heidelberg (2008)
5. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the Open Linked Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)

# BioNav: An Ontology-Based Framework to Discover Semantic Links in the Cloud of Linked Data

María-Esther Vidal[1], Louiqa Raschid[2], Natalia Márquez[1], Jean Carlo Rivera[1], and Edna Ruckhaus[1]

[1] Universidad Simón Bolívar, Caracas, Venezuela
{mvidal,nmarquez,jrivera,ruckhaus}@ldc.usb.ve
[2] University of Maryland
louiqa@umiacs.umd.edu

**Abstract.** We demonstrate BioNav, a system to efficiently discover potential novel associations between drugs and diseases by implementing Literature-Based Discovery techniques. BioNav exploits the wealth of the Cloud of Linked Data and combines the power of ontologies and existing ranking techniques, to support discovery requests. We discuss the formalization of a discovery request as a link-analysis and authority-based problem, and show that the top ranked target objects are in correspondence with the potential novel discoveries identified by existing approaches. We demonstrate how by exploiting properties of the ranking metrics, BioNav provides an efficient solution to the link discovery problem.

## 1 Introduction

Emerging infrastructures provide the basis for supporting on-line access to the wealth of scientific knowledge captured in the biomedical literature. The two largest interconnected bibliographic databases in biomedicine, PubMed and BIOISIS, illustrate the extremely large size of the scientific literature today. PubMed publishes at least 16 million references to journal articles, and BIOSIS more than 18 million of life science-related abstracts. On the other hand, a great number of ontologies and controlled vocabularies have become available under the umbrella of the Semantic Web and they have been used to annotate and describe the contents of existing Web available sources. For instance, MeSH, RxNorm, and GO are good examples of ontologies comprised of thousands of concepts and that are used to annotate publications and genes in the NCBI data sources.

Furthermore, in the context of the Linking Data project, a large number of diverse datasets that comprise the Cloud of Linked Data are available. The Cloud of Linked Data has had an exponential growth during the last years; in October 2007, datasets consisted of over two billion RDF triples, which were interlinked by over two million RDF links. By May 2009 this had grown to 4.2 billion of RDF triples interlinked by around 142 million of RDF links. At the time this paper was written, there were 13,112,409,691 triples in the Cloud of Linked Data; datasets can be about medical publications, airport data, drugs, diseases, clinical trials, etc. It is of particular interest, the portion of the Cloud that relates life science data such as diseases, traditional Chinese medicine, pharmaceutical companies, medical publications, genes and proteins, where concepts

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 441–445, 2010.

are derived from sites such as ClinicalTrials.gov, DrugBank, DailyMed, SIDER, TCM-GeneDIT, Diseasome and OMIM. To fully take advantage of the available data, and to be able to recognize novel discoveries, scientists will still have to navigate through the Cloud and compare, correlate and mine linked data; thus, they may have to spend countless hours to recognize relevant findings.

To provide support on the discovery task of potential novel associations between already published topics in existing bibliographic datasets, Literature-based discovery (LBD) techniques have been developed. LBD methods follow a disease-cure trajectory to guide the search in the space of implicit associations between scientific publications and their annotations or cites. Annotations correspond to concepts from controlled vocabularies or ontologies. LBD can perform Open or Closed discoveries, where a scientific problem is represented by a set of articles that discuss a particular topic A, and the goal is to prove the significance of the associations between A and some other topics C discussed in the set of articles reachable from the publications relevant to the topic A. Srinivasan et al [5] improved previous LBD techniques by recognizing that articles in PubMed have been curated and heavily annotated with controlled vocabulary terms from the MeSH (Medical Subject Heading) ontology. Srinivasan's algorithm considers that topics A, B and C are MeSH terms used to annotate or index PubMed publications. Thus, links from topic A to publications in the second layer are built by searching with topic A on PubMed. Links from publications in the second layer of the graph to MeSH terms in the third layer, are constructed by extracting the MeSH terms annotations from the publications and selecting the ones associated with the UMLS (Unified Medical Language System) semantic types: Gene or Genome; Enzyme; and Amino Acid, Peptide or Protein. The extracted MeSH terms or B set, are used to search on PubMed and a new set of publications is recovered. Again MeSH term annotations are extracted from these publications, and the terms of the UMLS types Disease or Syndrome and Neoplastic Process are retrieved to comprise the set of topics C. Figure 1 illustrates the LBD experiment reported by Srinivasan et al [5] where the MeSH term curcumin is associated with top-5 MeSH terms. Edges of the graph are labeled with weights that are computed by using an extension of the TF*IDF scores; these scores are used to rank the different paths and determine the potential novel associations.

Although Srinivasan's approach enhances previously LBD methods, this solution may be still costly. To provide an efficient solution, we propose the ontology-based

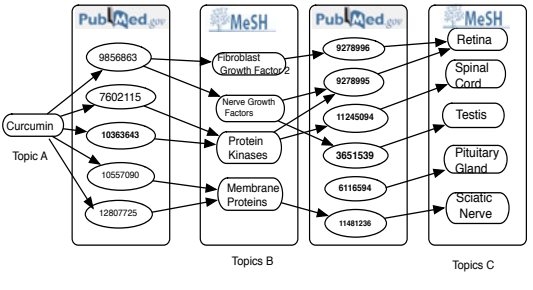



**Fig. 1.** Example of a Literature-Based Discovery experiment where the MeSH curcumin is related to the retinal disease

system BioNav that provides a framework to discover potential novel associations between drugs and diseases. In this demonstration, we will use the Srinivasan's experiment to show effectiveness and efficiency of the proposed discovery techniques. The paper is comprised of three additional sections: BioNav is presented in section 2, section 3 describes the use cases to be demonstrated, and conclusions are given in section 4.

## 2 The BioNav Architecture

In Figure 2 we present the BioNav architecture [6]. BioNav is comprised of four main components: a Catalog, a Query Optimizer, a Source Path Discovery component, and a Semantic Link Discovery Engine.

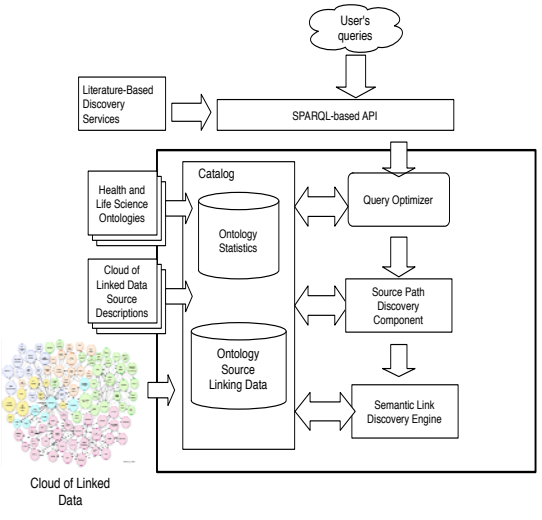



**Fig. 2.** The BioNav System

The Catalog maintains information about the sources and data that comprise the Cloud of Linked Data; also ontologies are used to describe the meaning of the data. Queries are expressed in terms of the ontology concepts and they are evaluated against the data stored in the Cloud. The result of evaluating a query corresponds to a list of MeSH terms that are semantically associated with terms in the query.

Once a discovery query is received, the parser checks if it is syntactically correct, and cost-based optimization techniques are performed to identify an efficient query execution. An execution plan corresponds to an ordering of the concepts referred in the query, and minimizes the cardinality of the intermediate facts that need to be computed to answer a query and the execution time [4].

Once the input query is optimized, it is rewritten in terms of the data sources that need to be accessed to evaluate the query; a graph-based meta-heuristic Best-First is used to enumerate the paths between data sources that need to be traversed to evaluate the query. Source paths are evaluated by traversing the sources in the order specified in the path.

The answer of the query will be comprised of paths in the Cloud of Linked Data that are locally stored in the BioNav catalog. Paths in the query answer will be computed by traversing the source paths. BioNav uses link-analysis and authority-flow ranking metrics to rank the discovered associations. BioNav ranking techniques assume that the data paths to be ranked comprise a layered graph $lgODG$=($V_{lg}$, $E_{lg}$) of $k$ layers, $l_1$, ..., $l_k$. Odd layers are composed of MeSH terms while even layers are sets of publications. An edge from a term $b$ to a publication $p$ indicates that $p$ is retrieved by the PubMed search engine when $b$ is the search term. An edge from a publication $p$ to a term $b$ represents that $p$ is annotated with $b$. Each edge $e = (b, p)$(resp., $e = (p, b)$) between the layers $l_i$ and $l_{i+1}$ is annotated with the $TF \times IDF$ that represents how relevant is the term $b$ in the collection of documents in $l_{i+1}$, or a document relevance regarding to a set of terms. The most important terms or publications correspond to the highly ranked Mesh terms or publications in the last layer of the $lgODG$. In this paper we focus on an extension of the Object Rank [1,2] and Path Count [3] metrics for layered graphs or *layered graph Weighted Path Count* (lgWP), which is defined as follows:

A ranking vector R of the target objects in the layered Open Discovery graph $lgODG$ of $k$ layers is defined by a transition matrix $A$ and an initial ranking vector $R_{ini}$:

$$R = A^{k-1} R_{ini} = (\prod_{l=1}^{k-1} A)\, R_{ini}$$

An entry $A[u, v]$ in the transition matrix $A$, where $u$ and $v$ are two data objects in $lgODG$, corresponds to $\alpha(u, v)$ or 0. The value of $\alpha(u, v)$ is the weight that represents how relevant is the object $u$ for the object $v$. Nodes with high lgWP scores are linked by many nodes or linked by highly scored nodes.

$$A[u, v] = \begin{cases} \alpha(u, v) & \text{if } (u, v) \in E_{lg}, \\ 0 & \text{otherwise.} \end{cases}$$

To speed up the tasks of computing the lgWP, we build a Bayesian network with the knowledge encoded in the layered Open Discovery graph, and we perform a Direct Sampling algorithm to traverse the network and just visit the publications or MeSH terms that conduce to potential novel discoveries. This sampling approach has the ability to identify a large number of the potential novel discoveries, while a reduced number of nodes that need to be visited by at least one order of magnitude.

## 3 Demonstration of Use Cases

In this demonstration, we will show the different steps of the BioNav LBD process in several real-world experiments. The objective is to show the applicability and performance of BioNav for the top-5 diseases that can be treated with the MeSH terms curcumin and aloe. We will demonstrate the following scenarios:

- We show effectiveness by demonstrating the process to compute the potential novel associations between the substance curcumin and MeSH terms that correspond to diseases. We use a ranking metric lgWP to discriminate the potential novel discoveries. Weights in the edges of the graph represent the relevance of the MeSH terms

or publications, and they are computed by using an extension of the $TF \times IDF$ scores. We show that the top-5 MeSH terms in the last layer of the graph correspond to 80% of the top-5 potential novel diseases discovered by Srinivasan et al. [5].
- We show efficiency by computing the number of intermediate and target MeSH terms and publications that need to be visited to identify the novel associations. In BioNav, properties of lgWP are exploited with approximate methods to avoid traversing MeSH terms or publications that do not directly or indirectly link a potential novel discovery. We show that for the MeSH terms curcumin and aloe, the number of visited MeSH terms and publications can be reduced by at least one order of magnitude when these approximate methods are executed.
- Finally, we show the different steps of the open LBD process by selecting a MeSH term that corresponds to a drug or substance, and the potential novel associations of this term with MeSH terms that correspond to diseases. We discuss the impact of the proposed techniques in the Linked Data Cloud.

## 4 Conclusions

In this demonstration, we present BioNav, an ontology-based tool that supports the discovery of semantic associations between linked data. We demonstrate how link-analysis and authority-based ranking metrics can be used in conjunction with ontology annotations on linked data, to discover potential novel discoveries; also we demonstrate how the properties of the ranking metrics can be exploited to avoid traversing MeSH terms and publications that do not conduce to novel potential discoveries. We show real-world use cases that suggest BioNav is able to efficiently discover almost all the associations identified by state-of-the-art approaches.

## References

1. Balmin, A., Hristidis, V., Papakonstantinou, Y.: Objectrank: Authority-based keyword search in databases. In: Proceedings VLDB, pp. 564–575 (2004)
2. Page, L., Brin, S., Motwani, R.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
3. Raschid, L., Wu, Y., Lee, W., Vidal, M., Tsaparas, P., Srinivasan, P., Sehgal, A.: Ranking target objects of navigational queries. In: WIDM, pp. 27–34 (2006)
4. Ruckhaus, E., Ruiz, E., Vidal, M.: Query evaluation and optimization in the semantic web. In: TPLP (2008)
5. Srinivasan, P., Libbus, b., Kumar, A.: Mining medline: Postulating a beneficial role for curcumin longa in retinal diseases. In: Hirschman, L., Pustejovsky, J. (eds.) LT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases, pp. 33–40 (2004)
6. Vidal, M.-E., Ruckhaus, E., Marquez, N.: BioNav: A System to Discover Semantic Web Associations in the Life Sciences. In: ESWC 2009-Poster Session (2009)

# StarLion: Auto-configurable Layouts for Exploring Ontologies

Stamatis Zampetakis, Yannis Tzitzikas,
Asterios Leonidis, and Dimitris Kotzinos

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

**Abstract.** The visualization of ontologies is a challenging task especially if they are large. We will demonstrate `StarLion`, a system providing exploratory visualizations which enhance the user understanding. `StarLion` combines many of the existing visualization methods with some novel features for providing better 2D layouts. Specifically, one distinctive feature of `StarLion` is the provision of Star-like graphs of variable radius whose layout is derived by a Force Directed Placement algorithm (*FDPA*) specially adapted for RDF Schemas. This approach enables users to gradually explore and navigate through the entire ontology without overloading them. `StarLion` can also handle multiple namespaces, a very useful feature for assisting the understanding of interdependent ontologies. Another distinctive characteristic of `StarLion` is the provision of a novel method for configuring automatically the *FDPA* parameters based on layout quality metrics, and the provision of an interactive configuration method offered via an intuitive tool-bar.

## 1 Introduction

The understanding of an ontology with many classes and properties represented as a directed graph (Figure 1(b)), is a hard and time consuming task. Our objective is to alleviate this problem by providing 2D visualizations that could aid users in tasks like: selection of a suitable ontology from a corpus of ontologies, understanding the structure of one particular ontology, and understanding a number of interrelated ontologies.

The field of graph drawing and visualization is very broad. There are many works using *FDP* algorithms and some of them also support star-like views with variable radius. Most of these works refer to general (plain) graphs and they are not RDF-specific. RDF graphs contain more information than plain graphs and have more visualization needs (e.g subclass hierarchies must be vertical). For this reason we did not rely on such algorithms but we designed a dedicated force directed algorithm which combines the *spring-model*([2,4,3]) with the *magnetic-spring model*([6,5]). Apart from this, the notion of namespaces does not exist in plain graphs, while in RDF graphs plays an important role. Finally, and since different graphs exhibit different graph features the layout algorithm must be

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 446–450, 2010.

configurable (ideally auto-configurable) for deriving aesthetically pleasing layouts. We will demonstrate `StarLion`[1] a system for ontology visualization, and we will focus on (a) the support of real-time exploration through star-like graphs of variable radius since this allows users to explore large schemas while controlling the amount of displayed information on the basis of user preferences or screen-size constraints, (b) the visualization of multiple (dependent) ontologies since every ontology usually extends and reuses elements from other ontologies, and (c) the manual (interactive) and the automatic (based on quality metrics) configuration of the *FDPA* layout algorithm.

## 2 The StarLion Approach

**Dependent Namespaces.** It is often useful to load along with a schema the schemata (Namespaces) on which it depends. This enhances and completes the understanding by the user for the schema who is interested in, but also multiplies the visualization difficulties and makes it a cumbersome task even for experienced users. To address this problem, we propose a feature where upon loading a specific namespace NS, the user is able to see all namespaces upon which NS depends on, and select those to be visualized, while each namespace's classes are drawn using a different color. `StarLion` offers 3 options for the visualization of namespaces. Transitive option (Fig1(a)(i)) loads a namespace (e.g NS1) plus all superclasses of the classes of NS1 that belong to different namespaces. Direct option (Fig1(a)(ii)) loads the namespace (e.g NS1) and only the directly connected elements from the other namespaces, while full option (Fig1(a)(iii)) loads all dependent namespaces completely.

**StarView.** During *StarView* interaction, the user selects one class $c$ as focal point. Class $c$ is visualized along with all related classes in distance (radius) k from $c$, creating Star-like graphs. Figure1(c) shows the *StarView* with k=2. The focal point can be changed by clicking on any of the visualized elements.

**Layout Algorithm.** For deriving automatically the 2D layout we view the graphs as mechanical systems. We adopt the force model that was proposed in [7] for visualizing E-R diagrams. Specifically, that model combines the *spring-model* (proposed and developed in [2,4,3]) with the *magnetic-spring model* (proposed in [6,5]). In our case we apply them on RDFS graphs. Nodes (in our case classes) are viewed as equally charged particles which *repel* each other. Edges (i.e. RDF properties and $isA$ relationships) are viewed as springs that *pull* their adjacent nodes. Moreover, we assume that the springs that correspond to $isA$ links are all magnetized and that there is a global magnetic field that acts on these springs. Specifically, this magnetic field is parallel (i.e. all magnetic forces operate in the same direction) and the $isA$ springs are magnetized unidirectionally, so they tend to align with the direction of the magnetic field, here upwards. This is because the classical way (in semantic web, object-oriented class diagrams, Formal Concept Analysis, Hasse

[1] Implemented in Java using the jgraph library, for more see http://www.ics.forth.gr/~tzitzik/starlion

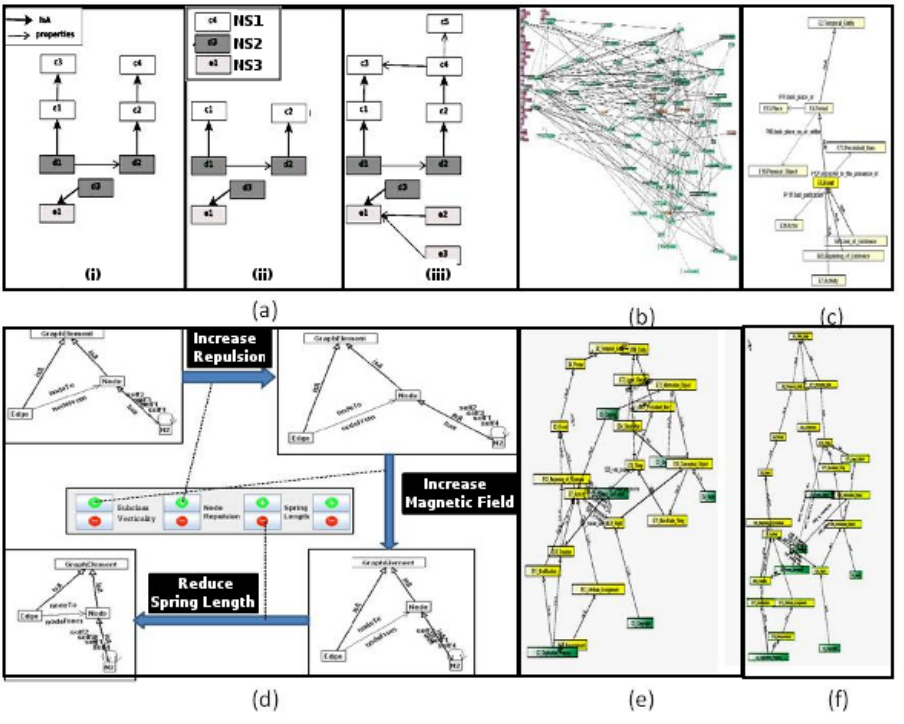



**Fig. 1.** StarLion distinctive features

diagrams in discrete mathematics, etc) of presenting specialization/generalization relationships is to put the superclass above the subclass.

**FDPA Configuration.** Repulsion strength ($K_e$), spring length ($L$) and magnetic strength ($K_m$), are the three parameters that affect the result of the algorithm, and their configuration is one of the crucial issues in *FDPA* algorithms. Since every graph has different features, we have to configure these parameters to get a satisfying layout and this is not easy for a casual user by entering explicitly numeric values. For this reason we added a tool-bar (as shown in Figure 1(d)) which allows the user to relatively increase/decrease each parameter with a few clicks. To further improve the layout and reduce the clicks on that tool-bar, we devised a novel method for configuring automatically these parameters based on a number of quality metrics. This functionality is offered through a, so called "magic" button, that initiates the quality measurement, applies the proper configuration to the parameters and executes the *FDPA* taking away the burden from the user of knowing anything about the parameters. Quality is measured according to two metrics, one measuring how vertical the subclass relationships are, and another measuring the density of the layout. We have conducted a user study which showed that this method does improve the layout.

## 3 Demonstration Scenario

During the live demonstration the audience will see `StarLion` in action over `CIDOC CRM`[2] ontology (Fig1(b)). Despite the big size (number of nodes and edges) of that ontology and the large number of inevitable edge crossings, the modified *FDPA* can give us a good overview. Then we will show how the user can start the exploration through *StarView* (Fig1(c)) for visualizing incrementally parts of the ontology. Someone who wants to obtain information about a specific class in the ontology usually wants to see all super-classes, subclasses and properties related to this class. We will present how this can be done with *StarView* and how easy is to continue this procedure for another class in the neighborhood. Next we are going to visualize `CIDOC CRM Digital` ontology which is an extension of `CIDOC CRM` with six new classes and a dozen of new properties. `CIDOC CRM Digital` obviously depends on `CIDOC CRM` and for this reason we are going to use the dependent name-spaces feature discussed earlier. Finally we will see how the *FDPA* can be configured through the tool-bar (Fig1(d)) to enhance the layout and some results from using the auto-configuration feature. Figure 1(e) shows the layout of a schema after an application of the *FDPA* with default parameters and Figure 1(f) shows the same layout after the auto-configuration of the parameters.

## 4 Related Work

One of the most popular open source tools is the *Jambalaya* plug-in of Protégé (http://protege.stanford.edu), which offers a set of different algorithms (trees, radial, grids) and allows the user to select the type of objects he wants to visualize (e.g. classes only, etc.), but offers no automatic method for restructuring the layout. *Touchgraph*[3] is a commercial product that offers a Star-like view where the selected node is automatically located at the center of the screen with only its directly connected nodes visible. *Welkin*[4] provides a layout algorithm based on a force directed model but it limits user interaction to configuration of the layout and presentation parameters only. *ISWIVE* [1], incorporates the topic features from Topic Maps into RDF and thus into the corresponding visualizations. Finally, *RDF-Gravity* provides a standard but non-configurable force directed layout with zooming facility, while the exploration is achieved only through filtering and textual information presentation.

Compared to ours the aforementioned tools do not offer methods for configuring automatically the layout and most of them lack the ability to extend visualizations beyond radius 1. The combination of the RDFS-adapted *FDPA* with *StarView* of variable radius and dependent namespaces, offers a powerful exploration method that is unique.

[2] The international standard (ISO 21127:2006) for controlled exchange of cultural heritage information. `http://cidoc.ics.forth.gr/`

[3] http://www.touchgraph.com/navigator.html

[4] http://simile.mit.edu/welkin/

## 5 Conclusion

`StarLion` is a new visualization and exploration tool of ontologies, aiming at enhancing their understanding by the ordinary user; especially for those ontologies that are either large or interdependent (depending on one another) or both. As also verified by a user study (whose results are not reported here for reasons of space) the exploration through star-like graphs and the layout improvement through the interactive/automatic configuration methods results in a flexible and intuitive interaction and highly improves users' understanding of the ontologies at hand.

## References

1. Chen, X., Fan, C., Lo, P., Kuo, L., Yang, C.: Integrated visualization for semantic web. In: Intern. Conf. on Advanced Information Networking and Applications, pp. 701–706 (2005)
2. Eades, P.: A heuristic for graph drawing. Congressus Numerantium 42(3), 146–160 (1984)
3. Fruchterman, T., Reingold, E.: Graph drawing by force-directed placement. Software - Practice and Experience 21(11), 1129–1164 (1991)
4. Kamada, T.: On Visualization of Abstract Objects and Relations. Ph.D. thesis, Dept. of Information Science, Univ. of Tokyo (1988)
5. Sugiyama, K., Misue, K.: A simple and unified method for drawing graphs: Magnetic-spring algorithm. In: Graph Drawing, pp. 364–375. Springer, Heidelberg (1994)
6. Sugiyama, K., Misue, K.: Graph drawing by magnetic-spring model. Journal on Visual Lang. Comput. 6(3), 217–231 (1995)
7. Tzitzikas, Y., Hainaut, J.: How to tame a very large er diagram (using link analysis and force-directed placement algorithms). In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 144–159. Springer, Heidelberg (2005)

# Concept Extraction Applied to the Task of Expert Finding

Georgeta Bordea

Unit for Natural Language Processing,
Digital Enterprise Research Institute,
National University of Ireland, Galway
georgeta.bordea@deri.org

## 1 Research Problem

The Semantic Web uses formal ontologies as a key instrument in order to add structure to the data, but building domain specific ontologies is still a difficult, time consuming and error-prone process since most information is currently available as free-text. Therefore the development of fast and cheap solutions for ontology learning from text is a key factor for the success and large scale adoption of the Semantic Web. Ontology development is primarily concerned with the definition of concepts and relations between them, so one of the fundamental research problems related to ontology learning is the extraction of concepts from text. To investigate this research problem we focus on the expert finding problem, i.e, the extraction of expertise topics and their assignment to individuals. The ontological concepts we extract are a person's skills, knowledge, behaviours, and capabilities.

For increased efficiency, competitiveness and innovation, every company has to facilitate the identification of experts among its workforce. Even though this can be achieved by using the information gathered during the employment process and through self-assessment, a person's competencies are likely to change over time. Information about people's expertise is contained in documents available inside an organisation such as technical reports but also in publicly available resources, e.g., research articles, wiki pages, blogs, other user-generated content. The human effort required for competency management can be reduced by automatically identifying the experts and expertise topics from text. Our goal is to explore how existing technologies for concept extraction can be advanced and specialised for extracting expertise topics from text in order to build expertise profiles.

## 2 Related Work

Expertise or competence management is a research topic from the area of knowledge management that is concerned with the "identification of skills, knowledge, behaviours, and capabilities needed to meet current and future personnel selection needs" [1]. In this thesis we focus on gathering the knowledge of an organisation in terms of scientific topics and technologies.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 451–456, 2010.

Extensive work has been done for the task of expert finding using information retrieval techniques. In these approaches users look for experts in a collection of documents by giving a query with the topics of interest [2,3,4]. The proposed solutions are matching the user's query against the document collection in order to find the experts. This approach makes the assumption that the user is looking for an expertise topic, so it is not possible to find the expertise profile for a person. Other approaches rely on ontologies for competency management, building inference services ([5,6]). The ontologies are used for matchmaking services that bring together the skills demand and supply [7], but these methods can not be applied for domains where an ontology of skills is not already built. An approach to build an ontology of competencies has been proposed in [8], making use of an already built domain ontology, but this can not be applied for domains where an ontology is not defined or where new concepts are introduced often.

In [9] the relations between people and skills are extracted as a network of associations, both people and competencies being handled as entities. Although here the dynamic and automated support of expertise management is addressed, a deeper analysis of expertise topics is needed. Expertise should be analysed on several levels (knowledge, ability, centrality, and context). We analyse expertise on all these levels and we implement an integrated text mining strategy for each of them. In [10] an approach based on text genre specific lexico-syntactic patterns from scientific publications is investigated , but only a short list of context patterns manually identified is considered. In our work we propose an automatic method to identify context patterns that will increase the number of expertise topics that are considered ([11,12]).

## 3 Proposed Approach

The thesis is based on state of the art techniques from term and keyword extraction but applies these techniques to authors rather than documents. The novel aspect that our work covers is how to learn topic extraction patterns using web based knowledge sources (Linked Data[1], ontologies, etc.) as background knowledge. An important research challenge is the lexical disambiguation in the context of Linked Data, i.e., how can we reliably disambiguate extracted topics that have more than one possible interpretation in order to assign them to a unique URI. Our work is based on state of the art techniques in word sense disambiguation extended with a notion of semantic context as provided by web based knowledge sources. The thesis focuses on the topics illustrated in the following sections.

### 3.1 Extracting Expertise Topics

Text genre specific lexico-syntactic patterns, i.e., frequently occurring patterns of particular lexical items or words in a certain syntactic sequence, are central to our approach. Some patterns are specific to a scientific area (e.g., for computer

[1] Linked Open Data: http://linkeddata.org

science: "implementation of", "algorithm for") while other patterns are used in any scientific domain (e.g., "approach for", "analysis of"). We first consider the list of context patterns identified in [10] and then we use the extracted expertise topics in order to identify other context patterns.

Using the syntactic description of a term, we discover candidate expertise topics in the vicinity of context patterns. Similar to the term weighting approach in information retrieval we use a combination of statistical measures to rank the candidate expertise topics, taking into consideration the frequency and the topic length. In addition we analyse the structure of a document and the relation between an expertise topic and the section of the document where it was extracted. We are using the Yahoo BOSS[2] search engine to filter out too general or too specific expertise topics by comparing the number of occurrences in the corpora and on the web. After extracting the linguistic realisations of the concepts from the text, we associate each expertise topic with background knowledge available from the Linked Data cloud. To achieve this, we disambiguate the terms that refer to several concepts using word sense disambiguation techniques. Different terms can refer to the same concept, therefore we explore the similarity of expertise topics to associate synonym expertise topics to a concept. In addition we plan to investigate the relation between expertise topics based on their co-occurrence and to explore different methods for expertise topic clustering.

### 3.2 Extracting Expertise Profiles

The expertise topics in a document are added to the expertise profile for the authors of the document, considering that they are subject matter experts as suggested in [13]. We assign a measure of relevance to each topic from the document collection. The measure of relevance for an author is computed using an adaptation of the standard information retrieval measure TF/IDF. The set of documents of a researcher is considered as a virtual document, and we measure the relevancy of each expertise topic over this virtual document. To identify a researcher's expertise level we take into consideration the performance indicators introduced in [14]: knowledge (coverage of the expertise graph), ability (practical skills associated with expertise) and transfer (centrality and application in different contexts).

## 4 Methodology and Current Contributions

We are planning to evaluate the proposed approach by implementing solutions for the problem of expertise topic and expertise profile extraction presented in Section 3. We will compare the extracted expertise topics with the terms extracted by systems that perform terminology extraction. We will compare our results from the computer science domain with the ACM topic hierarchy[3] and the

[2] Yahoo BOSS: http://developer.yahoo.com/search/boss

[3] ACM topic hierarchy: http://www.acm.org/about/class/1998

results from computational linguistics domain with the LT World[4] Ontology on Language Technology. Another method is to compare the expertise topics with the topics mentioned in the call for papers for the conference of each scientific publication. The evaluation of expertise profiles will be performed with a user study, asking a group of researchers to evaluate their own expertise profile, and also the expertise profiles for a set of well known researchers from their field. Advanced precision and recall measures (e.g. learning accuracy) will be used in this.

We currently use three different data sets for experiments: a corpus of scientific publications from Semantic Web conferences[5], a collection of articles published by researchers working in a web science research institute, and the ACL Anthology Reference Corpus[6]. The first dataset consists of 680 papers and 1692 researchers from 11 semantic web conferences starting from 2006 to 2009. The second dataset contains 405 scientific publications and 362 researchers. The ACL Anthology Reference Corpus is a much larger set that consists of 10921 scientific articles starting from 1965 to 2006 and 9983 researchers.

The first prototype for expertise mining is using 46 different context patterns manually identified by inspecting a set of publications. The system builds expertise profiles for an author as a list of ranked expertise topics. The top 5 expertise topics extracted from the Semantic Web Corpus are presented in Table 1(a). You can observe that the results are promising because the top expertise topics are relevant for the Semantic Web area. We also computed the expertise profile for researchers over the years. In Table 1(b) we can observe that the researcher was interested on topics related to data in 2006 but he was more interested in information retrieval in the next year.

**Table 1.** Top expertise topics from the Semantic Web Corpus

(a) Top expertise topics

| **Expertise Topics** |
|---|
| semantic web |
| social networks |
| Web services |
| semantic web services |
| SW objects |

(b) Top expertise topics for Stefan Decker

| **2006** | **2007** |
|---|---|
| Semantic Web | Semantic Web |
| RDF data | information retrieval |
| Semantic Web data | query processing |
| web pages | RDF data |
| inverted index | PDF documents |

## 5 Conclusions and Future Work

This paper explored how existing techniques for concept extraction can be advanced and specialised for the application of expert finding. After presenting our approach and methodology, we described some preliminary results that show we

[4] LT World: http://www.lt-world.org/
[5] Semantic Web Corpus: http://data.semanticweb.org
[6] ACL Anthology Reference Corpus: http://acl-arc.comp.nus.edu.sg

extracted relevant Semantic Web expertise topics. The next step is to improve the current prototype by automatic context pattern discovery. We will then analyse the performance of the context patterns and the relation between the document structure and the expertise topics. To encourage the use of the extracted results for other applications we are planning to set up a SPARQL[7] endpoint for expertise data access, based on an expertise ontology that we will also develop. For the evaluation of the proposed algorithms we will participate in the SemEval[8] evaluation challenge for Automatic Keyphrase Extraction from Scientific Articles.

## Acknowledgements

This work is supported in part by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). We wish to thank Paul Buitelaar for his valuable contributions and supervision.

## References

1. Draganidis, F., Metzas, G.: Competency based management: A review of systems and approaches. Information Management and Computer Security 14(1), 51–64 (2006)
2. Macdonald, C., Ounis, I.: Voting for candidates: adapting data fusion techniques for an expert search task. In: CIKM 2006: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 387–396. ACM, New York (2006)
3. Serdyukov, P., Rode, H., Hiemstra, D.: Exploiting sequential dependencies for expert finding. In: SIGIR 2008: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 795–796. ACM, New York (2008)
4. Craswell, N., Hawking, D., Vercoustre, A.M., Wilkins, P.: P@noptic expert: Searching for experts not just for documents. In: Ausweb, pp. 21–25 (2001)
5. Sure, Y., Maedche, A., Staab, S.: Leveraging corporate skill knowledge – from proper to ontoproper. In: Proceedings of the Third International Conference on Practical Aspects of Knowledge Management, pp. 30–31 (2000)
6. Kunzmann, C., Schmidt, A.: Ontology-based competence management for healthcare training planning: A case study. In: 6th International Conference on Knowledge Management, IKNOW 2006 (2006)
7. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Piscitelli, G., Coppi, S.: Knowledge based approach to semantic composition of teams in an organization. In: SAC 2005: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 1314–1319. ACM, New York (2005)
8. Posea, V., Harzallah, M.: Building a competence ontology. In: Proceedings of the Workshop Enterprise Modelling and Ontology of the International Conference on Practical Aspects of Knowledge Management, PAKM 2004 (2004)

[7] SPARQL: http://www.w3.org/TR/rdf-sparql-query
[8] SemEval: http://semeval2.fbk.eu/semeval2.php

9. Zhu, J., Goncalves, A.L., Uren, V.S., Motta, E., Pacheco, R.: Mining web data for competency management. In: Proceedings of Web Intelligence WI 2005, pp. 94–100. IEEE Computer Society, Los Alamitos (2005)
10. Buitelaar, P., Eigner, T.: Topic extraction from scientific literature for competency management. In: Proceedings of the Workshop Personal Identification and Collaborations: Knowledge Mediation and Extraction of the International Semantic Web Conference (2008)
11. Cimiano, P., Pivk, A., Schmidt-Thieme, L., Staab, S.: Learning taxonomic relations from heterogeneous sources of evidence. In: Ontology Learning from Text: Methods, Evaluation and Applications (2005)
12. Hearst, M.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: Proceedings of the 14th International Conference on Computational Linguistics, pp. 539–545 (1992)
13. Becerra-Fernandez, I.: Facilitating the online search of experts at nasa using expert seeker people-finder. In: Proceedings of the 3rd International Conference on Practical Aspects of Knowledge Management (2000)
14. Paquette, G.: An ontology and a software framework for competency modeling and management competency. Educational Technology & Society 10, 1–21 (2007)

# Towards Trust in Web Content Using Semantic Web Technologies

Qi Gao

TU Delft, Web Information Systems, EWI/ST/WIS,
P.O. Box 5031, 2600 GA Delft, The Netherlands
q.gao@tudelft.nl

**Abstract.** Since the amount of user-generated content has been sharply increasing in recent years, mainly due to Web 2.0 technology and effects of social networking, it is necessary to build mechanisms to assess the reliability of the content. On the web this notion of trust is a key ingredient for an effective manipulation of knowledge on a (world-wide) web scale. The web of trust has thus become an important research area both for web science and semantic web. In the PhD research we have laid out for us, we focus on the notion of trust and methods for representing and computing trust of users in the web content. This paper outlines the vision at the start of the PhD period on the research problem and the semantic web-based approach to solve that problem.

**Keywords:** trust, web science, semantic web.

## 1 Problem Statement and Motivation

The volume of web content is sharply increasing in the recent years mainly because of Web 2.0 technology and applications for social networking, e.g. discussion boards in online communities, social network profiles, videos, photos, reviews and ratings of products, hotels, restaurants.

With the enormous and still growing amount of content on the Web, it becomes more and more necessary for users to be able to assess the trustworthiness of content on the Web. For example, it will help for information retrieval systems or recommender systems to find the most relevant and reliable content that matches the user's preference. Building a trust mechanism for the Web is also helpful to make the interaction between users and content more reliable and controllable.

Dealing with trust on the Web is a difficult and complex task, since it involves a wide range of factors such as user history, user preferences, demographical information and the context in which the trust relationship is built. These factors have very different characteristics and it is a challenge to measure these factors. So, it is necessary to create an approach for modeling and computing the trustworthiness of content on the Web for individual user.

Ideally, there would be a trust model that tells for each user and each content element how trustworthy that content element is for that user. A giant barrier for obtaining such a trust model is that if we would have obtained accurately the trustworthiness

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 457–461, 2010.

for a given set of users and content elements, it is very hard to predict the trustworthiness for new content.

Instead of focusing on the content, many approaches therefore choose to build the notion of trust and trust metrics within networks of people, agents or peers that have trust relationships between them [3]. Such trust networks are often obtained by taking explicit trust relationships between pairs of people firstly and then propagating the trust through the network. In a social environment, such a trust network can easily be acquired from social networks of friends or connected people. The trustworthiness of content that is unknown for a given user is then predicted in an indirect way by utilizing the trust relationships for that user to other people.

A major shortcoming with these approaches is that they cannot adequately deal with the semantics of the content and so we observe that the current approaches are limited to work only in a specific context or domain. For example, the trust relationship predicted using a friends network acquired from a website for books does not work effectively in the domain of movies, even for the same people.

Since it is now feasible to know more of the content's semantics, exploiting explicit and implicit representations with semantic web technology and using access to linked open data, we see the opportunity to build a better global trust metric. In our approach, each user maintains the trustworthiness for a set of content elements. Instead of propagating this through the network of users, we directly predict the trustworthiness of new, unknown content by analyzing the semantic relationship between these two pieces of content. So, by using a more explicit representation of what we know about the concept and their relationships, we make it possible for the users to get a better grip on their trust in content and not only in the people behind the content.

This PhD work will propose a model for trust of users in Web content in order to make the user-oriented exploitation of data on the Web more reliable and controllable. In the model we will represent trust with semantic standards and utilize the analysis of semantic relations between content elements for the trust computation and inference.

## 2 State of the Art

In this section, we briefly discuss the state-of-the-art of approaches that are related to our work. The Web of trust [4] has become an important area both for academia and industry. Many trust metrics have been developed. Each approach stresses various characteristics of trust. Artz and Gil [1] demonstrated a comprehensive overview of existing trust metrics in computer science and semantic web. Due to the lack of space we focus on two tasks that are most relevant to our problem and motivation.

### 2.1 Representing Trust

One category of approaches to representing trust on the Web is based on the concept of a Web of Trust that is a network of people, agents or peers with trust relationships. Golbeck [3] represented trust within web-based social networks by defining the functional properties of trust. In Ray's paper [12], trust is specified as a relationship between a truster and a trustee at a particular time instance and for a particular context. Heath and Motta [8] provided a way to collect reviews and ratings data from distributed sources and generated a metric to represent trust relationships, which is used as

an input to a Web-based system with the purpose of information seeking and recommendation. These approaches focus on the trust among users on the Web.

In contrast to these approaches, some work has been done related to trust in the content itself. Gil et al. [2] used the term of "content trust" to acquire trust in the content provided by a web resource. Rich factors are included in their model. However, their work did not provide a mechanism to deal with the semantic of the content. Groth et al. [5] proposed a content-based trust assessment in electronic contracts based on similarity of content in different contracts. Hartig et al. [7] presented a trust model for RDF-based content for representing the trustworthiness of the data published on the Web. But these approaches do not provide a sufficient mechanism to represent the trustworthiness of certain content for individual user.

### 2.2 Computing Trust

In computer science, Marsh [10] was the first one to analyze trust as a computational concept. Guha et al. [6] proposed a method to propagate both trust and distrust through social networks. In Ziegler's work [13], a trust algorithm called Appleseed has been developed. It employs a spreading activation model that is originally from psychology, to produce a trust ranking of individuals in the network. TidalTrust [3] is another algorithm that performs a breadth-first search, using trust values within the social network. In a trust network, it calculates trust values for a sink node with a weighted average of the source's neighbors' ratings of the sink. Massa [11] proposed a trust propagation metric of searching for trustworthy users over the trust network to alleviate the cold start problem in a collaborative filtering (CF) algorithm.

## 3 Approach

The previous section addresses the lack of a uniform model for representation and computation of trust. We propose an approach that concentrates on the following research issues:

- How to represent a trust relationship between a user and web content?
- How to start to assess the trust value in that trust relationship?
- How to further compute trust by investigating the semantic relation between different pieces of content?

We discuss these issues now in more detail. First of all, our approach requires a representation model of trust that defines and represents a trust relationship between a user and content on the Web. There are three critical elements to be investigated in our representation model: the trust value that is a rating to certain content from a certain user, the semantics of the content, and the context. The trust model will be formalized using RDF-based vocabularies.

Based on this approach of trust representation, in order to boot the computation process of trust, we first need to acquire initial trust values. Basically there are two solutions: providing the trust values: manually by users and generating them automatically by utilizing existing data. For the first one, it is very difficult to choose a set of Web content as a sample for rating when our task is not domain specific. Another

disadvantage is that sometimes it is not practical to push the users to give these trust ratings. Since there are already many data sources that include the user's ratings and reviews on the Web content and more and more web content is enriched with semantic data, we adopt the latter one as our solution to start assessing the trust. Here we propose two kinds of automatic trust assessment methods: user-based rating methods that integrate various data of ratings and reviews; and semantic-based methods that consider meta-information such as provenance to assign the initial trust values.

The third task is the further computation and inference of trust, of which the main goal is to predict a trust value of web content that the user never rated or obtained before. Comparing the propagation method we mentioned in section 2, we reach this goal by analyzing the semantics of the relationship between two pieces of content, of which one already has a trust value and the other one not yet. Given the semantically annotated and linked data, it is possible to reveal the semantic relationship between the content. To illustrate, we give a simple example. Using "RelFinder"[9], a semantic relation discovery tool for DBpedia, we can find the relationship between two entities in DBpedia: Johannes Vermeer[1] and Night Watch[2]. By observing the graph of paths that represent the relationship, we found six different paths, each of which connects these two entities by chaining several other entities from the knowledge base (DBpedia). The discovered entities located in each path can demonstrate the specific semantics of the relationship that connects these two requested entities. For example, they can be geographically connected because of the entities "Netherlands" and "Amsterdam" in one path or connected by artistic characteristics because of the entity "Baroque painting" in another path. After matching these semantics and context information such as user preferences to analyze which path is more relevant to the trust computation in the specific context, a new trust value can be inferred.

## 4 Methodology

Our methodology for solving the above questions has three main phases. In the first phase, we will build the representation model for trust, which will take the context into account. Based on the model, a trust vocabulary will be developed to enable the description and access of trust information. In the second phase, based on the representation model and the approach we discussed in the previous section, we will process the computation of trust. For the first implementation, we limit the web content to structured data, e.g. DBpedia or DBLP, or some web pages, e.g. some movie list page, which can be easily transferred to structured data. We will develop a browser plug-in to recommend a trust value when the user browses the web content. The third phase will include the evaluation of our approach. Because of the nature of trust, we will conduct several user studies to evaluate our representation and computation methods. Furthermore, due to the semantic web technologies that we are using, we will consider uncertainty as a key factor in the evaluation.

[1] http://dbpedia.org/ page/Johannes_Vermeer
[2] http://dbpedia.org/page/ Night_Watch_%28painting%29

## 5 Conclusion

We identified the lack of a suitable model for trust between user and content on the Web. This PhD symposium paper sketched our approach that addresses these two main problems: we provide a mechanism for an individual user to represent his/her trust in certain web content and to assess trust values by collecting data from distributed sources. We also propose to compute the trust by utilizing the semantics of the relationship between the various content elements. With our work we hope to make the user-centric data exploitation on the web more reliable, efficient and controllable than it is today. For the first phase of the PhD work, according to the approach and methodology we proposed, the next step is to formalize trust in our scenario. We will build a representation model and an RDF-based vocabulary based on which the further actions can be pursued.

## References

1. Artz, D., Gil, Y.: A Survey of Trust in Computer Science and the Semantic Web. J. Web Semantic 5(2), 58–71 (2007)
2. Gil, Y., Artz, D.: Towards Content Trust of Web Resources. In: WWW 2004, pp. 565–574. ACM, New York (2004)
3. Golbeck, J.A.: Computing and Applying Trust in Web-based Social Networks. PhD thesis, University of Maryland (2005)
4. Golbeck, J.A., Parsia, B., Hendler, J.: Trust Networks on the Semantic Web. In: Proceedings of Cooperative Intelligent (2003)
5. Groth, P., Miles, S., Modgil, S., et al.: Determining Trustworthiness of New Electronic Contracts. In: Engineering Societies in the Agents World X, pp. 132–147 (2009)
6. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of Trust and Distrust. In: WWW 2004, pp. 403–412. ACM, New York (2004)
7. Hartig, O.: Querying Trust in RDF Data with tSparql. 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. 5–20. Springer, Heidelberg (2009)
8. Heath, T., Motta, E.: Revyu: Linking Reviews and Ratings into the Web of Data. J. Web Semantic 6(4), 266–273 (2008)
9. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: Revealing Relationships in RDF Knowledge Bases. In: SAMT 2009, pp. 182–187 (2009)
10. Marsh, S.P.: Formalising Trust as a Computational Concept. PhD thesis (1994)
11. Massa, P.: Trust Metrics in Recommender Systems. In: Golbeck, J.A. (ed.) Computing with Social Trust, pp. 259–285. Springer, London (2009)
12. Ray, I., Ray, I., Chakraborty, S.: An Interoperable Context Sensitive Model of Trust. J. Intell. Inf. Syst. 32(1), 75–104 (2009)
13. Ziegler, C.N., Lausen, G.: Propagation Models for Trust and Distrust in Social Networks. Information Systems Frontiers 7, 337–358 (2005)

# The Semantic Gap of Formalized Meaning

Sebastian Hellmann

AKSW Research Group,
Universität Leipzig, Germany
hellmann@informatik.uni-leipzig.de
http://aksw.org

**Abstract.** Recent work in Ontology learning and Text mining has mainly focused on engineering methods to solve practical problem. In this thesis, we investigate methods that can substantially improve a wide range of existing approaches by minimizing the underlying problem: The Semantic Gap between formalized meaning and human cognition. We deploy OWL as a Meaning Representation Language and create a unified model, which combines existing NLP methods with Linguistic knowledge and aggregates disambiguated background knowledge from the Web of Data. The presented methodology here allows to study and evaluate the capabilities of such aggregated knowledge to improve the efficiency of methods in NLP and Ontology learning.

## 1 Problem

Decades of research have been spent on answering the question "How can we teach machines to understand natural language?" and the unanimous response is: It is impossible (for obvious reasons). Any approach that indulges in formalizing the meaning of natural language faces the same problem: the Semantic Gap between formalized meaning and human cognition. So instead, we should modify the question and ask: "How big is the current Semantic Gap?", "How can we reduce it?" and "How can we measure the reduction?". We argue that, if we choose OWL as a unifying Meaning Representation Language (MRL), we are in a position to utilize several advantages (interoperability, expressiveness, available linguistic ontologies, available structured knowledge from the Web of Data, mature tool support) not only to reduce the Semantic Gap, but also to define processes for potential reductions, combined with an evaluation methodology.

## 2 State of the Art

Currently, work towards standardization of linguistic annotations is in progress (see work by Nancy Ide). Wintner (2009)[8] argues that interest in grounding NLP in Linguistic theories has been declining for two decades with the focus now being on "engineering solutions to practical problems" and corpora "as our source of (implicit) knowledge". A claim that can be verified by looking, for example, at methods included in two recent surveys of Ontology learning from

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 462–466, 2010.

text by Buitelaar et al.(2005)[1] and Zhou (2007)[9]. Parallel to these trends, a large number of mature linguistic ontologies has emerged formalizing knowledge about linguistic features. A representative example are the Ontologies of Linguistic Annotations ([2], OLiA), an architecture of modular OWL-DL ontologies that formalizes several intermediate steps of the mapping between concrete annotations (such as POS tags) and a Reference Model. While such ontologies exist, the question arises in which ways they can be exploited besides their intended purpose of unifying annotation schemes. One key to knowledge acquisition is clearly a deep linguistic analysis: by using OWL to aggregate knowledge on all levels of Linguistics in a single model and in a formal way, we are able to create a powerful preprocessing step, which can potentially improve a range of current methods.

The advantages become obvious, if we look, for example, at a recent approach by Völker et al. [7], which applies transformational rules to acquire OWL axioms from text. Völker et al. have a three-page long discussion about necessary linguistic features to improve their approach. As an enhancement, the conditional part of these proposed rules can, however, be directly replaced by expressive reasoner queries when employing our approach.

In his keynote talk at the ISWC 2009[1], Tom Mitchell presented an approach to extract facts from the Web with the help of a bootstrap ontology. He mentioned several shortcomings such as missing morphological features (why not more?) and incorporation of existing background knowledge (such as DBpedia[2]).

In this thesis, we will investigate the above-mentioned limitations as a whole, i.e. not just engineering solutions to specific tasks. By combining existing NLP methods with linguistic knowledge in OWL and adding disambiguated background knowledge from the Web of Data, we argue that we can shed light on the underlying problem of knowledge acquisition: The Semantic Gap of Formalized Meaning. Such an integrated approach has several advantages: errors in one layer can be corrected by other layers and external knowledge and also new OWL axioms can be induced based on given positive and negative sentences (see Section 5, where we learned the definition for a passive sentence.)

## 3 Proposed Approach

We begin with converting natural language text (a character sequence with implicit knowledge) into a more expressive formalism, in this case OWL, to grasp the underlying meaning. This explicated meaning then serves as input for (high-level) algorithms and applications (with a focus on machine learning). The central working thesis is the following:

*If features extracted by different NLP approaches (ranging from low-level morphology analysis to higher-level anaphora resolution) are explicated and combined with matching background knowledge (parser-ontology pair) in a model and if, additionally, this model is further enriched by fragments of existing knowledge*

[1] http://videolectures.net/iswc09_mitchell_ptsw
[2] http://dbpedia.org

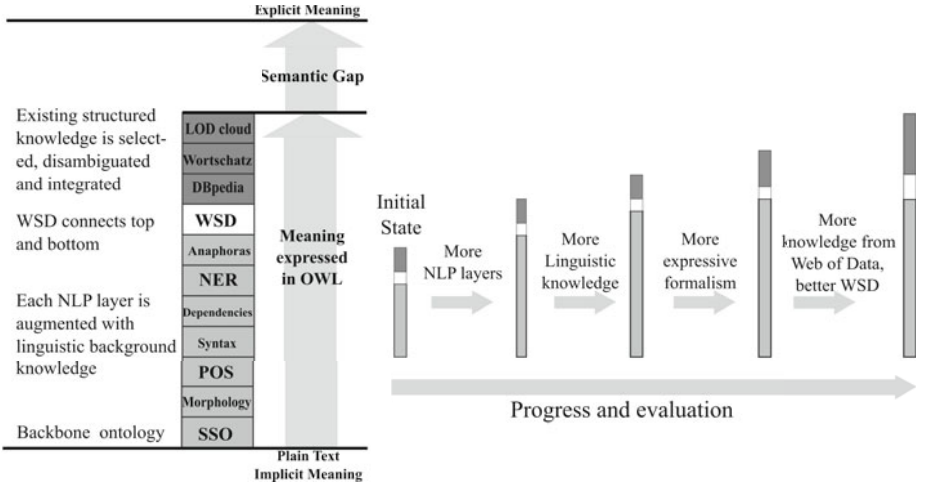



**Fig. 1.** Left: Display of the integration of NLP approaches and background knowledge connected via a Word Sense Disambiguation (WSD) component. Right: Actions which potentially lead to a measurable increase of the stack (cf. Section 4 for evaluation).

*bases from external sources such as DBpedia, it will be possible to reduce the Semantic Gap and improve performance on common knowledge acquisition tasks such as Ontology learning and Text understanding.*

Figure 1 gives an overview of the conversion approach in form of a stack (on the left side). In a first step, sentences are tokenized and aggregated in a *Structured Sentence ontology (SSO)*, consisting of a minimal vocabulary to denote the basic structure of the sentence such as tokens and relative position of a token in a sentence. The SSO (bottom) serves as the backbone model, which will be augmented by (1) features from NLP approaches (in light gray), (2) rich linguistic ontologies for these features, (3) background knowledge from the Web of Data (in dark gray) and, finally (4) knowledge which can be derived or inferred and which improves and corrects steps 1-3.

In most cases, output generated by NLP methods can be modeled in RDF in a straightforward way (e.g. POS-Tags are connected to tokens, dependencies are connections between tokens). An increasing number of linguistic ontologies already exist for certain NLP tasks (see [2] and Section 5 ) and serve as valuable addition to parser output (forming a *parser-ontology pair*). Both types of information aggregated from several parsers and integrated into the backbone model represent the basis for selecting background knowledge from the Web of Data. Fragments [3] of DBpedia, for example, can be retrieved on the basis of disambiguated entities as seed nodes and added to the model.

Preliminarily, we will define the Semantic Gap in a negative way, i.e. meaning that is not covered by the current stack. The stack in Figure 1 can be increased by the four measures depicted on the right side. We created a reference implementation NLP2RDF[3], which outputs an aggregated OWL ontology.

[3] Available as open-source at `http://code.google.com/p/nlp2rdf`

## 4 Methodology

Based on the formulation of the working thesis, progress will be driven by rigid evaluation. Although, the reduction of the Semantic Gap can not be conceived directly, the growth of the stack can be measured very well. We will collect a list of suitable machine learning approaches and knowledge acquisition tasks which normally depend on some form of NLP methods for preprocessing. Then, we will use the DL-Learner[5], a supervised concept learner, to repeat experiments and compare existing methods. As the expressiveness and availability of background knowledge directly affects the learning capability, we deem it an ideal evaluation setting because we can study the influence of available knowledge on the performance of certain tasks (e.g. relation extraction). In this way a benchmark suite can be created in which we can vary parsers or deactivate the inclusion of ontologies. This allows to evaluate the conditions under which the aggregated model was created and can thus measure potential improvements of the NLP2RDF stack in Figure 1 (right side).

Initially, we plan to participate in SemEval 2010[4] Tasks 4 (VP Ellipsis - Detection and Resolution) and 7 (Argument Selection and Coercion). Task 4 is similar to already achieved results on learning a concept for passive sentences presented in the next Section. Furthermore, we assume that the key to task 7 is a mixture of DBpedia and syntactical features. Likewise, the Reuters[5] benchmark is interesting, as we argue that text classification methods can be improved the more a machine can grasp the meaning of the text.

As parsers are error-prone, we will deploy several parsers at the same time and experiment with probabilistic logics for correction.

## 5 Results

The output of Stanford Parser[6] for POS-tagging and dependency parsing was converted to RDF and mapped to the Stuttgart-Tübinger Tagset ontology[7]. Based on 20 passive and 20 active German sentences from the Negra Corpus, the following concept was learned by DL-Learner: "$(Sentence \sqcap \exists hasToken.(VVPP \sqcap \exists previousToken.(APPR \sqcup VAFIN)))$" signifies a sentence with a 'past participle' (VVPP) preceded by 'temporal, causal, modal and local prepositions' (APPR) or 'finite auxiliary verbs' (VAFIN) – a precise description (for German passive), which can be understood as the intentional meaning of a class PassiveSentence and used to classify new sentences. Given a proper background corpus, it is possible to create expressive concepts just by selecting sentences. This scenario was realized in the TIGER Corpus Navigator[8], which achieved

[4] http://semeval2.fbk.eu/semeval2.php
[5] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[6] http://nlp.stanford.edu/software/lex-parser.shtml
[7] http://nachhalt.sfb632.uni-potsdam.de/owl/stts.owl by Chiarcos [2]
[8] http://tigernavigator.nlp2rdf.org

a high F-measure (above 86%) in an Active Learning approach with only few training sentences [4].

We assume that it is even possible to learn such concepts for semantic relations – just by selecting sentences – after sufficient concepts have been created at the syntactic level.

At the time of writing the work on this project is still in an early phase. Adapters to several other NLP tools have been implemented and evaluation has just begun. Furthermore, the Wortschatz[9] (a source for statistical data about sentences acquired from the web) was converted to RDF and mapped to DBpedia[6]. NER methods, a prerequisite for relation extraction, will be improved by linking to instances from DBpedia and including parts of the hierarchy.

## 6 Conclusions and Future Work

Planned contributions include: (1) analyzing and pinpointing the underlying problem in Text mining and Ontology learning, (2) theoretical research to acquire a clear definition of the Semantic Gap, (3) leveraging existing (statistical) NLP approaches with an ontology mapping (parser-ontology pair), (4) instrumentalization of background knowledge from the Web of Data (especially DBpedia and Wortschatz), (5) acquisition of additional linguistic knowledge by supervised machine learning, (6) evaluation-driven methodology and (7) creation of an open-source tool that can be used to improve existing solutions.

## References

1. Buitelaar, P., Cimiano, P., Magnini, B.: Ontology Learning from Text: An Overview, July 2005, vol. 123. IOS Press, Amsterdam (2005)
2. Chiarcos, C.: An ontology of linguistic annotations. LDV Forum 23(1), 1–16 (2008)
3. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL class descriptions on very large knowledge bases. IJSWIS 5(2), 25–48 (2009)
4. Hellmann, S., Unbehauen, J., Chiarcos, C., Ngonga, A.: The TIGER Corpus Navigator. Submitted to ACL system demonstrations (2010)
5. Lehmann, J.: DL-Learner: Learning Concepts in Description Logics. Journal of Machine Learning Research (JMLR) 10, 2639–2642 (2009)
6. Quasthoff, M., Hellmann, S., Höffner, K.: Standardized multilingual language resources for the web of data. In: 3rd prize at the LOD Triplification Challenge, Graz (2009), http://corpora.uni-leipzig.de/rdf
7. Völker, J., Hitzler, P., Cimiano, P.: Acquisition of OWL DL axioms from lexical resources. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 670–685. Springer, Heidelberg (2007)
8. Wintner, S.: What science underlies natural language engineering? Comput. Linguist. 35(4), 641–644 (2009)
9. Zhou, L.: Ontology learning: state of the art and open issues. Information Technology and Management 8(3), 241–252 (2007)

[9] http://corpora.informatik.uni-leipzig.de

# A Contextualized Knowledge Framework for Semantic Web

Mathew Joseph[1,2]

[1] FBK-IRST, Via Sommarive 18, 38050 Trento, Italy
[2] DISI, University of Trento, Italy
mathew@fbk.eu

**Abstract.** This thesis focuses on developing an efficient framework for contextualized knowledge representation on Semantic Web. We point out the drawbacks of existing formalism for contexts that hinder an efficient implementation and propose a context formalism that enables the development of a framework with desired properties. Some of the future milestones for this thesis work are to (i) develop a proof theory for the logical framework based on Description Logics (DL) (ii) develop reasoning algorithms (iii) verify and compare the performance of these algorithms to existing distributed reasoning formalisms and (iv) implement the system.

## 1 Problem, Motivation and Related Work

Contexts[1] have been appreciated for their ability to use the "Divide and Conquer" strategy for problem solving. They helps to localize reasoning[2] and search tasks, allow us to represent conflicting information by separating them in different contexts. Moreover they are a potential hope for the current state-of-the-art semantic web reasoners and ontology development tools for handling today's large Knowledge Bases (KB) by allowing to partition these KBs to distributed smaller ones.

In spite of past initiatives for contextualized knowledge representation[3,4] not enough focus has been applied on producing a computationally tractable framework. As far we know not a single approach has been implemented to demonstrate it's fruitfulness on the semantic web.

Also the initiatives in the past to provide a logical framework of contexts, many have been propositional in nature[5,6,7] and the other expressive versions based on *quantificational logic* such as the one given in [3] do not provide a sound and complete proof theory. [8] is based on the assumptions such as *rigid identifier property*[1], *barcan formula* compliance[2] and does not have a decidable logic. Also complications due to arbitrary nesting of contexts and lack of efficient decision procedures are some of the reasons for unsuitability of these frameworks on the Semantic web.

Some of the inherent problems in the works done so far is that contexts have been treated as first-class objects [3,8], allowing to use any fragment of first order logic on these objects. But how a relation $R(C_1, C_2)$ between two context objects $C_1$ and $C_2$ reflects reasoning in the knowledge part of these contexts is unspecified.

[1] Same URI denotes the same entity every where.
[2] Interpretation domains are the same for every context.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 467–471, 2010.

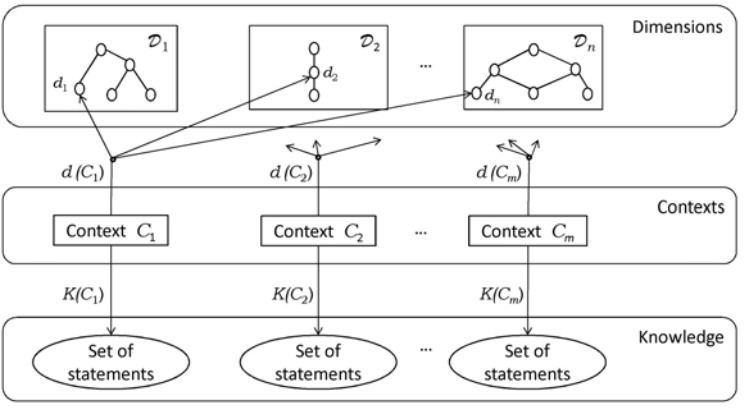



**Fig. 1.** Contextualized knowledge repository

## 2 Proposed Approach

A schematic diagram for a Contextualized knowledge repository is given in Fig. 1. Inspired by [2,9], we also comply to the fact that contexts have a logical theory and a fixed set of dimension values. These dimension values (time, location, topic etc.) can be used to qualify the contexts and are thus meta-knowledge for the contexts, hence the logical formalism employed for representation should allow for this meta-knowledge description and use this description for reasoning.

For reasons of tractability and applicability of our logical framework on the semantic web we ground our approach on OWL2 DL. We suppose that this can be extended to full first order logic. We arrive at the following definitions:

**Definition 1 (Contextual dimension).** *A context dimension, $\mathcal{D}$ is an OWL2 knowledge base whose signature $\Sigma(\mathcal{D})$ contains a set of constant symbols, $D$, called* dimension values*, and a strict partially ordered binary relation $\prec$ called* coverage relation*, which is the only relation present*[3]

**Definition 2 (Context).** *Given a set of $n$ dimensions $\{\mathcal{D}_i\}_{i=1..n}$, and a signature $\Sigma$, a* context *$\mathcal{C}$ in the dimensions $\{\mathcal{D}_i\}_{i=1..n}$ is a pair $\langle \boldsymbol{d}(\mathcal{C}), \mathrm{K}(\mathcal{C})\rangle$, where*

1. *$\boldsymbol{d}(\mathcal{C}) = \langle d_1, \ldots, d_n\rangle \in D_1 \times \ldots \times D_n$;*
2. *$\mathrm{K}(\mathcal{C})$ is an OWL2 knowledge base on the signature $\Sigma$;*

We define the knowledge about the context objects in *context description space* and restrict it's logic to contain only one predicate $\prec$ with a well defined semantics to relate any two context objects[4]. This is because we currently restrict our focus on qualifying context objects with dimensions – location, time and topic, where we use the $\prec$ to

[3] Strict partially ordered relations can be represented in OWL2.
[4] This can be extended to a fixed number of predicates with well defined semantics.

represent geographic containment, temporal containment and topic coverage relations respectively[5].

**Definition 3 (Context Description Space).** *A* context description space*, $\mathfrak{K}$ is an OWL2 knowledge base whose signature $\Sigma(\mathfrak{K})$ contains a set of* context objects *$\mathcal{C}_i$, a set of functional symbols $\mathcal{D}_i$-Of(.), the binary relation $\prec$ called* context cover relation*, which is the only binary relation on context objects.*

Intuitively the $\mathcal{D}_i$-Of(.) is a function to associate a context $\mathcal{C}$ to value of $\mathcal{D}_i$, *context-cover* relation $\prec$ is such that for any two contexts $\mathcal{C}_a$ and $\mathcal{C}_b$, $\mathcal{C}_a \prec \mathcal{C}_b$ iff $\mathcal{D}_i$-Of$(\mathcal{C}_a) \prec_i \mathcal{D}_i$-Of$(\mathcal{C}_b)$ for $i = 1, ..., n$.

**Definition 4 (Contextualized Knowledge Base).** *Given a signature $\Sigma$, a contextualized knowledge base (CKB) is a tuple $\mathfrak{K} = (\mathfrak{D}, \mathfrak{K}, \mathfrak{C})$ where*

1. *$\mathfrak{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n \rangle$ is an $n$-tuple of context dimensions*
2. *$\mathfrak{K}$ is the outer context*
3. *$\mathfrak{C}$ is a set of contexts on dimensions $\mathcal{D}_1, \ldots, \mathcal{D}_n$*

The relation $\prec$ between any two context objects $C_1$ and $C_2$ describe the relation between domains of these contexts. For example if $C_1 \prec C_2$ then, (i) domain of $C_1$ is contained in the domain of $C_2$ (ii) identifiers in $C_1$ and $C_2$ are rigid. We strongly identify the need of such a relation $\prec$ with such special semantics for contexts, because in the semantic web we are normally dealing with URIs and these normally retain their meaning across contexts.

Also note the use of meta-knowledge in our framework, such meta-knowledge values such as time, location, topic or provenance can reveal significant facts about contexts. For example if it is the case that all the dimension values for two Contexts $C_1$ and $C_2$ are respectively equal, then $C_1 \prec C_2$ and $C_2 \prec C_1$, this means that information in these contexts are highly coherent and there can be much greater extent of lifting of axioms and migration of individuals between them.

*Example 1.* Suppose $C_{greece}$ and $C_{europe}$ are two contexts in our context description space and $C_{greece} \prec C_{europe}$, then for a formula such as

$$C_{greece} : \mathsf{Conference}(\mathsf{eswc\text{-}2010})$$

it is also the case that

$$C_{europe} : \mathsf{Conference}_{greece}(\mathsf{eswc\text{-}2010})$$

In this case the assumption is that domain of discourse of $C_{greece}$ is covered by that of $C_{europe}$. The constant symbol eswc-2010 means the same across these contexts and extension of concepts Conference and $\mathsf{Conference}_{greece}$ are equal.such lifting of axioms across contexts can be taken care by rules of the system such as

$$\frac{\mathcal{C} : \Phi \quad \mathfrak{K} : \mathcal{C} \prec \mathcal{C}'}{\mathcal{C}' : \Phi'}$$

where $\mathfrak{K}$ is the context description space and $\Phi'$ and $\Phi$ are formulas

[5] This is a philosophically debatable assumption, but in general setting a fixed set of dimensions can completely qualify the context space[9].

## 3 Methodology

**Problem definition.** The problem is to develop a usable framework for context dependent knowledge representation for the semantic web. The framework developed should support the growing semantic web needs like distributed reasoning, tractability, scalability, reasoning on meta-knowledge for deriving facts in the knowledge.

**Survey of existing approaches.** A description of alternate approaches on contexts that has been considered is in Section 1. We also consider existing works on distributed reasoning like DDL[10,11] and collaborate with their inventors for reusing existing results and implementations. Some of the shortcomings of these existing frameworks and the ones in[12,13], are that they do not take into consideration, the use of meta-knowledge. Although approaches like [14] does reasoning on meta-knowledge, their semantics do not suffice to relate two distinct pieces of knowledge and how axioms can be lifted across them.

**Design of Initial framework and Prototype Implementation.** A brief description of the designed framework has been given in Section 2. We are currently working on specifications of syntax and semantics of our framework. The current problem we are addressing is the semantics of *lifting rules*[6]. Currently we limit our selves to simple syntax of lifting rules. A Prototype Implementation has been constructed for testing and for running future experiments.

**Development of proof theory and reasoning algorithms.** The next immediate step to be considered is development of a proof theory that is sound and complete with respect to semantics employed. For every DL axiom $\Phi$, We have an associated context $C$, and hence we denote this fact by $C : \Phi$. Then we need to consider development of sound and complete reasoning algorithms.

**System Implementation and Testing.** Proposed framework once designed need to be implemented to test it's ability of applicability on semantic web, verify correctness of implemented reasoning algorithms.

## 4 Results and Status

We have implemented a first prototype of a contextualized knowledge repository, by restricting the expressivity of data in the repository to RDF(S).[7] Such a restriction is explained by the scalability issues of available tools. Practically, we grounded our prototype on Sesame RDF(S) storage and querying framework[8]. We also demonstrated with this prototype application how search efficiency and accuracy on semantic web can be benefited with our notion of contexts[15]. In this work we demonstrated the use of context dimensions to form a hierarchical structure of contexts, which can guide the order of search. Another application of contexts for semantic enrichment was demonstrated in [16]. We are currently in the phase of development of the required proof theory.[9]

[6] The rules that relate information in different contexts.

[7] The prototype is available at `https://dkm.fbk.eu/index.php/CKR`

[8] http://www.openrdf.org/

[9] I am a first year PhD Student.

## 5 Conclusions and Future Work

This paper introduced the current problems of semantic web, that motivates the idea of a contextualized knowledge representation framework. We explained the related work done before on contexts. We introduced our approach (section 2) as a solution and stated the methodology of research adopted during this thesis (section 3) and results obtained.

Some of the future milestones to be achieved for this thesis work are (i) develop a proof theory for the logical framework based on Description Logics (DL) (ii) developing reasoning algorithms (iii) verify and compare the performance of these algorithms to existing distributed reasoning formalisms and (iv) system implementation.

## References


1. McCarthy, J.: Generality in Artificial Intelligence. Communications of the ACM 30(12), 1029–1035 (1987)
2. Benerecetti, M., Bouquet, P., Ghidini, C.: Contextual Reasoning Distilled. Experimental and Theoretical AI 12(3), 279–305 (2000)
3. Guha, R.: Contexts: a Formalization and some Applications. PhD thesis, Stanford (1992)
4. Guha, R., Mccool, R., Fikes, R.: Contexts for the semantic web. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 32–46. Springer, Heidelberg (2004)
5. Buvac, S., Mason, I.A.: Propositional logic of context. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 412–419 (1993)
6. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics, or: how we can do without modal logics. Artificial Intelligence 65(1), 29–70 (1994)
7. Ghidini, C., Giunchiglia, F.: Local models semantics, or contextual reasoning = locality + compatibility. Artificial Intelligence 127 (2001)
8. Buvac, S.: Quantificational logic of context. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 412–419 (1996)
9. Lenat, D.: The Dimensions of Context Space. Tech. Rep., CYCorp (1998), http://www.cyc.com/doc/context-space.pdf
10. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. J. Data Semantics 1, 153–184 (2003)
11. Serafini, L., Tamilin, A.: Drago: Distributed reasoning architecture for the semantic web. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 361–376. Springer, Heidelberg (2005)
12. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., Stuckenschmidt, H.: C-owl: Contextualizing ontologies. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 164–179. Springer, Heidelberg (2003)
13. Kutz, O., Lutz, C., Wolter, F., Zakharyaschev, M.: E-connections of abstract description systems. Artificial Intelligence 156(1), 1–73 (2004)
14. Schenk, S., Dividino, R.Q., Staab, S.: Reasoning with provenance, trust and all that other meta knowlege in owl. In: SWPM (2008)
15. Joseph, M., Serafini, L., Tamilin, A.: Context shifting for effective search over large knowledge bases. In: Workshop on Context, Information And Ontologies (CIAO 2009), collocated with the 6th European Semantic Web Conference, ESWC 2009 (2009)
16. Tamilin, A., Magnini, B., Serafini, L., Girardi, C., Joseph, M., Zanoli, R.: Context-driven semantic enrichment of italian news archive. In: ESWC 2010 (2010) (accepted for publication)

# Computational and Crowdsourcing Methods for Extracting Ontological Structure from Folksonomy

Huairen Lin and Joseph Davis

Knowledge Discovery and Management Research Group,
School of IT, The University of Sydney, NSW 2006, Australia
{lin,jdavis}@it.usyd.edu.au

**Abstract.** This paper investigates the unification of folksonomies and ontologies in such a way that the resulting structures can better support exploration and search on the World Wide Web. First, an integrated computational method is employed to extract the ontological structures from folksonomies. It exploits the power of low support association rule mining supplemented by an upper ontology such as WordNet. Promising results have been obtained from experiments using tag datasets from Flickr and Citeulike. Next, a crowdsourcing method is introduced to channel online users' search efforts to help evolve the extracted ontology.

## 1 Introduction

Social tagging systems, such as Flickr[1] , Delicious[2], have recently emerged as some of the rapidly growing web 2.0 applications. The product of this kind of informal social classification structure, also known as folksonomy, has provided a convenient way that allows online users to collectively annotate and categorize large number of distributed resources from their own perspectives. However, as the amount of resources annotated using folksonomy has increased significantly, exploration and retrieval of the annotated resources poses challenges due to its flat and non-hierarchical structure with unsupervised vocabularies.

At the same time, the development of semantic web is creating a cyberspace that contains resources with relations among each other and well-defined machine-readable meaning. In this vision, ontology is the enabling technology for most of the semantic applications, such as semantic search. However, there are significant challenges to be overcome before we can build tools for sophisticated semantic search. It is not easy to establish a single and unified ontology as a semantic backbone for a large number of distributed web resources, and manual annotation of resources requires skilled professionals or ontology engineers [1]. Furthermore, ontology needs to be constantly maintained to adapt the knowledge emerging from daily work of users [2].

[1] flickr.com Flickr is an online photo management and sharing application.
[2] delicious.com Delicious is a social bookmarking service.

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 472–477, 2010.

The goals of our research is to extract ontological structure from folksonomy and to facilitate its automatic evolution with changing usage patterns, in such a way that the resulting structure can better support semantics-based browsing and searching of online resources. With the unification of the seemingly exclusive features of folksonomy and ontology, they can complement each other by providing full advantage of colloquial terms from folksonomy and semantic relations from ontology. We can exploit the semantic relation in the ontological structure to satisfy users' query or navigation requests using terms that they are familiar with in order to access millions of annotated resources, and translating and integrating the resources from different sources.

## 2 State of the Art

**Computational Approaches.** There are several promising techniques for extracting knowledge from the existing resources, such as hierarchical clustering [3], statistical model [4], and association rules mining [5]. Most of hierarchical clustering algorithms are based on bottom-up methods. First it computes pairwise tag similarities, and then merges most similar tags into groups. After that, pairs of groups are merged as one until all tags are in the same group [3]. Association rule mining has also been adopted to analyse and structure folksonomies. The output of association rule mining on a folksonomy dataset are association rules like $A \rightarrow B$, which implies that users assigning the tag A to some resources often tend to also assign the tag B to them [5].

To further discover the relationships within tags in clusters, several existing upper ontology resources can be used as references, such as WordNet and other semantic web resources. Ontology mapping and matching techniques are commonly applied to identify relationships between individual tags, between tags and lexical resources, and between tags and elements in an existing ontology. For example, by mapping "apple and fruit" in a food ontology, we can find the relation that "apple" is a subclass of "fruit" [6,7].

**Crowdsourcing Human Computation.** While the most sophisticated computational techniques cannot substitute the participation of knowledge engineers, the recently proposed crowdsourcing method provides new ways to have users engage in ontology engineering and to aggregate their deep knowledge through a mass collaboration technique [8,9]. A computer program that can attract human's interest, fulfill their needs, and collect, interpret human's solution is important. Ontogame [10] proposed a game for ontology building. One of the game scenarios is to asks users to check the structure and abstraction from random wiki pages. Recently, experiments show that online service such as freeware, or a successful login procedure [11] can also be used to motivate public users to participate in a specific task. With a purpose-designed system, we will be also able to embed the task of building and maintaining ontologies into users' everyday work process and create the conditions for the ontology to continuously evolve without the help of knowledge engineers [2]. In [12], a semantically enriched bookmarks navigation system provides functionality that enables users

to reject or accept the more general/narrow tags. These inputs were recorded for further ontology maintenance.

In summary, although several computational approaches have been proposed to bring structure to folksonomies, they do not come without limitations. These include the inability to decide the super/sub class relations of terms generated by association rule mining. Such problem can be partially solved by introducing an upper ontology such as WordNet. However, such an approach can only deal with the standard terms and has no effect on terms that do not appear in the upper ontology. Moreover, several attempts have shown that crowdsourcing human computation is promising method to bring non-experts together to tackle some difficult problems. These include problems like ontology refinement and evolution which normally need domain experts' participation.

## 3 Methodology

In this paper, our research concerns following specific research problems: (1) How to extract shared vocabularies from large folksonomy datasets? (2) How to find the semantic relations for these shared vocabularies? (3) How to handle the non-standard tags in the folksonomies? For instance, terms like 'folksonomy', 'ESWC' that cannot be found in traditional dictionary. (4) How can the resulting ontological structure be automatically evolved with the constant change of domain knowledge and patterns of usage?

We first propose an integrated computational approach to extract ontological structures. Our approach combines the knowledge extracted from folksonomies using data mining techniques with the relevant terms from an existing upper-level ontology. Specifically, low support association rule mining is used to analyze a large subset of a folksonomy. Knowledge is expressed in the form of new relationships and domain vocabularies. We further divided the tag word-formation into standard tag, compound tag and jargon tag and handle them respectively. Standard tags in the vocabulary are mapped to WordNet to get semantic relations. Jargon tags and user defined compounds are then incorporated into the hierarchy based on domain knowledge extracted from folksonomy. Thus, the hidden semantic knowledge embedded in the folksonomies is transformed into formalized knowledge in the form of ontological structures.

A semantic search assist is designed based on crowdsourcing model to update the extracted ontology for evolving folksnomy while it suggests helpful search terms and semantic relationships to help refine users' search. First, we elicit the inputs from users by providing terms semantically related with their query keywords and candidate semantic relationships such as 'is-a' after user conducts a normal search. With this assist, user can make explicit the semantic concept of what s/he is looking for by simply selecting the related term provided by ontology and assigning the relationship between the query keyword and related term. The semantic search engine will then return better result with a reasoning technology based on the disambiguated query. For example, by appointing 'apple' as 'is-a' kind of 'computer', the system will expand the results to more specific class such

as 'Mac' or a individual model such as 'MacBook Air' and remove results belong to 'fruit'. We then collect and aggregate these terms and relationships from different search sessions. Every user-assigned relationship is recorded even it is a disagreement with existing knowledge. The long-term records will be split into several clusters to reflect knowledge from different domains. We assume that the user specified semantic relationship is correct based on some aggregation mechanisms such as the rule of majority. After that, we introduce a mechanism to periodically merge changes with old version of ontology and release improved version. In short, we show how users' search intent can be captured to help to evolve the ontology while helping to improve their desired search results.

We attempt to engage web users in evaluation tasks using a crowdsourcing medium such as Amazon Mechanical Turk (MTurk)[3]. Based on this service, we ask users to manually evaluate the collected term pairs and give monetary award to every complete task. We will also attempt to verify the quality of the extracted ontology against other manually built gold standard ontology. The measurement should reflect how well the extracted terms cover the target domain and the accuracy of the relationships among the terms, especially for standard terms. Furthermore, we take the task-based evaluation approach to measure how far the extracted ontological structure will help to influence and improve the search result. We investigate four potential application scenarios of the extracted ontological structures: multi-dimensional views, cataloguing and indexing, query expansion and tagging suggestion. We will use those widely used measures such as precision, recall, and F-measure to assess the quality and see how ontology would improve the search result.

## 4 Preliminary Results and Future Work

We have implemented a prototype system for the described computational extraction strategy. The implementation and evaluation are reported in [13]. Through the investigation into four kinds of word-formations (standard tags, jargon tags, compound tags, and nonsense tags) in folksonomies, our approach has produced promising initial results using two datasets from Flickr and Citeulike.

To explore potential application scenarios with the resulting ontological structure, we are building a semantic photo organizing system, SmartFolks, based on a subset of Flickr image collection. With the extracted ontology as background knowledge and Jena[4] tool kit as semantic web framework, the system enables user to find the resources through the navigation of ontological structure or to get better results with the technology of semantic web such as ontology based query expansion. This demo site is available at http://smartFolks.thetag.org

Our future work is focusing on the ontology evolution using crowdsourcing method. A semantic search assist component will be integrated in the SmartFolks system to channel users' search efforts for ontology evolution.

[3] mturk.com MTurk is a web based service that enable developers outsource certain task to human across the world. The unit work typically costs only few cents.

[4] jena.sourceforge.net/ Jena is a framework for building Semantic Web applications.

## 5 Conclusion

Recent research indicates that there are significant challenges in the area of ontological structure extraction from collaborative tagging systems. The integrated framework proposed in this thesis allows a systematic approach to this emerging area. We have not only identified computational methods for ontology extraction, but also presented a proposal for crowd-sourcing model which is capable of aggregating the human intelligence without the need for the involvement of ontology experts. A semantic search engine is recommended as the medium for this integration. The application of this conceptual framework might assist folksonomy based systems to improve the query performance and enhance the organization of resources. It is hoped that the crowdsourcing approach will complement the computational methods to help create robust ontological resources that can advance the state-of-the-art with respect to semantic search.

## References

1. Wu, X., Zhang, L., Yu, Y.: Exploring social annotations for the semantic web. In: Proceeding of the 15th WWW Conference, Edinburgh, Scotland, pp. 417–426. ACM, New York (2006)
2. Braun, S., Schmidt, A., Walter, A.: Ontology maturing: a collaborative web 2.0 approach to ontology engineering, Banff, Canda (2007)
3. Wu, H., Zubair, M., Maly, K.: Harvesting social knowledge from folksonomies. In: The 7th Conference on Hypertext and Hypermedia, Odense, Denmark (2006)
4. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical report, InfoLab, Stanford (2006)
5. Schmitz, C., Hotho, A., Jaschke, R., Stumme, G.: Mining association rules in folksonomies. In: The 10th IFCS Conference, Studies in Classification, Data Analysis, and Knowledge Organization (2006)
6. Angeletou, S., Sabou, M., Motta, E.: Semantically enriching folksonomies with FLOR. In: CISWeb, p. 65 (2008)
7. Cattuto, C., Benz, D., Hotho, A., Stumme, G.: Semantic grounding of tag relatedness in social bookmarking systems. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 615–631. Springer, Heidelberg (2008)
8. Brabham, D.C.: Crowdsourcing as a model for problem solving: An introduction and cases. Convergence 14(1), 75 (2008)
9. Niepert, M., Buckner, C., Allen, C.: Working the crowd: Design principles and early lessons from the Social-Semantic web (2009)
10. Siorpaes, K., Hepp, M.: Ontogame: Towards overcoming the incentive bottleneck in ontology building. In: 3rd International IFIP Workshop (2007)
11. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: reCAPTCHA: Human-Based character recognition via web security measures. Science 321(5895), 1465 (2008)

12. Limpens, F., Gandon, F., Buffa, M.: Collaborative semantic structuring of folksonomies. In: 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 132–135. IEEE Computer Society, Los Alamitos (2009)
13. Lin, H., Davis, J., Zhou, Y.: An integrated approach to extracting ontological structures from folksonomies. 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. 654–668. Springer, Heidelberg (2009)

# Debugging the Missing Is-A Structure of Networked Ontologies

Qiang Liu and Patrick Lambrix

Linköpings universitet,
581 83 Linköping, Sweden

**Abstract.** In parallel with the proliferation of ontologies and their use in semantically-enabled applications, the issue of finding and dealing with defects in ontologies has become increasingly important. Current work mostly targets detecting and repairing semantic defects in ontologies. In our work, we focus on another kind of severe defects, modeling defects, which require domain knowledge to detect and resolve. In particular, we are interested in detecting and repairing the missing structural relations (is-a hierarchy) in the ontologies. Our goal is to develop a system, which allows a domain expert to detect and repair the structure of ontologies in a semi-automatic way.

## 1 Problem Statement

Developing ontologies is not an easy task and often the resulting ontologies are not consistent or complete. Such ontologies, although often useful, also lead to problems when used in semantically-enabled applications. Wrong conclusions may be derived or valid conclusions may be missed. To deal with this problem we may want to repair the ontologies. Until now most work has been performed on finding and repairing the semantic defects such as unsatisfiable concepts and inconsistent ontologies. In this work we tackle the problem of repairing modeling defects and in particular, the repairing of the structure of ontologies.

In addition to its importance for the correct modeling of a domain, the structural information (e.g. is-a or part-of) in ontologies is also important in semantically-enabled applications. For instance, the is-a structure is used in ontology-based search and annotation. It is also important information in ontology engineering research. For instance, most current ontology alignment systems use structure-based strategies to find mappings between the terms in different ontologies (e.g. overview in [11]) and the modeling defects in the structure of the ontologies have an important influence on the quality of the ontology alignment results [1].

As the ontologies grow in size, it is difficult to ensure the correctness and completeness of the structure of the ontologies. Some structural relations may be missing or some existing or derivable relations may be unintended. Detecting and resolving these defects requires, in contrast to semantic defects, the use of domain knowledge. One interesting kind of domain knowledge are the other ontologies and information about connections between these ontologies. For instance, in the case of the Anatomy track in the 2008 and 2009 Ontology Alignment Evaluation Initiative (OAEI) two ontologies,

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 478–482, 2010.

Adult Mouse Anatomy Dictionary [2] (MA, 2744 concepts) and the NCI Thesaurus - anatomy [15] (NCI-A, 3304 concepts), and a partial reference alignment (PRA, a set of correct mappings between the terms of the ontologies) containing 988 mappings are given. Using one ontology and the mappings as domain knowledge for the other ontology (and vice versa), it was shown in [10] that at least 121 is-a relations in MA and 83 in NCI-A are missing and should be repaired.

In our work, we deal with detecting and repairing the missing is-a structure in ontologies in the context of domain knowledge represented by networked ontologies. Our goal is to develop a system which allows a domain expert to debug and repair the structure of ontologies in a semi-automatic way.

## 2 State of the Art

There is not much work on detecting and repairing modeling defects in networked ontologies. In [3] and [9] similar strategies to detect missing is-a relations are described. Given two pairs of terms between two ontologies which are linked by the same kind of relationship, if the two terms in one ontology are linked by an is-a relation while the corresponding terms in the other are not, it is deemed as a possible missing is-a relation.

Related to the detection of missing relations, there is much work on finding relationships between terms in the text mining area. Much of the work on detecting is-a relations is based on the use of Hearst patterns [5] or extensions thereof (e.g. [4,17]). Most of these approaches have good precision, but low recall. A semi-automatic approach for ontology refinement (including is-a relations) is given in [18]. Another paradigm is based on machine learning and statistical methods, such as k-nearest neighbors approach [12], association rules [13], and clustering techniques [19].

There is more work that addresses semantic defects in ontologies. In [16] minimal sets of axioms are identified which need to be removed to turn an ontology coherent. In [8,7] strategies are described for repairing unsatisfiable concepts detected by reasoners, explanation of errors, ranking erroneous axioms, and generating repair plans. In [6] and [14] the setting is extended to repairing networked ontologies. In this case semantic defects may be introduced by integrating ontologies. Both approaches assume that ontologies are more reliable than the mappings and try to remove some of the mappings to restore consistency. The solutions are often based on the computation of minimal unsatisfiability-preserving sets or minimal conflict sets.

## 3 Approach and Methodology

**Detecting the missing is-a relations.** Given an ontology, a naive way to detect the missing is-a relations would be to take every pair of concepts which have no inferable is-a relation and check whether there should be an is-a relation between them. This requires inspection by domain experts and it usually means a large amount of work. However, as discussed before, other ontologies together with mappings between these ontologies can be used for detecting missing is-a relations. Our research aims to facilitate the detection of missing is-a relations in this way. In particular, we address the following questions:

(a) How to use other ontologies and PRAs to detect missing is-a relations?

(b) How to use other ontologies and given (possibly incorrect) mappings to detect missing is-a relations?

For the first question it is assumed that the given mappings are correct. The case where the structure of the ontologies is assumed to be correct and the mappings are 1-1 equivalence mappings, is relatively straightforward (e.g. [9]). However, this approach needs to be extended for other kinds of mappings. When we cannot assume that the structure of the ontologies is correct, we will only be able to compute suggestions for missing is-a relations and additional validation by a domain expert will be needed.

The second question adds the additional difficulty that the mappings are not necessarily correct (e.g. they are generated by an ontology alignment system and not yet validated). In this case we will need to also deal with the repairing of the mappings (semantic defects) such as in [6] and [14], and analyze the interaction between the modeling defects and the semantic defects.

**Repairing the missing is-a relations.** Once missing is-a relations are found, we need to repair the structure. This can be done by adding a set of is-a relations (called a *structural repair* in [10]) to the ontology such that when these are added, all missing is-a relations can be derived from the extended ontology. Clearly, the missing is-a relations themselves constitute a structural repair, but this is not always the most interesting solution for the domain expert. For instance, in a real case based on the Anatomy track from the OAEI 2008, we know that an is-a relation between *wrist joint* and *joint* is missing in MA and thus {*wrist joint* is-a *joint*} is a structural repair. However, knowing that there is an is-a relation between *wrist joint* and *limb joint*, a domain expert will most likely prefer to use the structural repair {*limb joint* is-a *joint*} instead. This is correct from a modeling perspective as well as more informative and would lead to the fact that the missing is-a relation between *wrist joint* and *joint* can be derived. In this particular case, using the second structural repair would actually also lead to the repairing of 6 other missing is-a relations in MA (e.g. between *elbow joint* and *joint*).

Our research needs to address the following questions.

(a) How can we generate structural repairs?
(b) How can we recognize structural repairs that are interesting for a domain expert?
(c) How can we recommend structural repairs?
(d) How can we execute repairs?

For the first question we need to find sets of is-a relations that would allow us to derive the missing is-a relations. In our approach we use a description logic knowledge base to check this. Further, not all structural repairs are interesting from a modeling point of view and therefore we need to define ways to recognize relevant structural repairs. We do this by defining preference relations between structural repairs and develop algorithms that take these preference relations into account.

A domain expert will repair the ontologies in a semi-automatic way and essentially chooses between the generated structural repairs. As there may be many of these we need to define ways to recommend structural repairs. We do this by involving other domain knowledge (e.g. WordNet or UMLS). Further, as there may be many missing is-a relations, it may not be practical to deal with all at the same time, and methods for deciding and executing repairs in an iterative way need to be developed.

**Integrated approach.** Up to now we have used ontologies and mappings as background knowledge for detecting missing is-a relations in another ontology. However, in the case

of networked ontologies we can actually detect and repair missing is-a relations for all ontologies at the same time. The repairing of missing is-a relations in one ontology may lead to the detection and repairing of missing is-a relations in other ontologies. Therefore, our definitions should be generalized to deal with this case and algorithms need to be extended or new algorithms developed.

**Evaluation.** Our approaches will be implemented in a system for detecting and repairing missing is-a relations. We will evaluate our approaches with existing networked ontologies (as we did in [10]). We will evaluate properties of the algorithms, quality of the generated and recommended repairs, as well as usability issues of the system.

## 4 Results

In our work until now we have studied the case where the ontologies are defined using named concepts and subsumption axioms. Most ontologies contain this case and many of the most well-known and used ontologies, e.g. in the life sciences, are covered by this setting. In [9] we discussed the use of a PRA in the setting of ontology alignment. One of the approaches included detecting missing is-a relations by using the structure of the ontologies and the PRA. Missing is-a relations were found by looking at pairs of equivalence mappings. If there is an is-a relation between the terms in the mappings belonging to one ontology, but there is no is-a relation between the corresponding terms in the other ontology, then we concluded that an is-a relation is missing in the second ontology. The detected missing is-a relations were then added to the ontologies. This is the simplest kind of structural repair.

In [10] we focused on the repairing of the missing structure. First, we defined the notion of structural repair. Then, we defined three preference relations that relate to heuristics that may be used by a domain expert when deciding which structural repair is more interesting. The first preference prefers structural repairs without redundant or non-contributing is-a relations. The second one prefers structural repairs with more informative ways of repairing. The third one prefers structural repairs which do not change existing is-a relations in the original ontology into equivalence relations. Further, we developed algorithms for generating structural repairs that take the preference relations into account. We also developed an algorithm for recommending structural repairs using other domain knowledge as well as algorithms for executing repairs. Based on these algorithms we implemented a prototype system RepOSE (*Rep*air of *O*ntological *S*tructure *E*nvironment), that allows a user to repair the structure of an ontology in a semi-automatic way. We evaluated our approach using MA and NCI-A.

## 5 Conclusions and Future Work

Motivated by the impact of defects in ontologies on their use as domain models and in semantically-enabled applications, we work on debugging the missing is-a structure in ontologies. We have already proposed approaches, developed algorithms and implemented a prototype that allow a domain expert to repair the is-a structure of ontologies in a semi-automatic way for the basic cases discussed in section 3.

In future work we will address the issues discussed in section 3. This will include dealing with ontologies represented in more expressive representation languages as well

as finding ways to optimize the generation of results. For instance, as the generation of structural repairs may take a lot of time, we may want to investigate ways to partition the set of missing is-a relations into parts that can be processed independently. Another interesting track is to investigate the integration of this work with ontology alignment.

## References


1. Preliminary results of the OAEI 2009 (2009), http://oaei.ontologymatching.org/2009/results
2. AMA. Adult mouse anatomical dictionary, http://www.informatics.jax.org/searches/AMA_form.shtml
3. Bada, M., Hunter, L.: Identification of OBO nonalignments and its implications for OBO enrichment. Bioinformatics, 1448–1455 (2008)
4. Cimiano, P., Staab, S.: Learning by googling. ACM SIGKDD Explorations Newsletter 6(2), 24–33 (2004)
5. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: 14th Int. Conference on Computational Linguistics, July 1992, pp. 539–545 (1992)
6. Ji, Q., Haase, P., Qi, G., Hitzler, P., Stadtmuller, S.: RaDON - repair and diagnosis in ontology networks. In: Demo at the 6th European Semantic Web Conference, pp. 863–867 (2009)
7. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Gray, B.: Repairing unsatisfiable concepts in OWL ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 170–184. Springer, Heidelberg (2006)
8. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics 3(4), 268–293 (2006)
9. Lambrix, P., Liu, Q.: Using partial reference alignments to align ontologies. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 188–202. Springer, Heidelberg (2009)
10. Lambrix, P., Liu, Q., Tan, H.: Repairing the missing is-a structure of ontologies. In: 4th Asian Semantic Web Conference, pp. 371–386 (2009)
11. Lambrix, P., Strömbäck, L., Tan, H.: Information integration in bioinformatics with ontologies and standards. In: Semantic Techniques for the Web: The REWERSE perspective, pp. 343–376 (2009)
12. Maedche, A., Pekar, V., Staab, S.: Ontology learning part one - on discovering taxonomic relations from the web. In: Web Intelligence. Springer, Heidelberg (2002)
13. Maedche, A., Staab, S.: Discovering conceptual relations from text. In: 14th European Conference on Artificial Intelligence, pp. 321–325 (2000)
14. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing ontology mappings. In: 20th National Conference on Artificial Intelligence, pp. 1408–1413 (2007)
15. NCI-A. National cancer institute - anatomy, http://www.cancer.gov/cancerinfo/terminologyresources/
16. 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)
17. van Hage, W.R., Katrenko, S., Schreiber, G.: A method to combine linguistic ontology-mapping techniques. In: 4th Int. Semantic Web Conference, pp. 732–744 (2005)
18. Völker, J., Hitzler, P., Cimiano, P.: Acquisition of OWL DL axioms from lexical resources. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 670–685. Springer, Heidelberg (2007)
19. Zavitsanos, E., Paliouras, G., Vouros, G.A., Petridis, S.: Discovering subsumption hierarchies of ontology concepts from text corpora. In: IEEE / WIC / ACM Int. Conference on Web Intelligence, pp. 402–408 (2007)

# Global Semantic Graph as an Alternative Information and Collaboration Infrastructure

Yan Shvartzshnaider

School of Electrical and Information Engineering,
The University of Sydney, Australia
yshv6985@uni.sydney.edu.au
www.ee.usyd.edu.au

**Abstract.** We propose the development of a Global Semantic Graph (GSG) as the foundation for future information and collaboration-centric applications and services. It would provide a single abstraction for storing, processing and communicating information based on globally interlinked semantic resources. The GSG adopts approaches and methods from the Semantic Web and thus facilitates a better information sharing abstraction.

## 1 Problem and Motivation

New and emerging Web technologies are gradually transforming the way information is communicated. In particular, popular social networks sites such as Facebook, MySpace and Twitter represent a real shift in the communication paradigm. The latest paradigm puts the focus on information itself and draws on collaboration methods between users, applications and services. The core building block for collaboration between applications is sharing/syncing of information or state, however, now it is not just about syncing within a specific application or context, but rather into a single global context. A prime example of such is the recently unveiled Google Wave [1], where users collaborate within a shared context of a single document (i.e., Wave). Despite their enhanced capabilities, however, Google Wave and other upcoming technologies are strictly limited to social interactions and operate in isolation from other existing Web applications.

Therefore, as the Internet's usage patterns gradually become more involved with multiple interactions between a variety of services and systems, there is an increasing demand for an architecture and infrastructure for interconnectivity and information exchange between applications on a global level. Although some applications may interoperate, for example, an email client might have an access to a contacts manager[2], the current Internet architecture does not provide the desired interoperability[3] as "there is no consistent approach for allowing interoperation and a system wide exchange of data between applications[2]." So, each application operates on a separate data structure and is unaware of related and relevant data in other applications[2].

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 483–487, 2010.

Motivated by these shortcomings of the Internet architecture and inspired by the new collaboration approaches, we propose a standardised architecture for information capture as well as communication between different information realms in the form of a global semantic graph (GSG). The GSG is a persistent, globally shared information space that is securely accessible and manipulated from/by any application and service.

In other words, the GSG forms a single global context within which the applications sync their respective data and state, so that it can be accessed and used for a variety of collaboration and interoperability purposes.

## 2 Approach and Methodology

In our approach, as depicted by Figure 1, we propose to have all future collaboration and communication processes carried out via the GSG. The GSG adopts approaches and methods from the Semantic Web that provides the "means to build the semantic bridges necessary for data exchange and application integration [2]". It is essentially, large-scale distributed tuple store to which applications and services simply "publish" their internal state changes via simple tuple insertion while a "subscription" is essentially a standing query that keeps the internal state synchronised with any new information provided by others.

We are currently interested in the following research questions:

1. How to insure good load balancing, scalability and information routing capabilities in a global storage system?
2. What is most efficient way to implement the pub/sub paradigm on top of a scalable semantic graph?
3. What is a proper security model that will work well with the GSG?

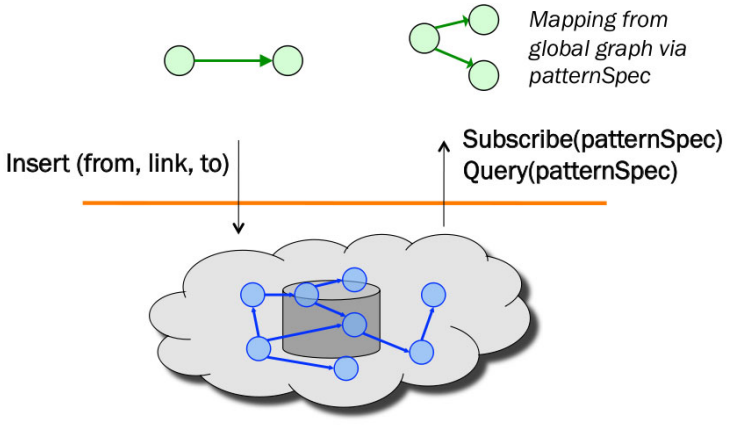



**Fig. 1.** The GSG combines Network, Database and Semantic Web in a single abstraction. Information is "filtered" by the subscription to a particular context pattern (e.g. John's emails) and then mapped on to a specific application (e.g. message reader).

To answers the above questions we are examining the work in the following research areas:

*Large tuple store.* We are considering the utilisation of a DHT-based *Structured P2P overlays* distributed storage system. However, while, on one hand it provides efficient data item discovery and uniform load distribution, in addition to self-organisation and node failure tolerance, on the other hand the DHT inherently does not support much needed wildcard and multi-attribute queries.

*Publish/Subscribe communication paradigm.* Our approach builds on evaluation of long-standing queries, also called continuous queries, which are issued once and remain in the system to return results every time the database is updated. We are looking at porting this approach to a large-scale distributed tuple store where an active standing query evaluation would act as a filter on the store's data and effectively provide a pub/sub service. In the case of the GSG, a standing query is essentially a set of tuple templates (or conditions) that return matching tuples. Therefore, an implementation of an efficient pattern matching algorithm (e.g. RETE [4]) with data-dependent, dynamic template re-writing on top of a distributed environment might serve as a possible solution.

*Reasoning.* The GSG architecture provides support to "anytime" distributed reasoning, similarly to the MARVIN [5] platform. Reasoning agents compute the deductive closure of the subscribed parts of the graph, and publish the results back into the GSG.

*Security.* We have some thoughts that involve using Named Graphs [6] for authentication and authorisation controls, however these initial ideas require further research and discussions with security experts.

To address above challenges we plan to follow the following research methodology: first, we will conduct an extensive literature review of current state-of-the-art technologies and methods in order to better understand them. Second, we plan to implement an efficient pattern matching algorithm and integrate it into a prototype system. Next, we will deploy the prototype system on top of a distributed environment through a testbeds framework (for LAN and WAN). Finally, we seek to evaluate the results and in order to optimise the system for better overall performance.

## 3 Results

This PhD research is a continuation of an Engineering Honours project thesis which looked into the GSG as an alternative to the Internet's information communication infrastructure. In particular, we were interested to see how the GSG abstraction impacts on the design and implementation of new services. That research project was awarded "The Best Research Project" prize in the Engineering Sydney Research Conversazione 2009[1], organised by The University

[1] http://www.eng.usyd.edu.au/engineeringsydney/conversazione.shtml

of Sydney. We are currently in the process of implementing and subsequently testing the GSG prototype. In particular, we have developed a basic prototype of a pub/sub system based on the RETE [4] algorithm. At the time of writing, the prototype was successfully deployed on top of the OMF (cOntrol and Management Framework)[2] testbed.

## 4 Related Work

The implementation of the GSG builds on many existing standards and systems. Due to space restrictions, we only discuss here related work highlights, in two key research areas: globally distributed content storage and the publish-subscribe communication paradigm.

**Distributed storage:** The increasing demand for online services has prompted a number of research efforts both in the academic community and the industry to develop a robust and scalable storage systems.

For example, Amazon's *Dynamo* [7] is a highly available key-value global storage system. Dynamo is guided by Amazon's e-commerce target requirements, which are mainly concerned with providing an "always-on" experience, even at a cost of temporarily sacrificing information consistency. Consequently, Dynamo completely lacks any security or data integrity mechanisms as it was designed to operate in a fully trusted domain. Nevertheless, there are a number of Dynamo's design aspects and assumptions that we see useful in relation to the GSG implementation. These include the use of a Distributed Hash Table (DHT) for data storage and routing, the "always-writeable" data store assumption (i.e. no updates, only writes), failures handling, versioning and replica synchronisation protocols.

In other related work, RDFPeers [8], one of the first distributed RDF stores was developed on top of *Multi-Attribute Addressable Network* (MAAN) [9]. RDFPeers supports atomic, disjunctive and conjunctive multi-predicate RDF queries, however it suffers from poor load balancing capabilities [10].

**Publish/Subscribe communication paradigm:** The implementation of a pub/sub schema on a global, Internet like scale is still an open research problem. Interesting work is being done by the EU funded PSIRP[3] project [11] that aims to develop an information-centric internetworking architecture based on the publish-subscribe paradigm[12]. The PSIRP project employs label switching *forwarding* to deliver information[3] and in this it differs to our approach described in Section 1.

## 5 Conclusion and Future Work

We have described our vision and motivation for a new Internet information-centric collaboration architecture in a form of a global semantic graph (GSG).

[2] http://omf.mytestbed.net/
[3] Publish-Subscribe Internet Routing Paradigm

GSG's design is motivated by the poor ability of Internet's architecture to provide the desired interoperability between applications [3]. The GSG is a persistent, distributed global graph that provides storage as well as publish/subscribe based communication. It differs from classical pub-sub systems in that publications to it "permanently" persist, and that the subscriptions are on the shared graph and not on the individual publications.

Our approach builds on many existing standards and systems that need to work together. Therefore, the implementation aspects include many challenges in a wide range of research areas. In particular, we wish to find a way to facilitate multi-attribute and range queries over distributed large-scale storage systems without compromising scalability and load balancing properties. Moreover, implementing a pub/sub communication paradigm on a global scale based on a persistent shared-state, together with the security aspects, is a clear must for a global system.

To seek answers to these and other related research questions, we plan to follow the methodology set out in Section 1, to develop a prototype implementation for the GSG.

## References


1. About Google Wave (January 2010), http://wave.google.com/help/wave/about.html
2. Groza, T., et al.: The Nepomuk Project - On the way to the Social Semantic Desktop. In: Proc. of I-Semantics, vol. 7, pp. 201–211 (2007)
3. Sarela, M., Rinta-aho, T., Tarkoma, S.: RTFM: Publish/subscribe Internetworking Architecture. In: ICT Mobile Summit, Stockholm (2008)
4. Forgy, C.: A Network Match Routine for Production Systems. Working Paper, Tech. Rep. (1974)
5. Anadiotis, G., et al.: MaRVIN: a distributed platform for massive RDF inference (2008), http://www.larkc.eu/marvin/btc2008.pdf
6. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs, Provenance and Trust. In: Proc. of the 14th Int. Conf. on the World Wide Web, p. 622 (2005)
7. DeCandia, G., et al.: Dynamo: Amazon's Highly Available Key-value Store. ACM SIGOPS Operating Systems Review 41(6), 220 (2007)
8. Cai, M., Frank, M.: RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network. In: Proc. of the 13th Int. Conf. on the World Wide Web, p. 657 (2004)
9. Cai, M., et al.: MAAN: A multi-attribute addressable network for grid information services. Journal of Grid Computing 2(1), 3–14 (2004)
10. Anadiotis, G., Kotoulas, S., Siebes, R.: An Architecture for Peer-to-Peer Reasoning. In: Proc. of 1st Workshop on New forms of reasoning for the Semantic Web, International Semantic Web Conference, ISWC 2007, Busan, Korea (2007)
11. Fotiou, N., Polyzos, G., Trossen, D.: Illustrating a Publish-Subscribe Internet Architecture. In: 2nd Euro-NF Workshop (2009)
12. PSIRP project homepage (December 2009), http://www.psirp.org/

# Scalable and Parallel Reasoning in the Semantic Web

Jacopo Urbani

Department of Computer Science, Vrije Universiteit Amsterdam
j.urbani@few.vu.nl

**Abstract.** The current state of the art regarding scalable reasoning consists of programs that run on a single machine. When the amount of data is too large, or the logic is too complex, the computational resources of a single machine are not enough. We propose a distributed approach that overcomes these limitations and we sketch a research methodology. A distributed approach is challenging because of the skew in data distribution and the difficulty in partitioning Semantic Web data. We present initial results which are promising and suggest that the approach may be successful.

## 1 Problem Statement

Most of the current reasoners are programs that are executed on a single machine. The scalability of these approaches is limited by the physical resources of the single machine. The size of the Semantic Web has grown to the point where this limitation notably affects the performance of the reasoners. Therefore, in order to realize the vision of Semantic Web, a scalable and efficient way to reason over an ever-growing amount of data is crucial.

The scalability is typically evaluated on two aspects: computational *complexity* (i.e. the ability to perform more complex tasks) and input *size* (i.e. the ability to process a larger input). It is essential that the reasoning process is *scalable* regarding both aspects. The research question we pose is:

> We aim to research algorithms which allow complex and scalable reasoning over a large amount of data (billions of statements).

The reasoning results should be accessed with interactive queries. In this context we identify two subtasks: reasoning and querying. Depending on the type of reasoning, these two tasks can be either independent from each others or strongly interlinked. When possible, we intend to put more emphasis on the task of reasoning than on the task of querying, but both tasks are important because if the results of the reasoning are unavailable to the end user then the entire process becomes meaningless.

## 2 State of the Art

The scientific community has already made notable efforts for reasoning in a large scale. An exhaustive list of all the relevant work goes beyond the scope of

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 488–492, 2010.

this paper. Here, we will only report some of the work that shows our intended goal.

Currently, several single-machine Semantic Web stores support reasoning and can load up to ten billion triples[1]. However, loading the data at this scale can take up to one week and it can only scale with more powerful hardware.

Hogan et al. [3] compute the closure of an RDF graph doing two passes over the data on a single machine. They have implemented only a fragment of the OWL Horst semantics, in order to prevent ontology hijacking.

Several distributed systems were proposed to calculate the closure and querying. Mika and Tummarello [4] use MapReduce to answer SPARQL queries over large RDF graphs, but details and results are not reported.

Soma and Prasanna [7] present a technique for parallel OWL inferencing through data partitioning. The experiments were conducted only on small datasets (1M triples) with a good speedup but the runtime is not reported.

Marvin [5] presents a technique which partitions the data in a peer-to-peer network but results with very large datasets have not been presented.

In Weaver and Hendler [12] incomplete RDFS reasoning is implemented on a cluster replicating the schema on all the nodes. This approach is embarrassingly parallel and it cannot be extended to more complicated logic like OWL.

Schlicht and Stuckenschmidt [6] present a promising technique to parallelize DL reasoning with a good speedup but the performance was evaluated on a small input.

Some of the approaches here presented have good scalability but on a weak logic ([12]), while others implement a complex logic like OWL but do not appear to scale to a very large size ([7],[3]). The approach we propose in the next section aims to research the best tradeoff between scalability and the complexity of richer logics so that we are able to perform complex but feasible reasoning over billions of statements (web-scale).

## 3 Proposed Approach

Our purpose is to reason over a very large amount of data which cannot be handled by a single machine. To overcome this problem, we propose a distributed approach where the reasoning task is executed simultaneously on several independent machines. We assume that we have complete control of the machines in which the data is processed.

We will consider only monotonic rule-based reasoning. The motivation behind this choice lies on several considerations:

- in the Web, the data is distributed and we cannot retract existing facts;
- there are already some standardized rule sets (RDFS, OWL Horst, OWL 2 RL) that are widely used and which allow us to compare our work with already existing one;
- currently there is no distributed rule-based reasoner which implements a complex logic on a very large scale.

[1] http://esw.w3.org/topic/LargeTripleStores

A distributed approach is potentially more scalable than a single machine approach because it can scale on two dimensions: the hardware and the number of the machines. However, it is more challenging to design because:

- In the Semantic Web, the data is not uniformly distributed, but instead there is an high data skew which generates load balancing problems;
- In rule-based reasoning the data must be joined together many times in order to find a possible derivation. Doing joins over distributed data is one of the most challenging tasks in a distributed environment.

Our approach aims to limit the exchange of the data (which is expensive) and try instead to move the computation (which is cheap) because rule based reasoning is mainly a data intensive task. There are several distributed programming models, like MapReduce [1] or Satin [11] which reduce the data communication and efficiently distribute the computation.

In some cases, there are additional technical problems introduced by a distributed approach. For example, the nodes must communicate to each other using an efficient protocol, otherwise the performance will suffer. For our work, we intend to use the Ibis framework [10] to ease the development of our approach. The Ibis framework offers many tools like IPL [2] or JavaGAT [9] which handles many technical aspects of the communication between the nodes and the heterogeneity of the system.

In our context, rule based reasoning can be applied either in a forward way or in a backward way. In forward reasoning the algorithm first materializes all the closure and then the data is queried in a database fashion. This approach is ideal when we have a dataset which does not change frequently and we need to query it extensively. It can be problematic if the closure is very large or if the data changes too frequently.

In backward reasoning the derivation is calculated on the fly when the data is queried. This approach is generally more complex and it makes the queries much slower, however it has the advantage that it works if the data changes often or if the complete closure is too large to materialize.

Our purpose is to apply the current programming models to the different types of reasoning (forward, backward, or a combination of the two) and to research what model is the most efficient and under which conditions. In case none of the existing programming model suits well for our purpose, our research will try to design a new programming model which overcomes the limitations of the existing ones.

## 4 Methodology

Our research will be carried out in several phases. In the first phase, we will study the existing literature and research some programming models which fit our need.

The second phase consists of implementing a reasoning algorithm using these specific programming models. We will first pick one programming model and we

will research an efficient algorithm which implements a weak logic (i.e. RDFS) using that model.

In the third phase we will evaluate the performance of the algorithm (the tests will be conducted mainly on the cluster DAS-4[2]). If the algorithm has good performance then we will increase the complexity of the logic, otherwise we will focus on the problems of the algorithm. If the problems are in the nature of the programming model, then we will try another model which could solve these problems (this means return to phase 1). If all programming models do not suit our problem, then the research will aim in finding a new programming model which fits our requirements and solve the problems that came up previously.

It is fundamental to clarify what we mean with "good" performance. Keeping in mind that our goal is to query the reasoning results in an interactive way, the evaluation will consider the total runtime necessary to reason and query the data. Another very important aspect in the evaluation is the scalability. The algorithm must be able to work with different input sizes and different number of nodes and it should implement a relatively complex logic (at least OWL-Horst or higher).

## 5 Results

Research was conducted to implement forward reasoning using the MapReduce programming model. At first the work focused on the RDFS semantics and several optimizations were introduced to speed up the computation. The performance showed linear scalability and an high throughput, but the work could not be extended to more complex logic because the optimizations exploited some specific characteristics of RDFS [8].

Next, the research aimed to see whether MapReduce could be used also to implement the more complex OWL reasoning. Other optimizations were introduced and the results of this work are currently under submission for the conference ESWC 2010.

The main limitation of the work done so far is that the reasoning cannot be applied only on a subset of the input. In a realistic scenario where the dataset is gradually extended with new data, computing the closure every time over all the input is not efficient.

The next step in our research consists of finding a way to query the derivation obtained by the reasoner. In order to do so, the results can be loaded on a database or an RDF store and queried with the standard techniques.

## 6 Conclusions and Future Work

In this paper, we have described the problem of scalable reasoning and proposed a distributed approach to solve it. The results we have obtained with forward reasoning and MapReduce showed that indeed this approach may be successful but further research is necessary.

[2] http://www.cs.vu.nl/das/

In our approach, we made some assumptions to narrow down the scope of the research to distribution and performance. We do not consider other important aspects of reasoning on a large scale like, for example, the quality of the data. Some work about it is presented in [3] and it could be integrated in our reasoning algorithms to make them more robust. Another assumption we make is that we have control of the machines in which the data is processed. If the data comes from the Web it must be first fetched and copied locally. Some extra work could extend the reasoning process to work directly on the distributed data.

## References

[1] Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of the USENIX Symposium on Operating Systems Design & Implementation (OSDI), pp. 137–147 (2004)
[2] Drost, N., et al.: Resource tracking in parallel and distributed applications. In: Proc. of 17th IEEE International Symposium on High-Performance Distributed Computing, HPDC (2008)
[3] Hogan, A., Harth, A., Polleres, A.: Scalable authoritative OWL reasoning for the web. International Journal on Semantic Web and Information Systems 5(2) (2009)
[4] Mika, P., Tummarello, G.: Web semantics in the clouds. IEEE Intelligent Systems 23(5), 82–87 (2008)
[5] Oren, E., et al.: Marvin: A platform for large-scale analysis of Semantic Web data. In: Proceedings of the International Web Science Conference (2009)
[6] Schlicht, A., Stuckenschmidt, H.: Distributed Resolution for Expressive Ontology Networks. In: Proceedings of the Web Reasoning and Rule Systems 2009, p. 87. Springer, Heidelberg (2009)
[7] Soma, R., Prasanna, V.: Parallel inferencing for OWL knowledge bases. In: International Conference on Parallel Processing, pp. 75–82 (2008)
[8] 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)
[9] van Nieuwpoort, R.V., Kielmann, T., Bal, H.E.: User-friendly and reliable grid computing based on imperfect middleware. In: Proceedings of the ACM/IEEE Conference on Supercomputing (2007)
[10] van Nieuwpoort, R.V., et al.: Ibis: a flexible and efficient Java based grid programming environment. Concurrency and Computation: Practice and Experience 17(7-8), 1079–1107 (2005)
[11] van Nieuwpoort, R.V., et al.: Satin: Simple and efficient Java-based grid programming. Scalable Computing: Practice and Experience 6(3) (2005)
[12] Weaver, J., Hendler, J.: 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)

# Exploring the Wisdom of the Tweets: Towards Knowledge Acquisition from Social Awareness Streams

Claudia Wagner

JOANNEUM RESEARCH,
Steyrergasse 17, 8010, Graz, Austria
claudia.wagner@joanneum.at

**Abstract.** Although one might argue that little wisdom can be conveyed in messages of 140 characters or less, this PhD research sets out to explore if and what kind of knowledge can be acquired from different *aggregations of social awareness streams*. The expected contribution of this research is a network-theoretic model for defining, comparing and analyzing different kinds of social awareness streams and an experimental prototype to extract semantic models from them.

**Keywords:** Knowledge Acquisition, Social Web, Semantic Analysis.

## 1 Introduction

In the last decade, the emergence of social media applications such as Wikipedia, Del.icio.us and Flickr has inspired a community of researchers to tap into user-generated data as an interesting alternative to knowledge acquisition. Instead of formally specifying meaning *ex-ante* through for example agreed-upon ontologies, the idea was to capture meaning from user-generated data *ex-post*.

With the emergence of social awareness streams, popularized by applications such as Twitter or Facebook and formats such as activitystrea.ms, a new form of communication and knowledge sharing has enriched the social media landscape. Personal awareness streams usually allow users to post short, natural-language messages as a personal stream of data that is being made available to other users. We refer to the aggregation of such personal awareness streams as *social awareness streams* (short streams), which contain a set of short messages from different users usually displayed in reverse chronological-order. Although one could argue that little wisdom can be conveyed in messages of 140 characters or less, this PhD research aims to explore (1) if and what kind of knowledge can be acquired from different *aggregations of social awareness streams* and (2) to what extent the semantics of social awareness streams are influenced by stream characteristics and vice versa.

## 2 Related Work

Semantic analysis of social media applications is an active research area, in part because on the one hand social media provides access to the "collective

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 493–497, 2010.

intelligence" of millions of users while on the other hand it lacks explicit semantics. Exploiting the "collective intelligence" of social media applications is therefore a promising and challenging aim of current research efforts.

The work of Mika [1] and Heymann et al [2] illustrate how graph-based measures, such as centrality and clustering coefficient, can be used to extract broader and narrower terms from tag spaces. Schmitz et al. [3] describe how a statistical subsumption model can be applied to induce hierarchical relations of tags. Clustering approaches (e.g., [4]) identify groups of highly related tags and establish hierarchical relationship between these clusters.

Although social awareness streams and tagging systems have common characteristics (e.g., in both systems users relate resources with free-form tags), they are used for different purpose and reveal significant, structural differences (e.g., in social awareness streams users may relate resources with other resources or other users). Due to its novelty, little research on social awareness streams exists to date. Some recent research (e.g., [5]) investigates user activities on Twitter. Another line of research (e.g., [6]) focuses on analyzing and characterizing content of social awareness stream messages.

## 3 Proposed Approach and Methodology

To characterize and compare different aggregations of streams we will develop a network-theoretic model and measures for social awareness streams. To explore if and what kind of knowledge can be acquired from streams, a system (KASAS) for characterizing and comparing stream aggregations and extracting emerging semantic models from them will be developed.

### 3.1 The KASAS System

Figure 1 shows the basic steps of the KASAS system which takes one or several keywords as input and returns as output a model of concepts and relations between them. The resulting model could for example be used to enrich existing ontologies such as DBpedia with semantic models (containing e.g. recent information about events or users relevant for a specific DBpedia concept). The **Stream Aggregation and Characterization** component creates for a given keyword one or several aggregations of streams and characterizes them via various stream measures. These measures can e.g. help to identify the most promising streams in terms of semantic analysis. The streams are then preprocessed by the **Lexical Normalization** component. Concepts (denoted by one or several labels) and their relations are extracted by the **Concept and Association Mining** component. This component will use simple network transformations, latent and explicit concept models to extract concepts from stream aggregations. To mine associations between concepts we will use various information-theoretic, statistical, semantic similarity measures. Finally, the **Entity Discovery** component will discover types of concepts by harnessing the Web of Data as background knowledge and by analyzing stream characteristics. Stream characteristics can help to infer the basic type of a stream's topic, which is e.g. `event` in case of the `#eswc2010` stream.

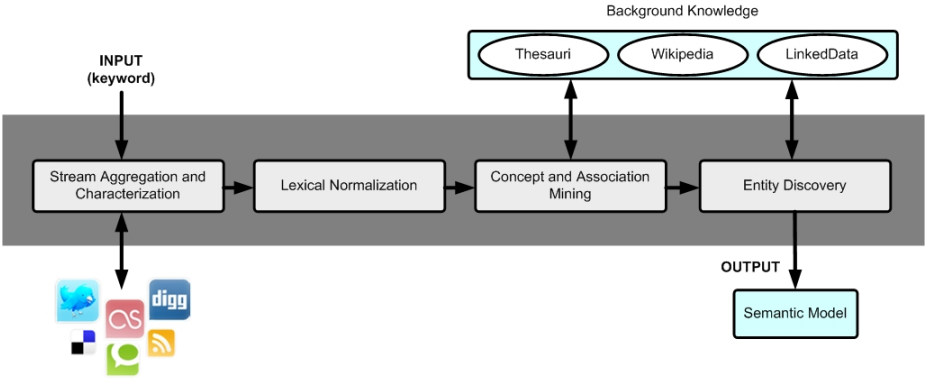



**Fig. 1.** The KASAS (**K**nowledge **A**cquisition from **S**ocial **A**wareness **S**treams) system

### 3.2 Evaluation

When it comes to the semantic analysis of social awareness streams, the extent to which different streams approximate the semantic understanding of users participating in these streams is interesting to investigate. We will conduct evaluations that include user assessments of concepts, relations and their ranking. In addition, to evaluate the quality of extracted concepts and their relations, we may use external, hand-crafted taxonomies, such WordNet, the Open Directory Project or DBpedia as semantic grounding, and internal, semantic models emerging through user's usage of special syntax, such as microsyntax[1] or HyperTwitter[2].

To evaluate concept-user relations we can select a defined set of user accounts belonging to researchers and compare their list of ranked concepts with a list of keywords of their papers. In addition, we may ask themselves (self-assessment) and other researchers (peer-assessment) to asses the extracted and ranked list of concepts related with them.

## 4 Results

Based on the existing tripartite structure of *folksonomies*, we introduce a network-theoretic model of social awareness streams consisting of messages, users and content of messages. As an adaption of the folksonomy data structure, the model of social awareness streams introduces qualifiers on the tripartite structure that allow to accommodate user generated syntax. We formally define the model as follows:

**Definition 1** *A social awareness stream $S$ is a tupel $S = (U_{q1}, M_{q2}, R_{q3}, Y, ft)$, where*

[1] http://microsyntax.pbworks.com/
[2] http://semantictwitter.appspot.com/

- *U, M and R are finite sets whose elements are called users, messages and resources.*
- *Qualifier q1 represents different ways users are involved in a stream (e.g. users can be author or target of message), q2 represents the different types of messages M (e.g. public broadcast messages or private direct messages), and q3 represents the different types of resources that can be included in messages of streams (e.g. hashtags, links or keywords)*
- *Y is a ternary relation* $Y \subseteq U \times M \times R$ *between U, M, and R.*
- *ft is a function which assigns to each Y a temporal marker.*

In addition, we created various stream measures (such as the social, conversational, temporal, informational and topical diversity measure) to characterize and compare different stream aggregations. Based on the formal model of social awareness streams and the predefined measures, we analyzed and compared different aggregations of Twitter streams (user list, user directory, hashtag and keyword streams), which were all related to the concept `semantic web` and were all recorded within the same time interval (8 days).

Since measures for similarity and relatedness are not well developed for three-mode networks we considered various ways to obtain qualified two-mode networks (resource-author, resource-message, resource-hashtag and resource-link networks) from these stream aggregations. To surface semantic relations between resources, we produced one-mode networks (see e.g. Figure 2) by multiplying the corresponding two-mode network matrices with their transpose $M * M^T$. Different semantic relations are created because of the different ways associations are established between resources. We compared the resulting networks by assessing their most important concepts and relations. Our empirical results indicate that hashtag streams are in general rather robust against external events (such as New Years Eve), while user list stream aggregations are more perceptible to such "disturbances". Networks generated via resource-hashtag transformations

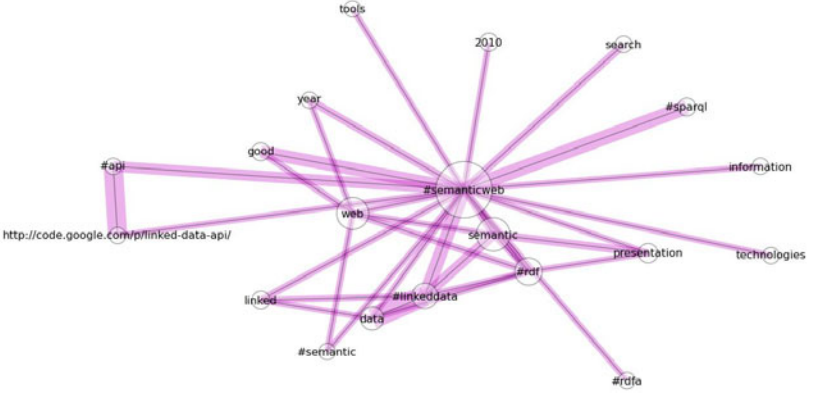



**Fig. 2.** Resource network computed from the resource-hashtag network of the *#semanticweb* hashtag stream

seem to have the power to reduce the non-informational noise in streams and reveal meaningful semantic models.

## 5 Conclusions and Future Work

While the developed network-theoretic model of social awareness streams is general, the first empirical results of this PhD research are limited to a single concept (`semantic web`). In future, we will expand our analysis to a broader variety of social awareness streams and conduct experiments over greater periods of time. Since the semantic analysis conduced in our first experiments is based on simple network transformations, we will study whether more sophisticated knowledge acquisition methods produce different results. Finally, we will evaluate the semantic models produced by different stream aggregations and explore to what extend they approximate the semantic understanding of users that are involved in these streams.

**Acknowledgments.** I would like to thank my supervisor Markus Strohmaier for his guidance, support and fruitful discussions and JOANNEUM RESEARCH for funding this research.

## References

1. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semant. 5(1), 5–15 (2007)
2. Heymann, P., Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Computer Science Department (April 2006)
3. Schmitz, P.: Inducing ontology from flickr tags. In: Proceedings of the Workshop on Collaborative Tagging at WWW 2006, Edinburgh, Scotland (May 2006)
4. Begelman, G., Keller, P., Smadja, F.: Automated tag clustering: Improving search and exploration in the tag space. In: Collaborative Web Tagging Workshop at WWW 2006, Edinburgh, Scotland (2006)
5. Huberman, B.A., Romero, D.M., Wu, F.: Social networks that matter: Twitter under the microscope. ArXiv e-prints (December 2008)
6. Naaman, M., Boase, J., Lai, C.H.: Is it all about me? user content in social awareness streams. In: Proceedings of the ACM 2010 Conference on Computer Supported Cooperative Work (2010)

# Two Phase Description Logic Reasoning for Efficient Information Retrieval

Zsolt Zombori

Budapest University of Technology and Economics, Hungary
zombori@cs.bme.hu

## 1 Overview

Description Logics (DLs) [1] is family of logic languages designed to be a convenient means of knowledge representation. They can be embedded into FOL, but – contrary to the latter – they are decidable which gives them a great practical applicability. A DL knowledge base consists of two parts: the TBox (terminology box) and the ABox (assertion box). The TBox contains general background knowledge in the form of rules that hold in a specific domain. The ABox stores knowledge about individuals. For example, let us imagine an ontology about the structure of a university. The TBox might contain statements like "Every department has exactly one chair", "Departments are responsible for at least 4 courses and for each course there is a department responsible for it". In contrast, the ABox might state that "The Department of Computer Science is responsible for the course Information Theory" or that "Andrew is the chair of the the Department of Music".

As DL languages are being used more and more frequently, there is an increasing demand for efficent automated reasoning services. Some reasoning tasks involve the TBox only. This is the case, for example, when we want to know what rules follow from the ones that we already know, or we want to verify that the model of a certain domain does not contain obvious mistakes in the form of contradictions and unsatisfiable concepts. We might want to make sure that there are not so many restrictions on the chair that it is impossible to be one (which is the case if he has to spend 70 percent of his time on research an another 70 percent on teaching). Other reasoning problems use both the ABox and the TBox: in such cases we might ask if a certain property holds for a certain individual (*instance check* – Is Andrew a chair?) or we might want to collect all individuals satisfying a given property (*instance retrieval* – What are the courses taught by the Department of Music?).

The Tableau Method [1] has long provided the theoretical background for DL reasoning and most existing DL reasoners implement some of its variants. Typical DL reasoning tasks can be reduced to concept consistency checking and this is exactly what the Tableau Method provides. While the Tableau itself has proven to be very efficient, the reduction to consistency check is rather costly for some ABox reasoning tasks. In particular, instance retrieval (i.e., to enumerate those individuals that belong to a given concept) requires running the

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 498–502, 2010.

Tableau Method for every single individual that appears in the knowledge base. Several techniques have been developed to make tableau-based reasoning more efficient on large data sets, (see e.g. [2]), that are used by the state-of-the-art DL reasoners, such as RacerPro [3] or Pellet [4].

Other approaches use first-order resolution for reasoning. A resolution-based inference algorithm is described in [5] which is not as sensitive to the increase of the ABox size as the tableau-based methods. The system KAON2 [6] is an implementation of this approach, providing reasoning services over the description logic language $\mathcal{SHIQ}$. The algorithm used in KAON2 in itself is not any more efficient for instance retrieval than the Tableau, but several steps that involve only the TBox can be performed before accessing the ABox, after which some axioms can be eliminated because they play no further role in the reasoning. This yields a qualitatively simpler set of axioms which then can be used for an efficient, query driven data reasoning. For the second phase of reasoning KAON2 uses a disjunctive datalog engine and not the original calculus. Thanks to the preprocessing, query answering is very focused, i.e., it accesses as little part of the ABox as possible. However, in order for this to work, KAON2 still needs to go through the whole ABox once at the end of the first phase.

## 2 Research Direction

In my PhD work I try to develop algorithms that can be used for reasoning over large ABoxes while the TBox is relatively small. These assumptions do not hold for all ontologies, but there are some very important examples when this is the case: one can, for instance, think of searching the WEB in the context of a specific, well characterized domain.

It seems that the complexity comes from two sources: on one hand the TBox contains complex background knowledge that requires sophisticated reasoning, and on the other the size of the ABox makes the sophisticated algorithm too slow in practice. An important lesson to be learned from KAON2 is that we might be able to cope with these two sources separately: let us perform the complex reasoning on the TBox – which we assume to be small – and turn it into a syntactically simpler set of rules before accessing the ABox. Afterwards, the simpler rules can be used for a focused, query driven ABox reasoning.

It is not clear how to separate the reasoning for the Tableau. This algorithm tries to build a model of the knowledge base, but a model of a small part of the knowledge base is not necessarily useful for constructing a model of the whole. Resolution approaches are more suitable: we can deduce implicit consequences of the axioms in one way at the beginning and then deduce further consequences in another way. In particular, we will be interested in solutions where we start with a bottom-up strategy and finish with a focused top-down strategy.

To perform first-order resolution, we need to transform the initial axioms to first-order clauses. While the initial knowledge base does not contain function symbols (there are no functions in DLs), existential restrictions and minimum restrictions in the TBox translate to existential quanifiers, which are eliminated

by introducing new function symbols, called *skolem functions*. This is problematic, because termination is hard to guarantee if we can obtain terms of ever increasing depth. Furthermore, some top-down reasoning algorithms (top-down reasoning is a must if the ABox is really large), such as datalog only work if there are no function symbols. For this reason, it is very important to find some way to eliminate function symbols before performing the data reasoning. Note that this is intuitively very possible: the ABox does not contain any knowledge about functions since they were introduced by us during clausifying the axioms from the TBox. Hence, everything that is to know about function symbols is in the clauses derived from the TBox and whatever role they play, they should be able to play it at the beginning of the reasoning.

## 3 Two Phase Reasoning

The above considerations motivate a two phase reasoning algorithm. In the first phase we only work with the clauses derived from the TBox. We use a bottom-up algorithm, deduce lots of consequences of the TBox, in particular all the important consequences of the clauses containing function symbols. By the end of the first phase, function symbols can play no further role and hence the clauses containing them can be eliminated. The second phase now begins and the reduced clause set can be used for a focused, top-down reasoning on the ABox.

This separation of TBox and ABox reasoning is only partially achieved in [6]. By the end of the first phase, we can only eliminate clauses with term depth greater than one. So, while function symbols persist, there is no more nesting of functions into each other. In order for the second phase to work, all function symbols are eliminated using a *syntactic* transformation: for every function symbol and every constant in the ABox a new constant is introduced. Note that this step involves scanning through the ABox and results in adding new constants whose number is linear in the size of the ABox.

Reading the whole ABox even once is not a feasible option in case the ABox contains billions of assertions or the content of the ABox changes so frequently that on-the-fly ABox access is an utmost necessity. Such scenarios include reasoning on web-scale or using description logic ontologies directly on top of existing information sources, such as in a DL based information integration system.

## 4 Results

I started my PhD at Budapest University of Technology in September 2009. I work as member of a team developing the DLog DL data reasoner [7], available to download at `http://www.dlog-reasoner.org`. This is a resolution based reasoner, built on principles similar to KAON2. One difference is that instead of a datalog engine, we use the reasoning mechanism of the Prolog language [8] to perform the second phase [9]. Reasoning with function symbols using Prolog is possible, unlike the datalog engine, but for considerations about termination it is equally important to eliminate function symbols during the first phase.

I work to provide DLog with a purely two phase reasoning algorithm. In [10] I presented a modified resolution calculus for the $\mathcal{SHIQ}$ language that allows us to perform more inferences in the first phase (compared with KAON2), yielding a simpler TBox to work with in the second phase. Namely, the new calculus ensures that no function symbols remain at all, without the need to go through the ABox. The modification makes the first phase somewhat slower, however, the speed of the second phase becomes independent of the amount of data that is irrelevant to the query. The greater the ABox the better DLog performs compared to its peers. Another great advantage of DLog is that its architecture allows for storing the ABox in an external database that is assessed through direct database queries.

Afterwards, I worked on a new DL calculus ([11] and [12]) where we move resolution from first-order clauses to DL axioms, saving many intermediary transformation steps. Even if the speed of the first phase is not as critical as that of the second, this optimisation is important. With the increase of the TBox the first phase can become hopelessly slow, such that DLog is impossible to use. Making the first phase faster slightly increases the critical TBox size within which it is still worth reasoning with DLog. On the other hand, the DL calculus is a complete algorithm for TBox reasoning. It is novel in that the metodology is still resolution, but the inference rules are given directly for DL expressions. It is not as fast for TBox reasoning as the Tableau, but it provides an alternative and I hope that it will motivate research in the area. I tried to extend the DL calculus to ABox reasoning, but I have not yet been successful in doing that.

## 5 Current Work

I am currently working to extend the DLog for more expressive ontologies. In particular, I try to extend [10] from $\mathcal{SHIQ}$ to $\mathcal{SHOIQ}$, i.e., provide reasoning support for nominals. [13] presents a resolution based algorithm for reasoning over $\mathcal{SHOIQ}$, but it is not clear if the desired separation of TBox and Abox reasoning can be achieved using their algorithm. In longer terms, I hope to develop purely two phase reasoning for even more expressive languages such as $\mathcal{RIQ}$ and $\mathcal{SROIQ}$, the logic of the new web ontology language OWL 2.

I am working to better explore the complexity of our algorithms. Bottom up reasoning in the first phase is very costly: it is at most triply exponential in the size of the TBox, although our experiments indicate that there could be a better upper bound. We also need to better explore the clauses that are deduced from the TBox. While our main interest is to eliminate function symbols, we deduce other consequences as well. Some of them make the data reasoning faster, some of them do not, and we cannot yet well characterize them.

## 6 Concluding Remarks

With the proliferation of knowledge intensive applications, there is a vivid research in the domain of knowledge representation. Description Logics are designed to be a convenient means for such representation task. One of the main

advantages over other formalisms is a clearly defined semantics. This opens the possibility to provide reasoning services with mathematical rigorousness.

My PhD work is concerned with Description Logic reasoning. I am particularly interested in ABox reasoning when the available data is really large. This domain is much less explored than TBox reasoning. Nevertheless, reasoning over large ABoxes is useful for problems like web-based reasoning.

I am one of the developers of the DLog data reasoner which implements a two phase reasoning: the first phase uses complex reasoning to turn the TBox into simple rules, while the second phase is geared towards fast query answering over large ABoxes. DLog currently supports the $\mathcal{SHIQ}$ DL language, but we plan to extend it as far as $\mathcal{SROIQ}$, the logic behind OWL 2.

## References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2004)
2. Haarslev, V., Möller, R.: Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In: Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004, Whistler, BC, Canada, June 2-5, pp. 163–173 (2004)
3. Haarslev, V., Möller, R., van der Straeten, R., Wessel, M.: Extended Query Facilities for Racer and an Application to Software-Engineering Problems. In: Proceedings of the 2004 International Workshop on Description Logics (DL 2004), Whistler, BC, Canada, June 6-8, pp. 148–157 (2004)
4. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semant. 5(2), 51–53 (2007)
5. Hustadt, U., Motik, B., Sattler, U.: Reasoning for Description Logics around SHIQ in a resolution framework. Technical report, FZI, Karlsruhe (2004)
6. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Universität Karlsruhe (TH), Karlsruhe, Germany (January 2006)
7. Lukácsy, G., Szeredi, P., Kádár, B.: Prolog based description logic reasoning. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 455–469. Springer, Heidelberg (2008)
8. Colmerauer, A., Roussel, P.: The birth of Prolog. ACM, New York (1996)
9. Lukácsy, G., Szeredi, P.: Efficient description logic reasoning in Prolog: the DLog system. Theory and Practice of Logic Programming 9(3), 343–414 (2009)
10. Zombori, Z.: Efficient two-phase data reasoning for description logics. In: Bramer, M. (ed.) IFIP AI. IFIP, vol. 276, pp. 393–402. Springer, Heidelberg (2008)
11. Zombori, Z., Lukácsy, G.: A resolution based description logic calculus. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009), CEUR Workshop Proceedings, Oxford, UK, July 27-30, vol. 477 (2009)
12. Zombori, Z.: A resolution based description logic calculus. Submitted to Acta Cybernetica (2009)
13. Kazakov, Y., Motik, B.: A resolution-based decision procedure for SHOIQ. Journal of Automated Reasoning 40(2-3), 89–116 (2008)

# Author Index