

Jeff Z. Pan Huajun Chen Hong-Gee Kim
Juanzi Li Zhe Wu Ian Horrocks
Riichiro Mizoguchi Zhaohui Wu (Eds.)

LNCS 7185

The Semantic Web

Joint International Semantic Technology Conference, JIST 2011
Hangzhou, China, December 2011
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Jeff Z. Pan Huajun Chen Hong-Gee Kim
Juanzi Li Zhe Wu Ian Horrocks
Riichiro Mizoguchi Zhaohui Wu (Eds.)

The Semantic Web

Joint International Semantic Technology Conference
JIST 2011
Hangzhou, China, December 4-7, 2011
Proceedings

Volume Editors

Jeff Z. Pan
University of Aberdeen, UK; jeff.z.pan@abdn.ac.uk

Huajun Chen
Zhaohui Wu
Zhejiang University, Hangzhou, China; huajunsir@msn.com, wzh@zju.edu.cn

Hong-Gee Kim
Seoul National University, Korea; hgkim@snu.ac.kr

Juanzi Li
Tsinghua University, Beijing, China; lijuanzi2008@gmail.com

Zhe Wu
Oracle Corporation, Redwood Shores, CA, USA; alan.wu@oracle.com

Ian Horrocks
Oxford University, UK; ian.horrocks@comlab.ox.ac.uk

Riichiro Mizoguchi
Osaka University, Japan; miz@ei.sanken.osaka-u.ac.jp

ISSN 0302-9743
ISBN 978-3-642-29922-3
DOI 10.1007/978-3-642-29923-0
Springer Heidelberg Dordrecht London New York

e-ISSN 1611-3349
e-ISBN 978-3-642-29923-0

Library of Congress Control Number: 2012936366

CR Subject Classification (1998): I.2.4, H.3.5, I.2.6, C.2, H.3.3, H.2

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

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

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

Printed on acid-free paper

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

Preface

The Joint International Semantic Technology Conference (JIST) is a joint event for regional Semantic Web-related conferences. This year's JIST brought together two regional conferences: ASWC 2011 (The Asian Semantic Web Conference 2011) and CSWC 2011 (The 5th Chinese Semantic Web Conference). This year's conference was held in Hangzhou, China. This volume contains the main proceedings of JIST 2011, including both full papers and short papers.

JIST 2011 received a total of 82 submissions to the research paper track. A rigorous review process was conducted by the Program Committee of recognised experts in the field of semantic technology from around the world. Each submission received at least three reviews. Final decisions were made during a meeting of the Program Committee Chairs, with 21 full papers eventually being accepted (a 25% acceptance rate). In addition, 12 short papers were accepted for the poster/demonstration program.

This year's technical program also included three invited talks given by leading figures from both the academic and business world. This year's talks were given by Ian Horrocks of Oxford University; Abraham Bernstein of the University of Zurich; and Mark Greaves of Vulcan, Inc. Furthermore, JIST 2011 included a full-day of tutorials prior to the technical program. Tutorial presenters included Jesse Wang, Ning Hu and Mark Greaves from Vulcan Inc; Jeff Z. Pan from the University of Aberdeen; and Shonali Krishnaswamy and Yuan-Fang Li from Monash University.

The hard work and close cooperation of a number of people have contributed to the success of this conference. We would like to thank the members of the Organizing Committee and Program Committee for their support; the supporting staff and students from Zhejiang University for their assistance with the management of the conference; and the authors and participants who are the primary reason for the success of this conference.

December 2011

Jeff Z. Pan
Huajun Chen
Hong-Gee Kim
Juanzi Li
Zhe Wu
Ian Horrocks
Riichiro Mizoguchi
Zhaohui Wu

Organization

Program Committee

Witold Abramowicz	The Poznan University of Economics, Poland
Franz Baader	TU Dresden, Germany
Enhong Chen	University of Science & Technology of China, China
Liming Chen	University of Ulster, UK
Gong Cheng	Nanjing University, China
Philippe Cudre-Mauroux	University of Fribourg, Switzerland
Bernardo Cuenca Grau	University of Oxford, UK
Claudia D'Amato	University of Bari, Italy
Mathieu D'Aquin	Knowledge Media Institute, The Open University, UK
Aba-Sah Dadzie	The University of Sheffield, UK
Mike Dean	Raytheon BBN Technologies, USA
Paola Di Maio	University of Strathclyde, UK
Zhongli Ding	Google, USA
Dejing Dou	University of Oregon, USA
Jianfeng Du	Guangdong University of Foreign Studies, China
Achille Fokoue	IBM Research
Zhiqiang Gao	Southeast University, China
Alexander Garcia	University of Bremen, Germany
Birte Glimm	The University of Oxford, UK
Peiqin Gu	Zhejiang University, China
Sung-Kook Han	Won Kwang University, South Korea
Siegfried Handschuh	Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, Ireland
Tom Heath	Talis Systems Ltd.
Kaoru Hiramatsu	NTT
Pascal Hitzler	Kno.e.sis Center, Wright State University, USA
Aidan Hogan	Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, Ireland
Laura Hollink	Delft University of Technology, The Netherlands
Masahiro Hori	Kansai University, Japan
Wei Hu	Nanjing University, China

VIII Organization

Zhisheng Huang	Vrije University Amsterdam, The Netherlands
Rajaraman Kanagasabai	Institute for Infocomm Research, Singapore
Takahiro Kawamura	Toshiba Corp., Japan
Eunghee Kim	Seoul National University, Korea
Yoshinobu Kitamura	I.S.I.R., Osaka University, Japan
Yuan Fang Li	National University of Singapore, Singapore
Dan Liu	Google, USA
Diana Maynard	University of Sheffield, UK
Jing Mei	IBM China Research Lab, China
Ralf Möller	TUHH, Germany
Hyun Namgoong	Electronics and Telecommunications Research Institute, South Korea
Ekawit Nantajeewarawat	Sirindhorn Intl. Inst. of Tech., Thammasat University, Thailand
Peter Patel-Schneider	Bell Labs Research
Dimitris Plexousakis	Institute of Computer Science, FORTH, Greece
Guilin Qi	Southeast University, China
Yuzhong Qu	Nanjing University, China
Jinghai Rao	Nokia
Marco Ronchetti	University of Trento, Italy
Matthew Rowe	Knowledge Media Institute, UK
Manuel Salvadores	University of Southampton, UK
Stefan Schlobach	Vrije Universiteit Amsterdam, The Netherlands
Murat Sensoy	University of Aberdeen, UK
Yi-Dong Shen	Institute of Software, The Chinese Academy of Sciences, China
Michael Sintek	DFKI GmbH, Germany
Kavitha Srinivas	IBM Research
Giorgos Stoilos	Oxford University Computing Laboratory, UK
Umberto Straccia	ISTI-CNR, Italy
Hideaki Takeda	National Institute of Informatics and The University of Tokyo, Japan
Kerry Taylor	CSIRO ICT Centre
Jilei Tian	Nokia Research Center
Vassilis Tzouvaras	National Technical University of Athens, Greece
Denny Vrandečić	KIT, Germany
Haofen Wang	Shanghai Jiao Tong University, China
Jesse Jiaxin Wang	Vulcan Inc.
Kewen Wang	Griffith University, Australia
Shenghui Wang	Wageningen UR, The Netherlands
Yimin Wang	Lilly Singapore Centre for Drug Discovery, Singapore
Zhe Wu	Oracle
Guotong Xie	IBM China

Bin Xu	DCST, Tsinghua University, China
Takahira Yamaguchi	Keio University, Japan
Yong Yu	Shanghai Jiao Tong University, China
Suckchan Yun	Daum Corporation
Guo-qiang Zhang	Case Western Reserve University, USA
Jun Zhao	University of Oxford, UK
Yuting Zhao	University of Aberdeen, UK
Hai-Tao Zheng	Tsinghua University, China
Yi Zhou	University of Western Sydney, Australia

Additional Reviewers

Abela, Charlie	Käfer, Tobias
Carral Martínez, David	Li, Xuan
Daskalaki, Evangelia	Ma, Yuanchao
Davis, Brian	Ma, Yue
Deng, Jun	Martiny, Karsten
Dragan, Laura	Morita, Takeshi
Du, Liang	Mutharaju, Raghava
Ell, Basil	Novacek, Vit
Feng, Yuzhang	Oezcep, Oezguer Luetfue
Gong, Saisai	Peñaloza, Rafael
Han, Sung-Kook	Ren, Yuan
Hong, Liang	Sen, Luo
Ichise, Ryutaro	Tudorache, Tania
Ji, Qiu	Wang, Jian
Joshi, Amit Krishna	Wieloch, Karol
Kharlamov, Evgeny	Wu, Huiyao
Kondylakis, Haridimos	Wu, Yuchen
Kritikos, Kyriakos	Zeng, Cheng

Table of Contents

A Method of Contrastive Reasoning with Inconsistent Ontologies	1
<i>Jun Fang, Zhisheng Huang, and Frank van Harmelen</i>	
Parallel ABox Reasoning of \mathcal{EL} Ontologies	17
<i>Yuan Ren, Jeff Z. Pan, and Kevin Lee</i>	
RP-Filter: A Path-Based Triple Filtering Method for Efficient SPARQL Query Processing	33
<i>Kisung Kim, Bongki Moon, and Hyoung-Joo Kim</i>	
Constructing Virtual Documents for Ontology Matching Using MapReduce	48
<i>Hang Zhang, Wei Hu, and Yuzhong Qu</i>	
Semantic Flow Networks: Semantic Interoperability in Networks of Ontologies	64
<i>Valeria Fionda and Giuseppe Pirró</i>	
Building a Large Scale Knowledge Base from Chinese Wiki Encyclopedia	80
<i>Zhichun Wang, Zhigang Wang, Juanzi Li, and Jeff Z. Pan</i>	
Dynamic <i>Is-a</i> Hierarchy Generation System Based on User's Viewpoint	96
<i>Kouji Kozaki, Keisuke Hihara, and Riiciro Mizoguchi</i>	
Mid-Ontology Learning from Linked Data	112
<i>Lihua Zhao and Ryutaro Ichise</i>	
An Ontological Formulation and an OPM Profile for Causality in Planning Applications	128
<i>Irene Celino and Daniele Dell'Aglío</i>	
A New Matchmaking Approach Based on Abductive Conjunctive Query Answering	144
<i>Jianfeng Du, Shuai Wang, Guilin Qi, Jeff Z. Pan, and Yong Hu</i>	
GeniUS: Generic User Modeling Library for the Social Semantic Web . . .	160
<i>Qi Gao, Fabian Abel, and Geert-Jan Houben</i>	
Enhancing Source Selection for Live Queries over Linked Data via Query Log Mining	176
<i>Yuan Tian, Jürgen Umbrich, and Yong Yu</i>	

Semantic Caching for Semantic Web Applications	192
<i>Mengdong Yang and Gang Wu</i>	
Evaluating Graph Traversal Algorithms for Distributed SPARQL Query Optimization	210
<i>Xin Wang, Thanassis Tsiropanis, and Hugh C. Davis</i>	
BipRank: Ranking and Summarizing RDF Vocabulary Descriptions	226
<i>Gong Cheng, Feng Ji, Shengmei Luo, Weiyi Ge, and Yuzhong Qu</i>	
Operational Semantics for SPARQL Update	242
<i>Ross Horne, Vladimiro Sassone, and Nicholas Gibbins</i>	
Knowledge-Driven Diagnostic System for Traditional Chinese Medicine	258
<i>Peiqin Gu and Huajun Chen</i>	
LODDO: Using Linked Open Data Description Overlap to Measure Semantic Relatedness between Named Entities	268
<i>Wenlei Zhou, Haofen Wang, Jiansong Chao, Weinan Zhang, and Yong Yu</i>	
What Should I Link to? Identifying Relevant Sources and Classes for Data Linking	284
<i>Andriy Nikolov, Mathieu d’Aquin, and Enrico Motta</i>	
Interacting with Linked Data via Semantically Annotated Widgets	300
<i>Armin Haller, Tudor Groza, and Florian Rosenberg</i>	
RDFA ² : Lightweight Semantic Enrichment for Hypertext Content	318
<i>Xi Bai, Ewan Klein, and Dave Robertson</i>	
GoRelations: An Intuitive Query System for DBpedia	334
<i>Lushan Han, Tim Finin, and Anupam Joshi</i>	
Proposed SKOS Extensions for BioPortal Terminology Services	342
<i>Cui Tao, Natalya F. Noy, Harold R. Solbrig, Nigam H. Shah, Mark A. Musen, and Christopher G. Chute</i>	
Learning Complex Mappings between Ontologies	350
<i>Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu</i>	
Discovering and Ranking New Links for Linked Data Supplier	358
<i>Nansu Zong, Sungkwon Yang, Hyun Namgoong, and Hong-Gee Kim</i>	
Probabilistic Multi-Context Systems	366
<i>Marco Sotomayor, Kewen Wang, Yidong Shen, and John Thornton</i>	
Web Schema Construction Based on Web Ontology Usage Analysis	376
<i>Jamshaid Ashraf and Maja Hadzic</i>	

Building Linked Open University Data: Tsinghua University Open Data as a Showcase	385
<i>Yuanchao Ma, Bin Xu, Yin Bai, and Zonghui Li</i>	
An Abductive CQA Based Matchmaking System for Finding Renting Houses	394
<i>Jianfeng Du, Shuai Wang, Guilin Qi, Jeff Z. Pan, and Che Qiu</i>	
An Ontological Approach to Oracle BPM	402
<i>Jean Prater, Ralf Mueller, and Bill Beaugregard</i>	
Shining Light on Complex RDF Data through Advanced Data Visualization	411
<i>Francois Bertault, Wendy Feng, Austris Krastins, Liangrong Yi, and Arturs Verza</i>	
OntoRevision: A Plug-in System for Ontology Revision in Protégé	417
<i>Nathan Cobby, Kewen Wang, Zhe Wang, and Marco Sotomayor</i>	
An Efficient Approach to Debugging Ontologies Based on Patterns	425
<i>Qiu Ji, Zhiqiang Gao, Zhisheng Huang, and Man Zhu</i>	
Author Index	435

A Method of Contrastive Reasoning with Inconsistent Ontologies

Jun Fang¹, Zhisheng Huang², and Frank van Harmelen²

¹ School of Automation, Northwestern Polytechnical University, China
junfang@nwpu.edu.cn

² Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
{huang, Frank.van.Harmelen}@cs.vu.nl

Abstract. Contrastive reasoning is the reasoning with contrasts which are expressed as contrary conjunctions like the word "but" in natural language. Contrastive answers are more informative for reasoning with inconsistent ontologies, as compared with the usual simple Boolean answer, i.e., either "yes" or "no". In this paper, we propose a method of computing contrastive answers from inconsistent ontologies. The proposed approach has been implemented in the system CRION (Contrastive Reasoning with Inconsistent ONtologies) as a reasoning plug-in in the LarKC (Large Knowledge Collider) platform. We report several experiments in which we apply the CRION system to some realistic ontologies. This evaluation shows that contrastive reasoning is a useful extension to the existing approaches of reasoning with inconsistent ontologies.

1 Introduction

1.1 Motivation

Contrastive reasoning is the reasoning with contrasts which are expressed as contrary conjunctions like the word "but" in natural language [1,2,3]. For instance, in real life, one would say that "all cars are polluting, *but* hybrid cars are not polluting"; or one would say that "The conference will be held in Holland, *but* not in Amsterdam". The first example expresses an exception that contradicts a general rule; the second example is contrary to a general expectation that one may have (namely that conferences in Holland are generally held in Amsterdam).

There exist some previous works on contrastive reasoning [1,2,3]. These previous works consider contrastive reasoning as a supplement for non-monotonic reasoning: they use non-monotonic reasoning to compute the all implications, and then determine the contrasts among them. Compared with normal Boolean query answering, contrastive reasoning gives users not only an answer to the original query, but also some *contrastive answers*.

Such contrastive reasoning has two main goals:

- *Avoidance of misleading information by extending the answer.* Contrastive answers provide not only an answer to the original query, but also some relevant contrasting answers. In our introductory example, the simple answer that all cars are polluting is misleading because hybrid cars are an exception to this rule.
- *Effective influence with surprising answers.* A psychologically effective influence can be achieved by providing an additional answer that is unexpected. In our introductory conference example, the contrastive answer that the conference is not in Amsterdam is surprising against the background expectation that all conferences in Holland will be in Amsterdam.

Contrastive reasoning exposes the contradiction that exists either between a knowledge base and external expectations (as in the conference example), or contradictions between different parts of the knowledge base (as in the polluting cars example). Because of this, contrastive reasoning is also very useful for reasoning with inconsistent ontologies, because it does not simply respond to queries with a Boolean answer of either “yes” or “no”, but also provides an informative answer with some “surprising” information.

Reasoning with inconsistent ontologies is a particularly important research topic in the Semantic Web for several reasons: i) Integration of existing ontologies easily leads to an inconsistency. ii) It may be ineffective or even impossible to repair inconsistencies before reasoning as the inconsistent ontologies may be too large or we may not have the right to repair inconsistencies in imported ontologies. iii) Ontologies may change at a high frequency and hence do not allow for any meaningful repair. In this paper, we will therefore focus on reasoning with inconsistent ontologies.

1.2 Simple Example

We consider a fragment of the well known MadCow ontology shown in Table 1¹, in which *MadCow* is defined as a *Cow* which eats brains of *Sheep* and *Cow* is defined as a *Vegetarian*, which leads to an inconsistency in the ontology.

Table 1. Fragment of the MadCow ontology

$Cow \sqsubseteq Vegetarian$	$MadCow(the_MadCow)$
$MadCow \sqsubseteq Cow \sqcap \exists eat.((\exists partof.Sheep) \sqcap Brain)$	
$Sheep \sqsubseteq Animal$	$Vegetarian \sqsubseteq \forall eat.\neg Animal$
$Vegetarian \sqsubseteq Animal \sqcap \forall eat.\neg(\exists partof.Animal)$	

When we ask “Is Cow a Vegetarian?”, the current methods will answer “yes”. However, using contrastive reasoning, the answer “yes, *but* MadCow is not a Vegetarian” is more informative. The latter answer has a touch of contrast (or surprise), which would provide more instructive information for users.

¹ We add $MadCow(the_MadCow)$ to expose the inconsistency.

1.3 Structure and Contributions of This Paper

We have presented the initial framework of contrastive reasoning in [4]. In this paper, we propose a method of computing contrastive answers from inconsistent ontologies. We introduce contrastive reasoning in the general setting of First-order Logic (FOL). The proposed approach has been implemented in the system CRION as a reasoning plug-in in the LarkC platform [2]. We will report our experiments of applying the proposed approach to some realistic ontologies. The experiments show that contrastive reasoning is a useful form of reasoning with inconsistent ontologies.

Summarizing, the main contributions of this paper are (1) a general approach of contrastive reasoning; (2) a method of computing contrastive answers; (3) the implementation of the CRION system that computes contrastive answers; and (4) evaluation of CRION, using human subjects to score the effectiveness of contrastive answers to queries.

This paper is organized as follows: Section 2 presents a general approach of contrastive reasoning with inconsistent ontologies. Section 3 explores how to compute contrastive answers in FOL. Section 4 discusses the implementation of CRION, reports the experiments of CRION with several inconsistent ontologies and presents the evaluation of CRION. After a discussion of related work in Section 5, the last section includes conclusions and future work.

2 Formalization of Contrastive Reasoning

2.1 Nonstandard Entailment for Inconsistent ontologies

The classical entailment in logics is *explosive*: any formula is a logical consequence of a contradiction. Therefore, conclusions drawn from an inconsistent knowledge base by classical inference may be completely meaningless. The general task of any system that reasons with inconsistent ontologies is: given an inconsistent ontology, return *meaningful* answers to queries. In [5], a general framework of reasoning with inconsistent ontologies has been developed. In that framework, an answer is “meaningful” if it is supported by a selected consistent subset of the inconsistent ontology, while its negation is not supported by the selected subset. PION is a system for reasoning with inconsistent ontologies, which can return such meaningful answers [5]. In the following, we will use the notation \models to denote the standard entailment, and the notation \approx to denote a nonstandard entailment.

Definition 1 (Nonstandard Entailment \approx [5]). *A nonstandard entailment \approx satisfies the following two requirements:*

1. *Soundness.* A nonstandard entailment \approx is sound if the formulas that follow from an inconsistent ontology \mathcal{O} follow from a consistent subset of \mathcal{O} using classical reasoning: $\mathcal{O} \approx \alpha \Rightarrow \exists(\mathcal{O}' \subseteq \mathcal{O})(\mathcal{O}' \not\perp \text{ and } \mathcal{O}' \models \alpha)$.

² <http://www.larkc.eu>

2. *Meaningfulness.* An answer given by an inconsistency reasoner is meaningful iff it is consistent and sound. Namely, it requires not only the soundness condition, but also $\mathcal{O} \approx \alpha \Rightarrow \mathcal{O} \not\approx \neg\alpha$. A nonstandard entailment \approx is said to be meaningful iff all of the answers are meaningful.

Properties of \approx are similar to those of the standard entailment \models . However, there is an important exception. Given an inconsistent \mathcal{O} and two formulas α and β with $\mathcal{O} \approx \alpha$ and $\mathcal{O} \approx \beta$, we cannot always conclude $\mathcal{O} \approx \alpha \wedge \beta$. One reason for it is that the selected subset that supports $\mathcal{O} \approx \alpha$ may differ from the selected subset that supports $\mathcal{O} \approx \beta$, while the union of the two subsets may be inconsistent; another reason is that $\alpha \wedge \beta$ may be a contradiction.

2.2 Contrastive Answers

Using the previous definition of nonstandard entailment, we can now define our central notion of *contrastive answers*. Informally, a contrastive answer contains three parts:

- **Original formula.** A formula which answers the original query³;
- **Contrastive formula.** A formula which contrasts with the original answer formula;
- **Clarification formula** A formula that explains the reason why the contradiction occurs. The clarification formula need not (but may) be implied by the ontology. In some application scenarios, the clarification formulas may be omitted in the query answer if the user does not require an explanation of contrastive answers.

In the MadCow example, when considering the query “Is Cow a Vegetarian?”, “Cow is a Vegetarian” is the original answer to the query, “MadCow is not a Vegetarian” is a contrastive formula, while “the_MadCow is a MadCow and MadCow is a Cow” is a clarification formula which explains why “Cow is a Vegetarian” and “MadCow is not a Vegetarian” are contrastive. This leads us to the formal definition of contrastive answers⁴:

Definition 2 (Contrastive Answer). Given an inconsistent ontology \mathcal{O} , a contrastive answer $\mathcal{O} \approx \alpha$ but γ although β contains the following parts: an original formula α , a contrastive formula γ , and a clarification formula β , such

³ Note that formulas in this paper mean First-order Logic formulas. Our work is built on FOL. Without loss of generality, a Description Logic axiom can be transformed into a (conjunctive) FOL formula. Thus, in the following, we will consider only a single formula.

⁴ In this paper, we focus on the approach of reasoning with inconsistent ontologies, in which a clarification formula is derivable from the ontology. We leave the cases of the clarification formula as an expectation (like that in the conference example) for future work.

that: $\mathcal{O} \vDash \alpha$, $\mathcal{O} \vDash \beta$ and $\mathcal{O} \vDash \gamma$, $\alpha \wedge \beta$ is not a contradiction, $\gamma \wedge \beta$ is not a contradiction, but $\alpha \wedge \beta \wedge \gamma$ is a contradiction⁵.

Sometimes it is not necessary to state the clarification formula explicitly in a contrastive answer. That leads to the following definition.

Definition 3 (Contrastive Answer without Explanation). *Given an inconsistent ontology \mathcal{O} , $\mathcal{O} \vDash \alpha$ but γ is a contrastive answer without explanation if there exists a formula β such that $\mathcal{O} \vDash \alpha$ but γ although β is a contrastive answer.*

The definitions above imply that contrastive answers have a nice exchange property. Namely, more contrastive answers can be obtained by exchanging the original formula, the contrastive formula and the clarification formula.

For instance, in the MadCow example, “Cow is a Vegetarian”, but “Mad-Cow is not a Vegetarian”, although “the_MadCow is a MadCow and MadCow is a Cow” is a contrastive answer. It is easy to observe that the symmetric answers such as “Madcows is not a vegetarian”, but “the_MadCow is a Mad-Cow and MadCow is a Cow”, although “Cow is a vegetarian” are contrastive answers.

Proposition 1 (Exchange Property of Contrastive Answers). *For an inconsistent ontology \mathcal{O} and three formulas α , β , γ , the following hold:*

- Exchange: $\mathcal{O} \vDash \alpha$ but γ although $\beta \Rightarrow \mathcal{O} \vDash \gamma$ but α although β
- Conditional Lifting: $\mathcal{O} \vDash \alpha$ but γ although β and $\alpha \wedge \gamma$ is not a contradiction $\Rightarrow \mathcal{O} \vDash \beta$ but γ although α
- Conditional Shifting: $\mathcal{O} \vDash \alpha$ but γ although β and $\alpha \wedge \gamma$ is not a contradiction $\Rightarrow \mathcal{O} \vDash \alpha$ but β although γ

Proof. It can be easily proved by using the definition of contrastive answer. 1) If $\mathcal{O} \vDash \alpha$ but γ although β , then $\mathcal{O} \vDash \alpha$ and $\mathcal{O} \vDash \beta$ and $\mathcal{O} \vDash \gamma$ and $\alpha \wedge \beta$ is not a contradiction and $\gamma \wedge \beta$ is not a contradiction and $\alpha \wedge \beta \wedge \gamma$ is a contradiction, according to definition 2, $\mathcal{O} \vDash \gamma$ but α although β . Conditional symmetry properties are proved in a similar way.

These exchange properties do not mean that α , β and γ do not differ in any way. Although they can be formally interchanged in the above way, such an interchange implies a change in the epistemological status of the formula: The original formula α is the answer to the original query. Thus, it is considered to be the most important one. The contrastive formula γ is an additional answer. The clarification formula β provides some information to explain the reason why the contradiction occurs, which may be ignored if an explanation is not necessary.

⁵ A formula is a contradiction iff there does not exist a model which can satisfy the formula, an ontology is inconsistent iff there does not exist a model which can satisfy all formulas in the ontology.

Besides the exchange property above, contrastive answers also have the expansion property: the formula α in the contrastive answer can be expanded with other formula α' if the conjunction $\alpha \wedge \alpha'$ is an answer of \models .

Proposition 2 (Expansion Property of Contrastive Answers). *For an inconsistent ontology \mathcal{O} , and three formulas α, β, γ , the following hold:*

- $\mathcal{O} \models \alpha$ **but** γ **although** β and $\alpha \wedge \alpha' \wedge \beta$ is not a contradiction and $\mathcal{O} \models \alpha \wedge \alpha'$
 $\Rightarrow \mathcal{O} \models \alpha \wedge \alpha'$ **but** γ **although** β
- $\mathcal{O} \models \alpha$ **but** γ **although** β and $\alpha \wedge \beta \wedge \beta'$ is not a contradiction and $\beta \wedge \beta' \wedge \gamma$
is not a contradiction and $\mathcal{O} \models \beta \wedge \beta' \Rightarrow \mathcal{O} \models \alpha$ **but** γ **although** $\beta \wedge \beta'$
- $\mathcal{O} \models \alpha$ **but** γ **although** β and $\beta \wedge \gamma \wedge \gamma'$ is not a contradiction and $\mathcal{O} \models \gamma \wedge \gamma'$
 $\Rightarrow \mathcal{O} \models \alpha$ **but** $\gamma \wedge \gamma'$ **although** β

Proof 1. *If $\mathcal{O} \models \alpha$ **but** γ **although** β , then $\mathcal{O} \models \alpha$ and $\mathcal{O} \models \beta$ and $\mathcal{O} \models \gamma$ and $\alpha \wedge \beta \wedge \gamma$ is a contradiction, so $\alpha \wedge \alpha' \wedge \beta \wedge \gamma$ is also a contradiction. Furthermore, $\alpha \wedge \alpha' \wedge \beta$ is not contradictions. Since $\mathcal{O} \models \alpha \wedge \alpha'$, according to Definition 2, $\mathcal{O} \models \alpha \wedge \alpha'$ **but** γ **although** β . Other situations can be proved similarly.*

3 Computing Contrastive Answers

In this section, we will propose a method of obtaining contrastive answers. Given an inconsistent ontology \mathcal{O} and a closed FOL formula α , if $\mathcal{O} \models \alpha$, how can we obtain related contrastive answers $\mathcal{O} \models \alpha$ **but** γ **although** β .

Our approach of computing contrastive answers is an extension to the method for reasoning with inconsistent ontologies proposed in [5]. To make this paper self contained, we will first give a brief overview of the general approach of reasoning with inconsistent ontologies, which is developed in [5,6].

3.1 The PION Approach

Selection functions are central in the PION approach of reasoning with inconsistent ontologies. It is used to determine which consistent subsets of an inconsistent ontology should be considered during the reasoning process. The selection function can either be syntactic, e.g. using a syntactic relevance measure, or can be based on semantic relevance, such as using the co-occurrence of terms in search engines like Google [7].

Given an ontology (i.e., a formula set) \mathcal{O} and a query α , a selection function s returns a subset of \mathcal{O} at each step $k > 0$. Let \mathbf{L} be the ontology language, which is denoted as a formula set. A selection function s is then a mapping $s : \mathcal{P}(\mathbf{L}) \times \mathbf{L} \times N \rightarrow \mathcal{P}(\mathbf{L})$ such that $s(\mathcal{O}, \alpha, k) \subseteq \mathcal{O}$.

A formula ϕ is syntactic relevant to a formula set Σ iff there exists a formula $\psi \in \Sigma$ such that ϕ and ψ are directly relevant. We can use the relevance relation above to define a selection function as follows:

- $s(\Sigma, \phi, 0) = \emptyset$
- $s(\Sigma, \phi, 1) = \{\psi \in \Sigma \mid \phi \text{ and } \psi \text{ directly relevant}\}$
- $s(\Sigma, \phi, k) = \{\psi \in \Sigma \mid \psi \text{ is directly relevant to } s(\Sigma, \phi, k - 1)\}$ for $k > 1$

In this paper, we use the syntactic method [5] to measure relevance between formulas. Two formula ϕ and ψ are directly syntactically relevant iff there is a common name which appears in both formulas. Although the syntactic-relevance-based selection function is specific and seems to be simple, the experiments show that even this simple selection function can obtain intuitive results in most cases for reasoning with inconsistent ontologies [5]. Furthermore, our approach of contrastive reasoning is independent of any specific selection function, because the syntactic-relevance-based selection function can be replaced with any other kinds of selection functions, like one with Normalized Google Distance [7].

The general strategy for reasoning with inconsistent ontologies is: given the syntactic selection function, we select a consistent subset from an inconsistent ontology. Then we apply standard reasoning on the selected subset to find meaningful answers. If a satisfying answer cannot be found, we use the selection function to extend the selected set for further reasoning. If an inconsistent subset is selected, we apply “over-determined processing” (ODP) [5]. One of the ODP strategies is to find a maximal consistent subset of the selected set. If the (firstly selected) maximal consistent subset entails the query, the algorithm will return ‘yes’, otherwise it will return ‘no’. A linear extension strategy with ODP for the evaluation of a query ‘ $\mathcal{O} \models \alpha$?’ is described in Algorithm 1.

Algorithm 1. Linear extension strategy for evaluating $\mathcal{O} \models \alpha$

```

1:  $\Omega := \emptyset$ 
2:  $k := 0$ 
3: repeat
4:    $k := k + 1$ 
5:    $\Omega' := s(\mathcal{O}, \alpha, k)$ 
6:   if  $\Omega' \subseteq \Omega$  then
7:     return  $\mathcal{O} \not\models \alpha$ 
8:   end if
9:   if  $\Omega'$  inconsistent then
10:     $\Omega'' := \text{maximal\_consistent\_subontology}(\Omega')$ 
11:    if  $\Omega'' \models \alpha$  then
12:      return  $\mathcal{O} \models \alpha$ 
13:    else
14:      return  $\mathcal{O} \not\models \alpha$ 
15:    end if
16:   end if
17:    $\Omega := \Omega'$ 
18: until  $\Omega' \models \alpha$ 
19: return  $\mathcal{O} \models \alpha$ 

```

3.2 The CRION Approach

In the following, we propose an algorithm for obtaining contrastive answers, based on the PION approach described above. From the definition of contrastive answers, the conjunction of the original formula α , the contrastive formula γ , and the clarification formula β must lead to a contradiction, i.e., $\{\alpha, \beta, \gamma\} \models \perp$. That means that, given an original answer α which is obtained by using the PION approach, we can try to obtain the contrastive formula and the clarification formula, by considering a *minimal inconsistent set* which contains α . A minimal inconsistent set is a minimal formula set that explains the inconsistency of an inconsistent ontology.

Definition 4 (Minimal Inconsistent Set (MIS)). *Given an inconsistent ontology \mathcal{O} , a formula set \mathcal{O}' is a minimal inconsistent set (MIS) of \mathcal{O} iff it satisfies the conditions: i) $\mathcal{O}' \subseteq \mathcal{O}$, ii) $\mathcal{O}' \models \perp$, and iii) $\forall \mathcal{O}'' (\mathcal{O}'' \subset \mathcal{O}' \Rightarrow \mathcal{O}'' \not\models \perp)$.*

A minimal consistent set is akin to a justification [8], which is a minimal formula set to explain the entailment. In [9], the justification method [8] is used to compute minimal consistent sets in inconsistent ontologies. In this paper, we are interested in computing one MIS which includes one specified formula, i.e. the original formula which needs to be included in the minimal inconsistent set of the inconsistent ontology.

Algorithm 2 describes the process for computing such a specific MIS. The algorithm is taken from algorithm 2 in [10] with a few modifications, it applies binary search to quickly find a MIS. The algorithm partitions the ontology into two halves, and checks whether one of them is inconsistent. If yes, it goes to the recursion on that half, throwing away half of the axioms in one step. Otherwise, essential axioms are in both halves. In this case, the algorithm goes on the recursion on each half, using the other half as the support set.

The main idea of the CRION approach is to extend the linear extension strategy of the PION approach until the selected set $\Omega \cup \{\alpha\}$ is inconsistent. We have then obtained a minimal inconsistent set which includes α by using algorithm 2. Then we pick up a clarification formula β in the MIS and construct a contrastive formula γ from the MIS. A straightforward approach to construct the formula γ is to take the conjunction of some subset of the MIS. We call that approach *Contrastive Answer by Conjunction* (CAC).

Given an original answer $\mathcal{O} \approx \alpha$ which is obtained as the selected set $s(\mathcal{O}, \alpha, k)$ at step k , the CAC algorithm for obtaining contrastive answers is described in Algorithm 3. In the algorithm, we use S_c to denote the set of returned contrastive answers. If the S_c is \emptyset , then that means that there are no contrastive answers for the query.

The algorithm consists of the three main steps: i) extend the selected set until it becomes inconsistent, ii) find a minimal inconsistent set which includes α , and iii) construct the clarification formula β and the contrastive formula γ . It is not hard to prove the following proposition.

Proposition 3 (Soundness of the CAC Algorithm). *The contrastive answers obtained in Algorithm 3 are sound.*

Algorithm 2. *mis_binarySearch*(S, \mathcal{O}, α)**Assume:** $|\mathcal{O}| > 1$ in the initial step

```

1: if  $|\mathcal{O}| == 1$  then
2:   return  $\mathcal{O}$ 
3: end if
4:  $S_1, S_2 := \text{halve}(\mathcal{O})$ 
5: if  $S \cup S_1$  is inconsistent then
6:   return mis_binarySearch( $S, S_1, \alpha$ )
7: else if  $S \cup S_2$  is inconsistent then
8:   return mis_binarySearch( $S, S_2, \alpha$ )
9: end if
10:  $S'_1 := \text{mis\_binarySearch}(S \cup S_2, S_1, \alpha)$ 
11:  $S'_2 := \text{mis\_binarySearch}(S \cup S'_1, S_2, \alpha)$ 
12: if  $\alpha \in S'_1 \cup S'_2$  then
13:   return  $S'_1 \cup S'_2$ 
14: end if
15: return  $\emptyset$ 

```

Algorithm 3. Contrastive Answers by Conjunction (CAC)

```

1:  $S_c := \emptyset$ 
2:  $j := k$ 
3:  $\Omega := s(\mathcal{O}, \alpha, j)$ 
4: while  $\Omega \cup \{\alpha\}$  consistent do
5:    $j := j + 1$ 
6:   if  $s(\mathcal{O}, \alpha, j) \subseteq \Omega$  then
7:     return  $\emptyset$ 
8:   end if
9:    $\Omega := s(\mathcal{O}, \alpha, j)$ 
10: end while
11:  $\Omega' := \text{mis\_binarySearch}(\emptyset, \Omega \cup \{\alpha\}, \alpha)$ 
12: for  $\rho \in \Omega'$  do
13:   if  $\{\alpha, \rho\}$  consistent then
14:      $\beta := \rho$ 
15:      $\gamma := \bigwedge(\Omega' - \{\alpha, \beta\})$ 
16:     if  $\{\beta, \gamma\}$  consistent and  $\mathcal{O} \approx \gamma$  then
17:        $S_c = S_c \cup \{\mathcal{O} \approx \alpha \text{ but } \gamma \text{ although } \beta\}$ 
18:     end if
19:   end if
20: end for
21: return  $S_c$ 

```

Proof. If the algorithm returns an answer $\mathcal{O} \approx \alpha$ **but** γ **although** β in S_c , we want to prove that α **but** γ **although** β is indeed a contrastive answer. From the given condition, we already have that $\mathcal{O} \approx \alpha$. Since any formula is considered to be always the most (syntactically or semantically) relevant to itself, we have $\mathcal{O} \approx \rho$ for any formula ρ such that $\rho \in \mathcal{O}$ and $\neg\rho \notin \mathcal{O}$ ⁶. Thus, we have $\mathcal{O} \approx \beta$. From the algorithm, we have $\mathcal{O} \approx \gamma$. It is easy to see that $\alpha \wedge \beta$ is not a contradiction and $\beta \wedge \gamma$ is not a contradiction, because $\{\alpha, \beta\}$ is consistent and $\{\beta, \gamma\}$ is consistent. Furthermore, from the algorithm, we know that $\alpha \wedge \beta \wedge \gamma$ is a contradiction because of the inconsistency of Ω' . Thus, we have the conclusion.

It is easy to see that the CAC algorithm can always terminate and that its computational cost is not significantly increased, compared with the complexity of the existing approaches in reasoning with inconsistent ontologies.

In the CAC algorithm, the contrastive formula is a conjunction of formulas selected from the ontology. We are more interested in contrastive formulas which are implied by a consistent subset of the minimal inconsistent set, rather than its subformulas which are contained by the ontology explicitly. Those contrastive answers may be obtained through the CAC approach by using the exchange property in Proposition 11.

A more general approach to obtaining those contrastive answers is to consider a contrastive formula γ which is a non-trivial consequence of the selected set, i.e., $\gamma \in Cn(\Omega' - \{\alpha, \beta\}) - Cn(\emptyset)$ in the algorithm⁷. We call that approach *Contrastive Answer by Logical Consequence* (CALC). The CALC algorithm is a revision of the CAC algorithm by constructing a formula γ which satisfies the condition above and inserting a step of contradiction checking for $\alpha \wedge \beta \wedge \gamma$ before Step 16 in Algorithm 3. There are various strategies to construct a contrastive formula γ for the CALC approach (e.g., depth-first search, breadth-first search, and best-first search). We will leave the investigation of variant CALC algorithms for future work.

4 Implementation and Evaluation

4.1 Implementation

We have implemented the prototype of CRION⁸ as a reasoning plug-in in the LarKC Platform. by using Pellet⁹ and OWLAPI¹⁰. Given a query answer in

⁶ In this paper we consider only the ontology \mathcal{O} in which there exists no a formula ρ such that $\rho \in \mathcal{O}$ and $\neg\rho \in \mathcal{O}$. Namely, the inconsistency in \mathcal{O} is not explicit. This condition is generally satisfied in practice.

⁷ Cn is a consequence operator such that $Cn(\mathcal{O}) = \{\varphi | \mathcal{O} \models \varphi\}$. $Cn(\emptyset)$ is the tautology set, which is considered to be trivial.

⁸ https://larkc.svn.sourceforge.net/svnroot/larkc/branches/Release_1.1_candidate/plugins/reason/CRION/

⁹ <http://clarkparsia.com/pellet/>

¹⁰ <http://owlapi.sourceforge.net/>

an inconsistent Description Logic (DL) ontology, CRION calculates contrastive answers based on the CAC approach. CRION uses PION¹¹ to compute the non-standard entailment in an inconsistent ontology. Syntax-based selection function defined in [5] is used in PION.

4.2 Evaluation

We have tested the CRION prototype by applying it to inconsistent ontologies. For that test, we selected two group of ontologies. The first group are several ontologies from the TONES ontology repository¹². Those ontologies are selected, because i) they are inconsistent, ii) Pellet supports them, and iii) we are familiar with the domains of those ontologies.

In order to test the run-time performance of our method in large scale ontologies, we construct the second group of ontologies by modifying from the LUBM¹³ benchmark ontology by inserting a specified number of conflicts using the Injector tool described in [11], where a conflict is a set of axioms violating a functional role restriction or a disjointness constraint. By LUBM-Lite $n+m$ we mean an LUBM-Lite ontology with assertional axioms of n universities and with m inserted conflicts. The profiles of the selected ontologies are shown in Table 2.

Table 2. Information about ontologies

Ontology	Syntax	#Cons	#Roles	#Inds	#Axioms	#MISs
MadCow	$\mathcal{ALCHQIN}(\mathbf{D})$	54	16	67	143	1
Pizza	\mathcal{SHIQN}	101	8	106	818	2
Economy	$\mathcal{ALCH}(\mathbf{D})$	338	45	818	1,947	51
Transportation	$\mathcal{ALCH}(\mathbf{D})$	446	89	629	1,786	62
LUBM-Lite $1+20$	$\mathcal{SHLF}(\mathbf{D})$	100	39	17,190	100,869	20
LUBM-Lite $2+40$				38,377	230,408	30
LUBM-Lite $4+80$				78,653	478,740	80
LUBM-Lite $8+160$				163,690	1,002,095	160
LUBM-Lite $16+320$				341,557	2,096,008	320

As the original formulas in a contrastive answer is related to a minimal inconsistent set, for each inconsistent ontology, we select the testing queries from the union of all minimal inconsistent sets calculated by using the explanation method in Pellet¹⁴. We evaluate the approach of contrastive reasoning with respect to the following three aspects:

¹¹ <http://wasp.cs.vu.nl/sekt/pion/>

¹² <http://owl.cs.manchester.ac.uk/repository/>, we expose their inconsistencies by adding a concept assertion for every named concept, i.e., $Con(the_Con)$.

¹³ <http://swat.cse.lehigh.edu/projects/lubm/>

¹⁴ It uses the method of `org.mindswap.pellet.owlapi.Reasoner.getExplanation()`.

- **Frequency:** Given an inconsistent ontology, how often can we obtain a contrastive answer? We measure the frequency by counting the amount of contrastive answers.
- **Usability:** Does the contrastive reasoning really achieve the main goals? Is it really useful or not? We evaluate the usability by examining the results with respect to the two main criteria: i) does it help avoiding misleading information and ii) does it improve the effective influence of the answer? Of course these criteria are necessarily “soft” in nature: they cannot be measured by any formal means, but must be subjected to human judgment.
- **Performance:** Is the contrastive reasoning computationally expensive or not? We evaluate the performance by examining the run-time performance of computing contrastive answers.

Frequency. Columns 2, 3 and 4 of Table 3 show that contrastive answers (CAs) occur frequently for inconsistent ontologies. For the MadCow ontology in which there is only one minimal inconsistent set, we have at least 25 contrastive answers for 5 queries. The total numbers of contrastive answers rise to hundreds (408) for the inconsistent ontologies which have dozens (51) of minimal inconsistent sets. For the second group of ontologies, the average numbers of contrastive answers are stable (around 3). There appear to be a reasonable number, and reasonably constant number, of contrastive answers per query across the tested ontologies (1-5). Moreover, note that the total number of contrastive answers will be at least doubled at by using the exchange and expansion property.

Table 3. Evaluation of number of contrastive answers by using CAC

Ontology	Number of queries	Total number of CAs	Average Number of CAs
MadCow	5	25	5.0
Pizza	8	33	4.1
Economy	160	408	2.55
Transportation	159	200	1.25
LUBM-Lite1 ₊₂₀	57	171	3.0
LUBM-Lite2 ₊₄₀	115	359	3.12
LUBM-Lite4 ₊₈₀	207	631	3.05
LUBM-Lite8 ₊₁₆₀	387	1157	3.04
LUBM-Lite16 ₊₃₂₀	703	2126	3.02

Usability. Five researchers score the computed contrastive answers of ontologies in the group one, based on the two main goals, which are discussed in Section 1, namely, avoiding misleading information and improving effective influence of the answer. Those two criteria are marked based on a five point scale: 0=valueless, 1=little value, 2=some value, 3=average value, 4=high value, and 5=perfect value. The average scores are listed in the second column and the third column of

Table 4. For the degree of avoiding misleading information, the scores range from 3.4 (= “average value”) to 4.2 (= “high value”). That means that the contrastive answers are considered to be somewhat useful to avoid misleading information for the four ontologies in our test. For the degree of improving effective influence, they have a very similar range, showing the answers to be somewhat useful for improving effective influence. The fact that all the scores in our small experiment are > 3 indicates that the approach of contrastive reasoning might indeed be useful for reasoning with inconsistent ontologies.

Table 4. Evaluation of value of contrastive answers

Ontology	Average value on	
	avoiding misleading information	improving effective influence
MadCow	4.2	4.0
Pizza	3.6	3.8
Economy	3.4	3.5
Transportation	3.7	3.5

Run-Time Performance. All the experiments are carried out on an ordinary PC (with a 2.60 GHz Pentium-4 processor and 2GB of physical memory, where the maximum Java heap size was set to 1280MB for applying Pellet). The maximal, minimal and average computation time (in seconds) for a query by using the CAC approach are shown in columns 2, 3 and 4 of Table 5.

The experimental results show that for all test ontologies in the first group, the CAC computation time for computing contrastive answers for a query is limited to a small number of seconds. The maximal computation time is just a few seconds (1.1s), the minimal computation time goes even to several milliseconds (0.007s), and the average computation time is less than one second (0.26s). For the large ontologies in the second group, the minimal computation time is only one second, the maximal computation time is less than several minutes when there are millions of axioms in the ontologies, and the average computation time is less than dozens of seconds.

It shows that the calculation of contrastive answers by using the CAC approach does not significantly increase the computational cost. Thus, it is an efficient extension to the existing reasoners with inconsistent ontologies.

4.3 Discussion

Contrastive answers are related with minimal inconsistent sets. One contrastive reasoning method is to calculate all minimal inconsistent sets in the inconsistent ontology (offline) firstly, then compute contrastive answers from these minimal inconsistent sets. In this paper, we use the CAC method for several reasons: i) the calculation of all minimal inconsistent sets is very difficult for a large scale

Table 5. Evaluation of the run-time performance of CAC

Ontology	Max run time	Min run time	Average run time
MadCow	0.22s	0.033s	0.12s
Pizza	1.10s	0.015s	0.26s
Economy	0.44s	0.016s	0.08s
Transportation	0.51s	0.007s	0.11s
LUBM-Lite1 ₊₂₀	1.17s	0.50s	0.58s
LUBM-Lite2 ₊₄₀	16.36s	1.00s	1.77s
LUBM-Lite4 ₊₈₀	37.32s	1.00s	3.27s
LUBM-Lite8 ₊₁₆₀	100.22s	1.00s	7.44s
LUBM-Lite16 ₊₃₂₀	440.19s	1.00s	19.92s

ontology, as demonstrated in [9,12], ii) ontologies may be dynamical, especially in the Web setting, which may make an offline computation meaningless, iii) the CAC method can obtain a large amount of contrastive answers with a little cost, as shown in the experiments.

It is worth pointing out that algorithm 2 is an incomplete method, i.e., it may not find a *MIS* which includes the specific α although the *MIS* exists in the inconsistent set. The reason why we use it instead of a complete method lies in the complexity of the complete one. Generally, the complete method needs to compute all *MIS*s gradually until finding the required one, which is an *NP-hard* problem. In fact, we have already carried out some experiments to perform contrastive reasoning by using a complete *MIS* calculation method, the results show that the subprogram for calculating the specific *MIS* would not terminate in several hours for some queries with the Economy and Transportation ontologies. Hence, it is impractical.

Contrastive reasoning with DL ontologies can be extended by considering another kind of inconsistency (*incoherence* [13]), i.e., there exists an unsatisfiable named concept in the ontology. Incoherence is very important as many classical inconsistencies are caused by it, e.g., concept assertions of an unsatisfiable concept. In order to deal with incoherence, we need to consider it as well as the classical inconsistency when checking inconsistency in contrastive reasoning with DL ontologies.

5 Related Work

McGill and Klein address the differences in the use of covariation information implied by contrastive reasoning, which involves comparing the target episode to contrasting background instances [3]. Francez proposes the notion of bilogic as a logical treatment of a contrastive conjunction such as ‘but’, and argues that ordinary logics are not sufficient to express the contrastive nature of ‘but’, because of the neutral conjunction (‘and’) in classical logics [2]. Based on the contrastive operators proposed by Francez, a modal approach to contrastive logic is presented in [1]. Their contrastive logic is actually a simple modal logic which is an extension to the well-known S5 logic with Francez’s contrastive operator.

Default reasoning [14] is somehow similar to contrastive reasoning. It can be considered as a kind of reasoning service where the consequences may be derived only due to lacking evidence of the contrary. However, contrastive reasoning is different from default reasoning, because our approach is based on reasoning with inconsistent ontologies, whereas default reasoning is based on a non-monotonic logic.

6 Conclusions and Future Work

We have presented a general approach for answering queries over inconsistent ontologies by using contrastive reasoning. It is more practical for reasoning with inconsistent ontologies, as it provides not only an original answer, but also more relevant and maybe surprising answers. We have proved that obtaining contrastive answers can be achieved by a slight extension to the existing approach for reasoning with inconsistent ontologies. Furthermore, this extension does not significantly increase the computational cost. Our proposal has been implemented in the system CRION. We have reported several experiments with CRION and have presented an initial evaluation. The tests show that contrastive reasoning is useful and promising for reasoning with inconsistent ontologies.

There is a lot of the future research to be done. Here are just some of them:

- Contrastive Answer by Logical Consequence. As discussed in Section 3, various strategies for obtaining contrastive answers of the CALC approach will be very interesting to gain more useful answers.
- Reasoning with contrastive ontologies. In this paper, we have provided contrastive answers only at the query language level. We have not yet allowed to express contrastive conjunctions in the ontology level. Thus, one of the interesting future works is to reason with inconsistent ontologies that contain contrastive conjunction axioms such as “but”.

Acknowledgment. This work is supported by the European Commission under the 7th framework programme, Large Knowledge Collider (LarKC) Project (FP7-215535).

References

- [1] Meyer, J.J.C., van der Hoek, W.: A modal contrastive logic: The logic of ‘but’. *Annals of Mathematics and Artificial Intelligence* 17, 291–313 (1996)
- [2] Francez, N.: Contrastive logic. *Logic Journal of the IGPL* 3(5), 725–744 (1995)
- [3] McGill, A.L., Klein, J.G.: Counterfactual and contrastive reasoning in causal judgment. *Journal of Personality and Social Psychology* (64), 897–905 (1993)
- [4] Fang, J., Huang, Z., van Frank, H.: Contrastive reasoning with inconsistent ontologies. In: *Proceedings of 2011 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2011)*, pp. 191–194 (2011)
- [5] Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: *Proceedings of IJCAI 2005*, pp. 454–459 (2005)

- [6] Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A Framework for Handling Inconsistency in Changing Ontologies. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 353–367. Springer, Heidelberg (2005)
- [7] Huang, Z., van Harmelen, F.: Using Semantic Distances for Reasoning with Inconsistent Ontologies. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 178–194. Springer, Heidelberg (2008)
- [8] Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all Justifications of OWL DL Entailments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
- [9] Horridge, M., Parsia, B., Sattler, U.: Explaining Inconsistencies in OWL Ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS, vol. 5785, pp. 124–137. Springer, Heidelberg (2009)
- [10] Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic \mathcal{EL}^+ . In: KR-MED 2008. CEUR-WS, vol. 410 (2008)
- [11] Du, J., Shen, Y.D.: Computing minimum cost diagnoses to repair populated dl-based ontologies. In: WWW, pp. 565–574 (2008)
- [12] Du, J., Qi, G.: Decomposition-Based Optimization for Debugging of Inconsistent OWL DL Ontologies. In: Bi, Y., Williams, M.-A. (eds.) KSEM 2010. LNCS, vol. 6291, pp. 88–100. Springer, Heidelberg (2010)
- [13] Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: Proc. of AAAI 2006, pp. 1295–1300 (2006)
- [14] Brewka, G.: Preferred subtheories: An extended logical framework for default reasoning. In: Proceedings of IJCAI 1989, pp. 1043–1048 (1989)

Parallel ABox Reasoning of \mathcal{EL} Ontologies

Yuan Ren¹, Jeff Z. Pan¹, and Kevin Lee²

¹ University of Aberdeen, Aberdeen, UK

² NICTA, Australia

Abstract. In order to support the vision of the Semantic Web, ontology reasoning needs to be highly scalable and efficient. A natural way to achieve scalability and efficiency is to develop parallel ABox reasoning algorithms for tractable OWL 2 profiles to distribute the load between different computation units within a reasoning system. So far there have been some work on parallel ABox reasoning algorithms for the pD* fragment of OWL 2 RL. However, there is still no work on parallel ABox reasoning algorithm for OWL 2 EL, which is the language for many influential ontologies (such as the SNOMED CT ontology). In this paper, we extend a parallel TBox reasoning algorithm [5] for $\mathcal{ELH}_{\mathcal{R}+}$ to parallel ABox reasoning algorithms for $\mathcal{ELH}_{\perp, \mathcal{R}+}$, which also supports the bottom concept so as to model disjointness and inconsistency. In design of algorithms, we exploit the characteristic of ABox reasonings to improve parallelisation and reduce unnecessary resource cost. Our evaluation shows that a naive implementation of our approach can compute all ABox entailments of a Not-Galen⁻ ontology with about 1 million individuals and 9 million axioms in about 3 minutes.

1 Introduction

Ontologies are the knowledge infrastructures of the Semantic Web and many intelligent systems. In order to support the vision of the Semantic Web, ontology reasoning services need to be highly scalable and efficient. The modern ontology language standard, the W3C OWL Recommendation, is based on (different) description logics (DLs). In the last decades, DL reasoning technologies have been developed to support inference with ontologies. Well-optimised DL reasoning systems, such as FaCT++^[1], Hermit^[2], Pellet^[3], CEL^[4], CB^[5] and TrOWL^[6], have been implemented with different reasoning technologies. So far, these systems are designed for a single computation core. Reasoning is performed sequentially and can not be parallelised.

A natural way to achieve scalability and efficiency is to develop parallel ABox reasoning algorithms for tractable OWL 2 profiles, such as OWL 2 EL and OWL 2 RL, that can distribute the load between different computation units within a reasoning system:

¹ <http://owl.man.ac.uk/factplusplus/>

² <http://hermit-reasoner.com/>

³ <http://clarkparsia.com/pellet/>

⁴ <http://lat.inf.tu-dresden.de/systems/cel/>

⁵ <http://code.google.com/p/cb-reasoner/>

⁶ <http://trowl.eu/>

One direction is to perform parallel reasoning with a cluster of multiple computer nodes (or simply, *peers*). In Marvin [9], peers use a divide-conquer-swap strategy for RDFS inference. Weaver and Handler propose a parallel RDFS inference engine [18]; peers use an ABox partitioning approach to RDFS inference. In SAOR [4], peers use optimised template rules for join-free inference in pD^* [15]. In DRAGO [12], peers performs OWL DL reasoning under the setting of Distributed Description Logics [2], which support local reasoning at the price of sacrificing expressiveness in the links between local models. A distributed resolution algorithm for \mathcal{ALC} was proposed in [10]. Different from the aforementioned work where each peer has the same capability, in this algorithm each peer is responsible for inferences for different types of literals, making certain peer(s) become potential bottleneck and a single-point-of-failure. This issue is addressed by the authors when they extend their algorithm for \mathcal{ALCHIQ} [11]. MapReduce [3] has also been adopted to support ABox reasoning in RDFS [17], pD^* [16] as well as justifications in pD^* [19], and TBox reasoning in \mathcal{EL}^+ [7] (there is no implementation for the \mathcal{EL}^+ case yet).

Another direction is to perform parallel reasoning with multiple computation cores (or simply, *workers*) in a single computer. Soma and Prasanna [13] propose to use data-partitioning and rule-partitioning in their parallel algorithms for pD^* . Liebig and Müller [6] exploit the non-determinism introduced by disjunctions or number restrictions in the \mathcal{SHN} tableau algorithm so that multiple workers can apply expansion rules on independent alternatives. Similarly, Meissner [8] proposes parallel expansions of independent branchings in an \mathcal{ALC} tableau and experimented with 3 different strategies. Aslani and Haarslev [1] propose a parallel algorithm for OWL DL classification. Recently, Kazakov et al. [5] presented a lock-free parallel completion-based TBox classification algorithm for $\mathcal{ELH}_{\mathcal{R}^+}$.

As discussed above, there have been work on parallel ABox reasoning for the pD^* fragment of OWL 2 RL. However, there is still no work on parallel ABox reasoning algorithm for OWL 2 EL, in which many influential ontologies (such as the SNOMED CT ontology) are written. In this paper, we extend a parallel TBox reasoning algorithm [5] for $\mathcal{ELH}_{\mathcal{R}^+}$ to a parallel and lock-free ABox reasoning algorithm for $\mathcal{ELH}_{\perp, \mathcal{R}^+}$, which also supports the bottom concept so as to model disjointness and inconsistency. We exploit the different characteristic of ABox reasoning from TBox reasoning and optimise the design of completion rules and algorithms accordingly to improve parallelisation and reduce unnecessary resource cost. Particularly, we parallelise the initialisation of algorithms, separate TBox and ABox saturation, and streamline the processing of each axiom in each worker. Our evaluation shows that a naive implementation of our approach can handle combined complex TBox and large ABox efficiently.

The remainder of the paper is organised as follows: In Sec. 2 we introduce background knowledge of DLs $\mathcal{ELH}_{\mathcal{R}^+}$ and $\mathcal{ELH}_{\perp, \mathcal{R}^+}$, and the parallel $\mathcal{ELH}_{\mathcal{R}^+}$ TBox classification algorithm in [5]. In Sec. 3 we explain the technical challenges, before presenting the completion rules and parallel ABox reasoning algorithms for $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ in Sec. 4. We present an implementation of our approach and our evaluation in Sec. 5, before we conclude the paper in Sec. 6.

The proof of all lemmas and theorems are included in our online tech report at <http://www.box.net/shared/mpqgxxydhhl2bpuus5f7>.

2 Preliminary

2.1 The $\mathcal{ELH}_{\mathcal{R}_+}$ and $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ DLs

A *signature* of an ontology \mathcal{O} is a triple $\Sigma_{\mathcal{O}} = (\mathcal{CN}_{\mathcal{O}}, \mathcal{RN}_{\mathcal{O}}, \mathcal{IN}_{\mathcal{O}})$ consisting of three mutually disjoint finite sets of atomic concepts $\mathcal{CN}_{\mathcal{O}}$, atomic roles $\mathcal{RN}_{\mathcal{O}}$ and individuals $\mathcal{IN}_{\mathcal{O}}$. Given a signature, complex concepts in $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ can be defined inductively using the $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ constructors as in Table 1. $\mathcal{ELH}_{\mathcal{R}_+}$ supports all $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ constructors except \perp . Two concepts C and D are equivalent if they mutually include each other, denoted by $C \equiv D$. An ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ consists of a TBox \mathcal{T} and an ABox

Table 1. $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ Syntax and Semantics

Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \exists y. \langle x, y \rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
Roles:		
atomic role	r	$r^{\mathcal{I}}$
TBox Axioms:		
general concept inclusion (GCI):	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion (RI):	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
role transitivity:	$Trans(t)$	$t^{\mathcal{I}} \times t^{\mathcal{I}} \subseteq t^{\mathcal{I}}$
ABox Axioms:		
class assertion:	$A(a)$	$a^{\mathcal{I}} \in A^{\mathcal{I}}$
role assertion:	$r(a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$
individual equality:	$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality:	$a \not\dot{=} b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

\mathcal{A} , which are finite sets of TBox axioms and ABox axioms, respectively. $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ allows all axioms listed in Table 1. $\mathcal{ELH}_{\mathcal{R}_+}$ allows all except individual inequalities.

An *interpretation* \mathcal{I} is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function that maps each atomic concept A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each atomic role r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each individual a to an object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Interpretation function $\cdot^{\mathcal{I}}$ can be extended to complex concept as shown in Table 1.

An interpretation \mathcal{I} is a model of an ontology \mathcal{O} , written $\mathcal{I} \models \mathcal{O}$, if it satisfies all axioms of \mathcal{O} as shown in the lower part of Table 1. An axiom α is entailed by an ontology \mathcal{O} , written $\mathcal{O} \models \alpha$, iff all models of \mathcal{O} satisfy α . A concept C is *satisfiable* w.r.t. an ontology \mathcal{O} if there exists some model \mathcal{I} of \mathcal{O} such that $C^{\mathcal{I}} \neq \emptyset$. Given an ontology \mathcal{O} , we use $\sqsubseteq_{\mathcal{O}}^*$ to represent the reflexive transitive closure of RIs. It is easy to see that in an $\mathcal{ELH}_{\mathcal{R}_+} / \mathcal{ELH}_{\perp, \mathcal{R}_+}$ ontology, all of such $\sqsubseteq_{\mathcal{O}}^*$ relations can be computed in polynomial time w.r.t. the size of \mathcal{O} .

In ABox reasoning, we are particularly interested in finding all atomic types and relations of all individuals, i.e. finding all $A(a)$ s.t. $a \in \mathcal{IN}_{\mathcal{O}}$, $A \in \mathcal{CN}_{\mathcal{O}}$, $\mathcal{O} \models A(a)$

and all $r(a, b)$ s.t. $a, b \in \mathcal{IN}_{\mathcal{O}}$, $r \in \mathcal{RN}_{\mathcal{O}}$ and $\mathcal{O} \models r(a, b)$. We call such a reasoning task *ABox classification*. Computing and maintaining ABox classification results can be very useful for efficient on-line instance retrieval and/or query answering.

2.2 Parallel TBox Classification of $\mathcal{ELH}_{\mathcal{R}+}$ Ontologies

Given an ontology \mathcal{O} , TBox classification is a reasoning task that computes all inclusions over atomic concepts in \mathcal{O} . Kazakov et. al [5] proposed an approach to parallel TBox classification for $\mathcal{ELH}_{\mathcal{R}+}$. They devise a set of completion rules as follows, where $D \rightarrow E$ is used to denote the special form of GCIs where D and E are both existential restrictions. Given an $\mathcal{ELH}_{\mathcal{R}+}$ ontology \mathcal{O} that has no ABox, these rules infer $C \sqsubseteq D$ iff $\mathcal{O} \models C \sqsubseteq D$ for all C and D such that $C \sqsubseteq C \in \mathbf{S}$ and D occurs in \mathcal{O} (Theorem 1 of [5]), where \mathbf{S} is the set of axioms closed under the following inference rules.

$$\begin{array}{l}
\mathbf{R}_{\sqsubseteq} \frac{C \sqsubseteq D}{C \sqsubseteq E} : D \sqsubseteq E \in \mathcal{O} \\
\mathbf{R}_{\sqcap} \frac{C \sqsubseteq D_1 \sqcap D_2}{C \sqsubseteq D_1; C \sqsubseteq D_2} \\
\mathbf{R}_{\exists} \frac{C \sqsubseteq \exists R.D}{D \sqsubseteq D} \\
\mathbf{R}_{\top}^+ \frac{C \sqsubseteq C}{C \sqsubseteq \top} : \top \text{ occurs in } \mathcal{O} \\
\mathbf{R}_{\sqcap}^+ \frac{C \sqsubseteq D_1, C \sqsubseteq D_2}{C \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O} \\
\mathbf{R}_{\exists}^+ \frac{C \sqsubseteq D}{\exists s.C \rightarrow \exists s.D} : \exists s.D \text{ occurs in } \mathcal{O} \\
\mathbf{R}_{\mathcal{H}} \frac{D \sqsubseteq \exists r.C, \exists s.C \rightarrow E}{D \sqsubseteq E} : r \sqsubseteq_{\mathcal{O}}^* s \\
\mathbf{R}_{\mathcal{T}} \frac{D \sqsubseteq \exists r.C, \exists s.C \rightarrow E}{\exists t.D \rightarrow E} : r \sqsubseteq_{\mathcal{O}}^* t \sqsubseteq_{\mathcal{O}}^* s, \text{Trans}(t) \in \mathcal{O}
\end{array}$$

The completion rules are designed in a way that *all premises of each rule have a common concept* (the concept C in each rule), which is called a *context* of the corresponding premise axioms. Each context maintains a queue of axioms called *scheduled*, on which some completion rule can be applied, and a set of axioms called *processed*, on which some completion rule has already been applied. *An axiom can only be included in the scheduled queues and/or processed sets of its own contexts*. To ensure that multiple workers can share the queues and sets without locking them, they further devised a concurrency mechanism in which: (i) each worker will process a single context at a time and vice versa; (ii) the processing of all axioms in the scheduled queue of a context requires no axioms from the processed sets of other contexts. To realise all these, all contexts with non-empty schedules are arranged in a queue called *activeContexts*. A context can be added into the *activeContexts* queue only if it is not already in the queue.

Here are the key steps of the parallel TBox algorithm:

1. Tautology axiom $A \sqsubseteq A$ for each $A \in \mathcal{CN}_{\mathcal{O}}$ is added to the scheduled queues of A . All active contexts are added into the queue of *activeContexts*.
2. Every idle worker always looks for the next context in the *activeContexts* queue and processes axioms in its scheduled queue.
 - (a) Pop an axiom from the scheduled queue, add it into the processed set of the context.
 - (b) Apply completion rules to derive conclusions.
 - (c) Add each derived conclusion into the scheduled queue of its corresponding contexts, which will be activated if possible.

Before we extend the parallel TBox reasoning algorithm to support ABox reasoning in Sec. 4, we first discuss the challenges to deal with in parallel ABox reasoning.

3 Technical Challenges: Parallel ABox Reasoning

When we design parallel ABox classification algorithms, we need to consider the characteristic of ABox classification that distinguish it from TBox classification — the number of individuals is often much larger than the number of concepts and roles.

A naive way of doing ABox classification is to internalise the entire ABox into TBox (i.e., by converting assertions of the form $C(a)$ into $\{a\} \sqsubseteq C$ and $R(a, b)$ into $\{a\} \sqsubseteq \exists R.\{b\}$) and treat the internalised “nominals” as ordinary atomic concepts with the TBox classification algorithm. This is inefficient due to redundant computations. For example, axiom $\{a\} \sqsubseteq \exists r.C$ has two contexts $\{a\}$ and C . Thus this axiom will be added into the *scheduled* queues of both $\{a\}$ and C . In context C , this axiom will be saved into the *processed* set of C and further retrieved as the left premise of Rule \mathbf{R}_H and/or \mathbf{R}_T , for some future right premise $\exists s.C \rightarrow E$. However our target language $\mathcal{ELH}_{\perp, \mathcal{R}_+}$ does not support nominals. Therefore it is unnecessary to maintain $\{a\} \sqsubseteq \exists r.C$ in context C because any corresponding right premise $\exists s.C \rightarrow E$ will not contain any nominal, hence it can always be computed independently from (or before) the derivation of $\{a\} \sqsubseteq \exists r.C$. This provides means to optimise reasoning because the concept hierarchies and RI closures will be static when doing ABox reasoning (cf. Sec 4.2).

Furthermore, it is important to optimise the seemingly trivial parts, which could become non-trivial due to the large number of individuals, of the algorithm in order to speed up the reasoning. Particularly:

1. Instead of initialising the contexts in a sequential manner one should parallelise this process in order to gain further efficiency (cf. Sec 4.3).
2. When applying completion rules to derive conclusions, as described by steps 2 at the end of the last section, a reasoner usually needs to check the forms of input axioms, decide the applicable rules, check the forms of conclusion axioms, etc. Some of the checking could be skipped, as they are all dependent thus can be streamlined (cf. Sec 4.4).
3. After derivation, the conclusions are maintained in a set, and then immediately retrieved to get contexts. All retrieved contexts are also maintained in a set, and then immediately retrieved for activation. One should be able to skip such save/retrieve steps and directly use the conclusions and their contexts given that the forms of conclusions are known to the reasoner (cf. Sec 4.4).

4. When an axiom is added into the *scheduled* queue of a context, the worker needs to activate this context for further processing. However if the context is the context currently under processing of the worker, such activation can be skipped. We only need to activate a context if we do not know it is the same as the current context (cf. Sec 4.4).

4 Approach

In this section we present parallel TBox and ABox classification algorithms for $\mathcal{ELH}_{\perp, \mathcal{R}+}$. We first present the new completion rules and then the algorithms.

4.1 TBox Completion Rules

We first extend the $\mathcal{ELH}_{\mathcal{R}+}$ TBox completion rules to support the bottom concept with the following rule for the \perp concept:

$$\mathbf{R}_{\perp} \frac{D \sqsubseteq \exists r.C, C \sqsubseteq \perp}{D \sqsubseteq \perp}$$

In what follows, we call the set containing the above rule and the $\mathcal{ELH}_{\mathcal{R}+}$ rules in Sec. 2.2 the \mathbf{R} rule set, which is sound and complete for $\mathcal{ELH}_{\perp, \mathcal{R}+}$ classification:

Lemma 1. *Let S be any set of TBox axioms closed under the \mathbf{R} rule set, then $C \sqsubseteq D$ iff $C \sqsubseteq \perp \in S$ or $C \sqsubseteq D \in S$ for any C and D such that $C \sqsubseteq C \in S$, D occurs in \mathcal{O} and $\perp \sqsubseteq \perp \in S$ if \perp occurs in \mathcal{O} .*

With the \mathbf{R} rules we can perform TBox reasoning:

Definition 1. (TBox Completion Closure) *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontology, its TBox completion closure, denoted by $S_{\mathcal{T}}$, is the smallest set of axioms closed under the rule set \mathbf{R} such that:*

1. for all $A \in \mathcal{CN}_{\mathcal{O}}$, $A \sqsubseteq A \in S_{\mathcal{T}}$;
2. $\perp \sqsubseteq \perp \in S_{\mathcal{T}}$ if \perp occurs in \mathcal{O} .

According to Lemma 1, we have $A \sqsubseteq C \in S_{\mathcal{T}}$ or $A \sqsubseteq \perp \in S_{\mathcal{T}}$ for any A and C where A is an atomic concept and C occurs in \mathcal{T} . This realises TBox classification.

4.2 ABox Completion Rules

Now we present the ABox completion rules for $\mathcal{ELH}_{\perp, \mathcal{R}+}$. Although $\mathcal{ELH}_{\perp, \mathcal{R}+}$ does not support nominals ($\{a\}$), we still denote individuals with nominals since this helps simplify the presentation: (i) ABox rules are more readable, as they have similar syntactic forms to the TBox ones, and (ii) some of the ABox rules can be unified. More precisely, we establish the following mappings as syntactic sugar:

$$\begin{aligned} C(a) &\Leftrightarrow \{a\} \sqsubseteq C \\ a \doteq b &\Leftrightarrow \{a\} \equiv \{b\} \\ a \neq b &\Leftrightarrow \{a\} \sqcap \{b\} \sqsubseteq \perp \\ r(a, b) &\Leftrightarrow \{a\} \sqsubseteq \exists r.\{b\} \end{aligned}$$

Obviously, these mappings are semantically equivalent. In the rest of the paper, without further explanation, we treat the LHS and RHS of each of the above mappings as a *syntactic* variation of one another. Together with the mapping, all ABox axioms in the original \mathcal{O} can be represented in a similar form of TBox axioms. Note that, axioms such as $C \sqsubseteq \{b\}$ and $C \sqsubseteq \exists r.\{b\}$ will not be in the ontology since they are invalid ABox axioms in $\mathcal{ELH}_{\perp, \mathcal{R}^+}$.

We present the ABox completion rules as follows — we call them the **AR** rules, which should be applied *after* a complete closure $S_{\mathcal{T}}$ is constructed from the **R** rules. In contrast to the **R** rules, the **AR** rules contain nested concepts $D_{(i)}$ and E that can take multiple forms including nominals. Thus the mapping between ABox and TBox axioms allows us to describe the rules in a more compact manner which would otherwise require additional rules to achieve the same purpose.

$$\begin{aligned}
\mathbf{AR}_{\sqsubseteq} & \frac{\{a\} \sqsubseteq D}{\{a\} \sqsubseteq E} : D \sqsubseteq E \in S_{\mathcal{T}} \cup \mathcal{A} \\
\mathbf{AR}_{\mathcal{H}}^* & \frac{\{a\} \sqsubseteq \exists r.D}{\{a\} \sqsubseteq E} : \exists s.D \rightarrow E \in S_{\mathcal{T}}, r \sqsubseteq_{\mathcal{O}}^* s \\
\mathbf{AR}_T^* & \frac{\{a\} \sqsubseteq \exists r.D}{\exists t.\{a\} \rightarrow E} : \exists s.D \rightarrow E \in S_{\mathcal{T}}, r \sqsubseteq_{\mathcal{O}}^* t \sqsubseteq_{\mathcal{O}}^* s, \text{Trans}(t) \in \mathcal{O} \\
\mathbf{AR}_{\sqcap}^- & \frac{\{a\} \sqsubseteq D_1 \sqcap D_2}{\{a\} \sqsubseteq D_1; \{a\} \sqsubseteq D_2} \\
\mathbf{AR}_{\sqcap}^+ & \frac{\{a\} \sqsubseteq D_1, \{a\} \sqsubseteq D_2}{\{a\} \sqsubseteq D_1 \sqcap D_2} : D_1 \sqcap D_2 \text{ occurs in } \mathcal{O} \\
\mathbf{AR}_{\exists}^+ & \frac{\{a\} \sqsubseteq D}{\exists s.\{a\} \rightarrow \exists s.D} : r \sqsubseteq_{\mathcal{O}}^* s, \exists r.D \text{ occurs in } \mathcal{O} \\
\mathbf{AR}_{\perp} & \frac{\{b\} \sqsubseteq \exists r.\{a\}, \{a\} \sqsubseteq \perp}{\{b\} \sqsubseteq \perp} \\
\mathbf{AR}_{\perp}^* & \frac{\{a\} \sqsubseteq \exists r.D}{\{a\} \sqsubseteq \perp} : D \sqsubseteq \perp \in S_{\mathcal{T}} \\
\mathbf{AR}_{\mathcal{H}} & \frac{\{b\} \sqsubseteq \exists r.\{a\}, \exists s.\{a\} \rightarrow E}{\{b\} \sqsubseteq E} : r \sqsubseteq_{\mathcal{O}}^* s \\
\mathbf{AR}_T & \frac{\{b\} \sqsubseteq \exists r.\{a\}, \exists s.\{a\} \rightarrow E}{\exists t.\{b\} \rightarrow E} : r \sqsubseteq_{\mathcal{O}}^* t \sqsubseteq_{\mathcal{O}}^* s, \text{Trans}(t) \in \mathcal{O} \\
\mathbf{AR}_{\mathcal{H}}^R & \frac{\{b\} \sqsubseteq \exists r.\{a\}}{\{b\} \sqsubseteq \exists s.\{c\}} : r \sqsubseteq_{\mathcal{O}}^* s, a = c \text{ or } \{a\} \sqsubseteq \{c\} \in \mathcal{A}
\end{aligned}$$

The **AR** rules deserve some explanations:

- There are clear correspondences between the **R** rules and **AR** rules. For example, $\mathbf{AR}_{\sqsubseteq}$ is an ABox counterpart of \mathbf{R}_{\sqsubseteq} except that the context is explicitly a nominal, and TBox results are used as side conditions. The last rule $\mathbf{AR}_{\mathcal{H}}^R$ is an additional rule to handle relations.

- Note that directly applying the **R** rules together with the **AR** rules could introduce unnecessary performance overheads such as axiom scheduling, processing and maintenance as we discussed in Sec. 3. In our approach, we separate TBox reasoning from ABox reasoning, and use TBox reasoning results as side conditions in ABox rules. This helps reduce memory usage and computation time.

Now we show below with an example on how the two-stage ABox reasoning works in operation. Suppose we have the following ontology:

$$PlanarStructure \sqsubseteq PhysicalStructure \quad (1)$$

$$PhysicalStructure \sqsubseteq GeneralisedStructure \sqcap \exists hasCountability.discrete \quad (2)$$

$$PlanarStructure \equiv \exists hasShape.(\exists hasAS.Laminar \sqcap Shape) \quad (3)$$

$$\{a\} \sqsubseteq \exists hasShape.\{b\} \quad (4)$$

$$\{b\} \sqsubseteq \exists hasAS.\{c\} \quad (5)$$

$$\{c\} \sqsubseteq Laminar \quad (6)$$

$$\{b\} \sqsubseteq Shape \quad (7)$$

We can see that (1)-(3) are TBox axioms and (4)-(7) are ABox axioms. Note that the input contains the assertions $hasShape(a, b)$, $hasAS(b, c)$, $Laminar(c)$ and $Shape(b)$, corresponding to (4)-(7) respectively. This conversion is expected to be performed before execution of the completion rules.

In the first stage, we compute the saturation of the TBox axioms (i.e., (1)-(3)) by applying the **R** rules. As an example, we illustrate how the axiom below is derived:

$$\exists hasShape.(\exists hasAS.Laminar \sqcap Shape) \sqsubseteq GeneralisedStructure \quad (8)$$

To begin, we apply \mathbf{R}_{\sqsubseteq} on (1) and (3) to get (9), then again on (2) and (9) to get (10):

$$\exists hasShape.(\exists hasAS.Laminar \sqcap Shape) \sqsubseteq PhysicalStructure \quad (9)$$

$$\begin{aligned} \exists hasShape.(\exists hasAS.Laminar \sqcap Shape) \sqsubseteq GeneralisedStructure \\ \sqcap \exists hasCountability.discrete \quad (10) \end{aligned}$$

Lastly, we apply the \mathbf{R}_{\sqcap}^- -rule to (10) to get (8). Similarly, we infer all other TBox axioms by applying the completion rules repeatedly. Once saturation of the TBox rules is completed and the closure $\mathbf{S}_{\mathcal{T}}$ is constructed, we use the output $\mathbf{S}_{\mathcal{T}}$ from the first stage as part of input to the second stage to compute the saturation of the ABox axioms. Below, we demonstrate how the ABox axiom $\{a\} \sqsubseteq GeneralisedStructure$ is inferred through the ABox rules. We start by applying \mathbf{AR}_{\exists}^+ on (6) and we get:

$$\exists hasAS.\{c\} \rightarrow \exists hasAS.Laminar \quad (11)$$

From (5) and (11) we apply the $\mathbf{AR}_{\mathcal{H}}$ -rule to infer:

$$\{b\} \sqsubseteq \exists hasAS.Laminar \quad (12)$$

We then apply \mathbf{AR}_{\sqcap}^+ on (12) and (7) to obtain the following:

$$\{b\} \sqsubseteq \exists hasAS.Laminar \sqcap Shape \quad (13)$$

Similarly, we apply $\mathbf{AR}_{\sqsupset}^+$ on (13), followed by $\mathbf{AR}_{\mathcal{H}}$ on (4)-(14):

$$\exists hasShape.\{b\} \rightarrow \exists hasShape.(\exists hasAS.Laminar \sqcap Shape) \quad (14)$$

$$\{a\} \sqsubseteq \exists hasShape.(\exists hasAS.Laminar \sqcap Shape) \quad (15)$$

Finally, we use the $\mathbf{AR}_{\sqsubseteq}$ -rule on (15) and (10) to get $\{a\} \sqsubseteq GeneralisedStructure$. It can be converted back into assertion form $GeneralisedStructure(a)$.

Definition 2. (Ontology Completion Closure) Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ ontology, its ontology completion closure, denoted by \mathbf{S} , is the smallest set of axioms closed under the \mathbf{AR} rule set such that:

1. $S_{\mathcal{T}} \subseteq \mathbf{S}$;
2. $\mathcal{A} \subseteq \mathbf{S}$ (axioms mapped as elaborated at the beginning of this section);
3. for all $a \in \mathcal{IN}_{\mathcal{O}}$, $\{a\} \sqsubseteq \{a\} \in X_{\mathcal{O}}$, and $\{a\} \sqsubseteq \top$ if \top occurs in \mathcal{O} ;

Similar to the \mathbf{R} rules, the above rules are also complete, sound and tractable. The soundness and tractability of rules are quite obvious. The completeness on ABox classification can be shown by the following Theorem:

Theorem 1. For any $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, we have either there is some $\{x\} \sqsubseteq \perp \in \mathbf{S}$, or

1. $\mathcal{O} \models D(a)$ only if $\{a\} \sqsubseteq D \in \mathbf{S}$ for D occurs in \mathcal{O} ;
2. $\mathcal{O} \models r(a, b)$ only if $\{a\} \sqsubseteq \exists r.\{b\} \in \mathbf{S}$ for $r \in \mathcal{RN}_{\mathcal{O}}$.

As we can see, the \mathbf{AR} rules also preserve the feature that all premises of each rule have a same common part as context. Therefore, they still enjoy the lock-free feature in reasoning. In later sections, we will further elaborate this point.

4.3 Parallel Algorithms

In this section, we present the parallel algorithms corresponding to the $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ completion rules. We reuse some notions such as *context*, *activeContexts* queue, *scheduled* queue and *processed* set from the original TBox algorithm for $\mathcal{ELH}_{\mathcal{R}^+}$ presented in [5] to realise the lock-free property. Most importantly, we need to make refinements to tailor the algorithm for $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ ABox reasoning. Here is a summary on how to deal with the challenges mentioned in Sec. 3:

1. In our algorithm, reasoning is separated into two stages: the first stage is $\text{saturate}(\text{TBoxInput})$, where contexts are $\mathcal{ELH}_{\perp, \mathcal{R}^+}$ concepts and the \mathbf{R} rules are applied. The second is $\text{saturate}(\text{ABoxInput})$, where contexts are the (mapped) nominals and the \mathbf{AR} rules are applied. See Algorithm 1 for details of the $\text{saturate}()$ method.

2. Different from the TBox saturate algorithm in [5], in our algorithm, we parallelise the initialisation (line 3-8 of Algorithm 1) to improve efficiency. As mentioned in Sec. 3, initialisation could become non-trivial due to the large number of individuals. The introduction of parallelisation could help speed up these parts.

The revised saturation algorithm (Algorithm 1) is presented as follows. The saturation of an ontology is realised by first performing saturation of the TBox, the output of which (i.e., $S_{\mathcal{T}}$) is then used in the saturation of the ABox. The saturation of the ABox yields S which satisfies Theorem 1. All necessary tautology axioms must be added to the input prior to saturation. For TBox, we add axioms of the form $C \sqsubseteq C$ into $S_{\mathcal{T}}$ for all concepts C such that $C \in \mathcal{CN}_{\mathcal{O}} \cup \{\perp\}$. Similarly, for ABox we add $\{a\} \sqsubseteq \{a\}$ into S for all individuals $a \in \mathcal{IN}_{\mathcal{O}}$.

Algorithm 1. saturate(input): saturation of axioms under inference rules

Input: input (the set of input axioms)
Result: the saturation of input is computed in context.processed

```

1 activeContexts  $\leftarrow$   $\emptyset$ ;
2 axiomQueue.addAll(input);
3 loop
4   axiom  $\leftarrow$  axiomQueue.pop();
5   if axiom = null then break;
6   for context  $\in$  getContexts(axiom) do
7     context.scheduled.add(axiom);
8     activeContexts.activate(context);
9 loop
10  context  $\leftarrow$  activeContexts.pop();
11  if context = null then break;
12  loop
13    axiom  $\leftarrow$  context.scheduled.pop();
14    if axiom = null then break;
15    process(axiom);
16  context.isActive  $\leftarrow$  false;
17  if context.scheduled  $\neq$   $\emptyset$  then activeContexts.activate(context);

```

In the saturation (Algorithm 1), the *activeContexts* queue is initialised with an empty set (line 1), and then all input axioms are added into an *axiomQueue* (line 2). After that, two main loops (lines 3-8 and lines 9-17) are sequentially parallelised. In the first main loop, multiple workers independently retrieve axioms from the *axiomQueue* (line 4), then get the contexts of the axioms (line 6), add the axioms into corresponding *scheduled* queues (line 7) and activate the contexts.

In the first loop of Algorithm 1 we need to get contexts of a given axiom (line 6), by calling the *getContexts()* method (Algorithm 2). As explained earlier, for TBox and ABox reasoning, the contexts are different. In ABox reasoning, only “nominals” can be contexts. Note that the *getContexts()* method is only used in Algorithm 1 during

Algorithm 2. `getContext(axiom)`

Input: an axiom
Result: the set of contexts that needs to be activated for the input axiom

```

1 result  $\leftarrow \emptyset$ ;
2 if axiom contains no nominal then // contexts for R rules
3   if axiom match  $C \sqsubseteq D$  then result.add( $C$ );
4   if axiom match  $D \sqsubseteq \exists r.C$  then result.add( $C$ );
5   if axiom match  $\exists s.C \rightarrow E$  then result.add( $C$ );
6 else // contexts for AR rules
7   if axiom match  $\{a\} \sqsubseteq C$  then result.add( $\{a\}$ );
8   if axiom match  $C \sqsubseteq \exists r.\{a\}$  then result.add( $\{a\}$ );
9 return result;
```

initialisation. The `process()` method (line 15 in Algorithm 1, to be discussed in Sec. 4.4), does not call the `getContexts()` method but directly get the contexts based on the form of input axiom. This is also different from the parallel TBox algorithm for $\mathcal{ELH}_{\mathcal{R}+}$ presented in [5].

The activation of a context (Algorithm 3) is the same as in the TBox algorithm for $\mathcal{ELH}_{\mathcal{R}+}$ [5]: an atomic boolean value *isActive* is associated with each context to indicate whether the context is already active. A context is added into the *activeContexts* queue only if this value is *false*, which will be changed to *true* at the time of activation. This procedure continues until the *axiomQueue* is empty.

Algorithm 3. `activeContexts.activate(context)`

Input: the context to be activated

```

1 if context.isActive.compareAndSwap(false, true) then
2   activeContexts.put(context);
```

In the second main loop of Algorithm 1, multiple workers independently retrieve contexts from the *activeContexts* queue (line 10) and process its *scheduled* axioms (line 15). Once *context.scheduled* is empty, *context.isActive* is set to *false* (line 16). A re-activation checking is performed (line 17) in case other workers have added new axioms into *context.scheduled* while the last axiom is being processed (between line 14 and line 16). This procedure will continue until the *activeContexts* queue is empty.

4.4 Cascading Processing

In this subsection, we describe the details of the `process()` method, which covers items 2.(a), 2.(b), 2.(c) at the end of Sec. 2.2. As mentioned in Sec. 3, it is important to optimise the seemingly trivial parts, which could become non-trivial due to the large number of individuals. To address many of the issues mentioned in Sec. 3, we present a cascading processing procedure (Algorithm 4).

Algorithm 4. process(axiom) for context $\{a\}$

Input: the axiom to be processed

```

1 if axiom match  $\{a\} \sqsubseteq D$  then
2   if  $D \in \{a\}.\text{subsumptions}$  then break;
3    $\{a\}.\text{subsumptions.add}(D)$ ;
   // For rule  $\mathbf{AR}_{\sqsubseteq}$ 
4   for  $E \in (D.\text{subsumptions} \cup D.\text{originalTypes})$  do
5     if  $E \notin \{a\}.\text{subsumptions}$  then
6        $\{a\}.\text{scheduled.add}(\{a\} \sqsubseteq E)$ ;
7       if  $E$  match  $\exists r.\{b\}$  then
8          $\{b\}.\text{scheduled.add}(\{a\} \sqsubseteq E)$ ;
9          $\text{activeContexts.activate}(\{b\})$ ;
   // similarly for rules  $\mathbf{AR}_H^*$ ,  $\mathbf{AR}_T^*$ ,  $\mathbf{AR}_{\sqcap}^-$ ,  $\mathbf{AR}_{\sqcap}^+$ ,  $\mathbf{AR}_{\sqcup}^+$ ,
    $\mathbf{AR}_{\perp}^*$  and  $\mathbf{AR}_{\perp}$  right premise
10 if axiom match  $\{b\} \sqsubseteq \exists r.\{a\}$  then
11   if  $\langle r, \{b\} \rangle \in \{a\}.\text{predecessors}$  then break;
12    $\{a\}.\text{predecessors.add}(\langle r, \{b\} \rangle)$ ;
13   if  $\perp \in \{a\}.\text{subsumptions} \setminus \{b\}.\text{subsumptions}$  then
14      $\{b\}.\text{scheduled.add}(\{b\} \sqsubseteq \perp)$ ;
15      $\text{activeContexts.activate}(\{b\})$ ;
   // similarly for rules  $\mathbf{AR}_H$ , left premise,  $\mathbf{AR}_T$ , left
   // premise and  $\mathbf{AR}_H^R$ 
16 if axiom match  $\exists s.\{a\} \rightarrow E$  then
17   if  $\langle s, E \rangle \in \{a\}.\text{implications}$  then break;
18    $\{a\}.\text{implications.add}(\langle s, E \rangle)$ ;
19   for  $r \in (\{a\}.\text{predecessors.keySet}() \cap s.\text{subRoles})$  do
20     for  $\{b\} \in \{a\}.\text{predecessors.get}(r)$  do
21       if  $E \notin \{b\}.\text{subsumptions}$  then
22          $\{b\}.\text{scheduled.add}(\{b\} \sqsubseteq E)$ ;
         // similar as line 7-9
   // similarly for rules  $\mathbf{AR}_T$ , right premise
23 return result;

```

We match the form of input axiom once (line 1) and check whether it has been processed before (line 2); if not it will be added into the *processed* set (line 3). Based on the form of axiom, applicable completion rules can be determined. Meanwhile, checking if the conclusion is already in corresponding context's processed set can be performed (line 5). Once a completion rule has been applied, the conclusion axioms and their forms are determined. Once a conclusion is derived, its contexts and whether they are definitely the same as the current context are determined. The conclusion axioms can directly be added into corresponding *scheduled* queues (line 6 and 8). For the

brevity of the paper, we only present the processing of some ABox axioms. Processing of TBox axioms and the other forms of ABox axioms can be done in a similar manner.

In Algorithm 4 certain axioms are maintained by several indexes to facilitate more efficient access. Most of them are the same as in the parallel TBox algorithm for \mathcal{ELH}_{R+} [5]. The additional one is $D.originalTypes$, which is used to maintain original ABox axioms:

$$\begin{aligned} D.originalTypes &= \{E | D \sqsubseteq E \in \mathcal{A}\}, \\ r.subRoles &= \{s | s \sqsubseteq_O^* r\}, \\ C.subsumptions &= \{D | C \sqsubseteq D \in \text{processed}\}, \\ C.predecessors &= \{\langle r, D \rangle | D \sqsubseteq \exists r.C \in \text{processed}\}, \\ C.implications &= \{\langle r, E \rangle | \exists r.C \rightarrow E \in \text{processed}\}, \end{aligned}$$

5 Evaluation

We implemented our algorithms in our PEL reasoner (written in JAVA). Inspired by the ELK reasoner [5], we also use thread-safe datatypes `ConcurrentLinkedQueue` for all the queues, including `activeContexts`, `axiomQueue` and `scheduled`. And we use `AtomicBoolean` for the `isActive` value of a context thus its `compareAndSwap` operation is atomic and thread-safe. The indexes we used in Algorithm 4 are implemented with normal `HashSet` and `HashMap`. We use OWL API to parse ontologies.

To compare our system against sequential reasoners we use the Amazon Elastic Computer Cloud (EC2) High-CPU Extra Large Instance⁷. It has 7 GB of memory and 8 cores with 2.5 EC2 compute units each, where each EC2 unit “provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor”. The OS is 64-bit Linux platform and running JVM version 1.6.0_20 with 7 GB memory. We run our implementation with sequential reasoners Pellet 2.2.2, FaCT++ 1.5.2, HermiT 1.3.2 and (the OWL 2 EL reasoner in) TrOWL 0.8 because they (except HermiT) implemented the EL algorithm and support ABox reasoning. Other reasoners, including dedicated EL reasoners such as the OWL API-compliant CEL, jCEL and Snorocket, etc. and consequence-driven reasoner CB and ELK, do not fully support ABox reasoning yet.

Our test cases include a slightly simplified real world ontology VICODI, and a real world TBox NotGalen with generated ABox⁸. The VICODI⁹ ontology is developed to represent the history of Europe. It has a simple TBox and moderate number of individuals. NotGalen⁻ is extracted from an earlier version of Galen¹⁰ by removing functional role assertions. It contains a moderate-size TBox and no ABox. To populate the ontology we use the SyGENiA system [14] to generate ABoxes for a small part of the Galen ontology and we combined the generated ABoxes of different sizes with the NotGalen⁻ TBox and aligned the namespaces to make reasoning more complicated; in this way, we have test ontologies NG-1, NG-2, NG-5 etc. Such ABoxes are not completely random because, as generated by SyGENiA, they cover axioms that can lead to

⁷ <http://aws.amazon.com/ec2/instance-types/>

⁸ Our test ontologies can be found at,

<http://www.box.net/shared/gok98u39s9lrmy3ie2n1>

⁹ <http://www.vicodi.org/about.htm>

¹⁰ <http://www.opengalen.org/>

all possible sources of incompleteness w.r.t. a TBox and certain query. Being able to handle such ABoxes means that the reasoner won’t miss any result when dealing with any real-world ABoxes. The stats of our ontologies are illustrated in the Table 2.

For each ontology, we perform ABox classification, i.e. to compute the atomic types of all the individuals and the atomic relations between all pairs of individuals. If such assertion are not “pre-computable” when a reasoner classifies the ontology, we use the reasoner API functions to retrieve these results to make sure they are computed. The time shown in our evaluation is the overall computation time. Results of sequential reasoners are presented in Table 2. Results of our implementation PEL are presented in Table 3. The timeout is one hour. Time unit is second.

Table 2. Ontologies and Results of Sequential Reasoners (in sec)

Ontology	$ \mathcal{CN} $	$ \mathcal{RN} $	$ \mathcal{IN} $	$ \mathcal{A} $	TrOWL	Pellet	Hermit	FaCT++
VICODI	184	10	29614	114164	2.014	9.971	13.138	timeout
NG-1	2748	413	4236	8008	4.284	210.945	307.93	timeout
NG-2			12161	23976	9.342	757.379	timeout	timeout
NG-5			47756	118458	28.947	timeout	timeout	timeout
NG-8			78899	278365	63.833	timeout	timeout	timeout
NG-13			97995	665304	143.288	timeout	timeout	timeout

Table 3. Results of PEL (in sec)

Ontology	1 worker	2 workers	4 workers	6 workers
VICODI	1.136	1.05	1.054	1.059
NG-1	2.339	1.361	1.169	1.069
NG-2	3.025	2.939	2.848	2.77
NG-5	6.427	6.004	5.114	5.125
NG-8	12.604	10.474	9.449	9.75
NG-13	23.609	20.853	16.877	17.539

From the comparison between Table 2 and 3 we can see that PEL is in general faster than sequential reasoners, especially when more workers are used. PEL is also good when dealing with combination of complex TBox and large ABox. For example, in the relatively simpler VICODI ontology, PEL is about 2 times faster than TrOWL, which is also highly optimised for \mathcal{EL} reasoning. While in the more complex NotGalen⁻ ontologies with a large ABox, PEL is up to 6 times faster than TrOWL with one worker, and up to about 8-9 times faster with multiple workers.

To further evaluate the scalability of PEL, we generated a different set of NotGalen⁻ ontologies with larger numbers of individuals, denoted by NGS-1, NGS-5, NGS-10, etc. And we use PEL to reason with these ontologies on a EC2 High-Memory Quadruple Extra Large Instance which has 8 virtual cores with 3.25 EC2 units each and 60 G memory allocated to JVM. Results are shown in Table 4.

From the comparison between different numbers of workers in Table 3 and Table 4 we can see that multiple parallel workers can indeed improve the reasoning performance, even when the ontology contains complex TBox and very large number of individuals. In general, the improvement is most profound from 1 worker to 2 workers, and

Table 4. Results of PEL (in sec) for Scalability Tests

Ontology	$ \mathcal{N} $	$ \mathcal{A} $	1 worker	2 workers	4 workers	6 workers
NGS-1	4031	8001	1.396	0.977	0.757	0.676
NGS-5	62572	119832	3.81	2.885	2.376	2.341
NGS-10	211408	437637	10.282	8.007	7.04	6.493
NGS-20	596007	1642876	52.753	40.208	40.228	34.507
NGS-30	866136	3542257	108.252	90.72	81.877	77.453
NGS-40	971222	6036910	172.743	146.411	131.536	129.827
NGS-50	995985	9025426	270.806	234.905	189.706	190.303

start to decrease when more workers are involved. With more than 4 workers, the performance may even decrease. We believe one of the potential reasons is that although the CPU cores can work in parallel, the RAM bandwidth is limited and RAM access is still sequential. In relatively “light-weight” ABox reasoning with large ABox, the RAM access will be enormous and very often so that multiple workers will have to compete for RAM access. This makes memory I/O a potential bottleneck of parallelisation and wastes CPU cycles. In our algorithm, especially the cascading processing, we have already tried to reduce unnecessary memory I/O. A better management of memory will be an important direction of our future work.

6 Conclusion

In this paper we extended early related work to present a parallel ABox reasoning approach to $\mathcal{ELH}_{\perp, \mathcal{R}+}$ ontologies. We have proposed new completion rules and show that they are complete and sound for ABox reasoning. We have revised the lock-free saturation procedure with optimisations that take the features of ABox reasoning into account. Particularly, we separate TBox and ABox reasoning to simplify derivation and parallelise many seemingly trivial steps to improve efficiency and reduce memory access. Our evaluation shows that ABox reasoning can benefit from parallelisation. Even with our naive implementation, we can outperform highly optimised \mathcal{EL} reasoners.

The evaluation results suggested that improving performance with more than 4 workers becomes difficult, which is also observed in [5]. In our future work we will further investigate its reason and pay special attention on the management of memory.

Acknowledgement. This work is partially funded by the K-Drive and ITA projects. We would also like to thank reviewers for their very constructive and helpful comments.

References

1. Aslani, M., Haarslev, V.: Parallel tbox classification in description logics –first experimental results. In: Proceeding of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, pp. 485–490. IOS Press, Amsterdam (2010)
2. Borgida, A., Serafini, L.: Distributed Description Logics: Assimilating Information from Peer Sources. In: Spaccapietra, S., March, S., Aberer, K. (eds.) Journal on Data Semantics I. LNCS, vol. 2800, pp. 153–184. Springer, Heidelberg (2003)

3. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. *Commun. ACM* 51, 107–113 (2008)
4. Hogan, A., Pan, J.Z., Polleres, A., Decker, S.: Saor: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 337–353. Springer, Heidelberg (2010)
5. Kazakov, Y., Krötzsch, M., Simančík, F.: Concurrent Classification of \mathcal{EL} Ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) *ISWC 2011, Part I. LNCS*, vol. 7031, pp. 305–320. Springer, Heidelberg (2011)
6. Liebig, T., Müller, F.: Parallelizing Tableaux-Based Description Logic Reasoning. In: Meersman, R., Tari, Z. (eds.) *OTM-WS 2007, Part II. LNCS*, vol. 4806, pp. 1135–1144. Springer, Heidelberg (2007)
7. Maier, R.M.F., Hitzler, P.: A mapreduce algorithm for el+. In: *Proc. of International Workshop of Description Logic (DL 2010)* (2010)
8. Meissner, A.: Experimental analysis of some computation rules in a simple parallel reasoning system for the \mathcal{ALC} description logic. *Applied Mathematics and Computer Science* 21(1), 83–95 (2011)
9. Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., van Harmelen, F.: Marvin: Distributed reasoning over large-scale semantic web data. *Web Semant.* 7, 305–316 (2009)
10. Schlicht, A., Stuckenschmidt, H.: Distributed resolution for alc. In: *Description Logics Workshop* (2008)
11. Schlicht, A., Stuckenschmidt, H.: Distributed Resolution for Expressive Ontology Networks. In: Polleres, A., Swift, T. (eds.) *RR 2009. LNCS*, vol. 5837, pp. 87–101. Springer, Heidelberg (2009)
12. 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)
13. Soma, R., Prasanna, V.K.: Parallel inferencing for owl knowledge bases. In: *Proceedings of the 2008 37th International Conference on Parallel Processing, ICPP 2008*, pp. 75–82. IEEE Computer Society, Washington, DC, USA (2008)
14. Stoilos, G., Grau, B.C., Horrocks, I.: How incomplete is your semantic web reasoner? In: *Proc. of AAI 2010*, pp. 1431–1436. AAI Publications (2010)
15. ter Horst, H.J.: Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *J. Web Sem.* 3(2-3), 79–115 (2005)
16. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.: Owl Reasoning with Webpie: Calculating the Closure of 100 Billion Triples. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) *ESWC 2010. LNCS*, vol. 6088, pp. 213–227. Springer, Heidelberg (2010)
17. 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)
18. Weaver, J., Hendler, J.A.: Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) *ISWC 2009. LNCS*, vol. 5823, pp. 682–697. Springer, Heidelberg (2009)
19. Wu, G., Qi, G., Du, J.: Finding all justifications of owl entailments using tms and mapreduce. In: *The ACM Conference on Information and Knowledge Management* (2011)

RP-Filter: A Path-Based Triple Filtering Method for Efficient SPARQL Query Processing

Kisung Kim¹, Bongki Moon², and Hyoung-Joo Kim¹

¹ Seoul National University, Seoul, Korea
kskim@idb.snu.ac.kr, hjk@snu.ac.kr

² University of Arizona, Tucson, U.S.A
bkmoon@cs.arizona.edu

Abstract. With the rapid increase of RDF data, the SPARQL query processing has received much attention. Currently, most RDF databases store RDF data in a relational table called triple table and carry out several join operations on the triple tables for SPARQL query processing. However, the execution plans with many joins might be inefficient due to a large amount of intermediate data being passed between join operations. In this paper, we propose a triple filtering method called RP-Filter to reduce the amount of intermediate data. RP-Filter exploits the path information in the query graphs and filters the triples which would not be included in final results in advance of joins. We also suggest an efficient relational operator RFLT which filters triples by means of RP-Filter. Experimental results on synthetic and real-life RDF data show that RP-Filter can reduce the intermediate results effectively and accelerate the SPARQL query processing.

Keywords: RDF store, SPARQL query processing, triple filtering, intermediate results.

1 Introduction

RDF(Resource Description Framework)[\[1\]](#) is the standard data model recommended by W3C for the sake of describing data in the semantic web. RDF data is a set of triples(subject, predicate, object) which describe the relationship between two resources(subject and object). The RDF data forms a graph called RDF graph which consists of the resources and their relationships. SPARQL[\[2\]](#) is the standard query language for RDF data and expresses the user's data needs as graph patterns. The SPARQL query processing can be viewed as the sub-graph pattern matching problem for the RDF graph[\[3\]](#). RDF features flexibility with little schema restriction and expressive power which can represent graph-structured data. By virtue of these features, RDF is widely used in many areas. For example, RDF has been used for the purpose of integrating heterogeneous databases or publishing data on the web in many areas, e.g. life science[\[4,5\]](#), open government[\[6\]](#), social networking[\[7\]](#) and multimedia[\[8\]](#).

With the fast growth of RDF data, there has been a lot of research on storing and querying of RDF data[\[9,10,11,12\]](#). Most state-of-the-art RDF engines employ

the relational model to store and manipulate RDF data. They store RDF data in a relation with three columns(S,P,O) which is called a *triple table* and evaluate SPARQL queries through a sequence of joins on the triple table. Let us consider the following SPARQL query.

```
SELECT ?n1 ?n2 ?n3 ?n4
WHERE {?n1 <p1> ?n2.
      ?n2 <p2> ?n3.
      ?n3 <p3> ?n4.}
```

The above SPARQL query consists of three triple patterns, which form a graph pattern. The evaluation of the SPARQL query is to find all subgraphs in the RDF graph matching with the query graph pattern. Fig. 1(a) shows a possible execution plan for the SPARQL query, which have three scan operators(one for each triple pattern) and two join operators. Each operator in the execution plans makes the partially matching fragments for the query graph pattern. For example, *Join₁* in Fig. 1(a) produces all the matching fragments for the graph pattern which consists of the second and the third triple pattern of the SPARQL query.

This form of the execution plan is widely adopted by many RDF engines but has a problem that it might generate many useless intermediate results. The useless intermediate results are the results which are generated by some operators but not included in the final results of the query. Assume that the numbers in Table 1 are the result cardinalities for the subgraph patterns included in the query graph pattern. *Join₁* in Fig. 1(a) generates all the matching fragments for the graph pattern in the third row of Table 1 and the number of the result rows is 500,000. However, the number of the final results(the first row in Table 1) is only 1,000. Consequently, at least 499,000 rows of 500,000 rows become the useless intermediate results. The cost which are consumed for generating and processing them is wasted because the useless intermediate results do not contribute to the query results. And in large-scale RDF dataset, the size of the intermediate results intends to increase and the overhead for the useless intermediare results becomes more serious.

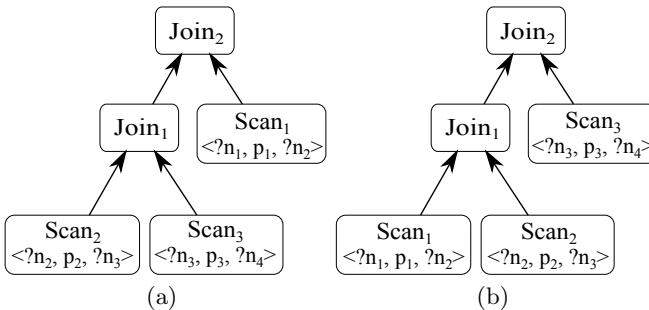
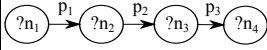
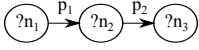
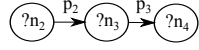


Fig. 1. Execution Plan

Table 1. Cardinalities of Intermediate Results

Graph Pattern	Cardinality
	1,000
	1,000,000
	500,000

Most RDF engines try to reduce these intermediate results by choosing an execution plan with the optimal join order when compiling the query. For example, Fig. 1(b) shows another execution plan whose results are the same with the Fig. 1(a) but whose join order is different from that of the execution plan in Fig. 1(a). The query optimizer prefers the execution plan in Fig. 1(a) to the execution plan in Fig. 1(b) because the latter would generate 500,000 more rows than the former plan. However, as we can see in this example, the execution plan with the optimal join order could not remove all the useless intermediate results.

In this paper, we propose a novel triple filtering method called *RP-Filter* (RDF Path Filter) to reduce the useless intermediate results effectively and efficiently using graph-structural information of RDF data. RP-Filter provides the list of the nodes in the RDF graph which are reached by paths with a specific path pattern. For example, we can obtain the list of nodes which can be reached by paths which are matching for a path pattern $\{(?n_1, p_1, ?n_2), (?n_2, p_2, ?n_3)\}$ from RP-Filter. This node list can be used as filter data to filter the result triples of *Scan₂* or *Scan₃* in Fig. 1(a) and then we can prune the triples which would not be joined in *Join₂* in advance. As a result, we can reduce the number of intermediate results using the path pattern information. Through these node lists, we can reduce the useless intermediate results effectively for complex SPARQL queries.

RP-Filter utilizes some properties of RDF engines, one of which is that they store the triples in a sorted order and the scanned triples are also sorted. Many RDF engines store their triples as sorted because it gives many optimization opportunities. The filtering process of RP-Filter is very efficient and incurs little overhead to the normal query processing because it utilizes this ordering property of scanned triples.

We propose the definition of RP-Filter and an relational operator called RPFLT which filters the input triples using RP-Filter. We also propose a method to generate an execution plan with the RPFLT operators using heuristic method. We carried out with several queries on large real-life and benchmark RDF datasets to evaluate RP-Filter. These results demonstrate that RP-Filter effectively and efficiently reduces the useless intermediate results and consequently accelerates the SPARQL query evaluation.

2 Related Work

Early RDF systems provide storage systems which use row-oriented RDBMSs as their back-end storages :e.g like Jena [10] and Sesame [9] (currently, Jena and Sesame provide also native storage systems which do not use RDBMSs). These RDBMS-based RDF systems store RDF triples in a triple table and utilize the query processing modules of RDBMSs when processing RDF queries. However, RDBMSs are not optimized to store and query the graph-structured RDF data and have several scalability problems.

SW-Store [11] uses a column-oriented store as its underlying store and propose the vertical partitioning method. SW-Store partitions the triple table by the predicate column and shows that the partitioning of the triple table have many advantages, like reduced I/O costs and the compact storage size. In addition, the triples are stored as sorted in a column-oriented store so that fast merge joins can be used when processing a query.

Currently the fastest RDF engine according to the published numbers is RDF-3X [12]. It stores triples in six clustered indexes redundantly as sorted by six different orderings(SPO,SOP,PSO,POS,OSP,OPS). RDF-3X can read matching triples for any type of triple patterns sorted by any ordering using the six indexes. Also RDF-3X uses a block compression techniques which store only deltas between triples rather than writing actual values. The compression technique reduces the size of storage and the number of disk I/O requests needed for reading triples. RDF-3X generates an execution plan which consists of mainly scan operators and join operators for a SPARQL query. Each scan operator in an execution plan scans one of the six indexes and the ordering of scanned triples depends on which index is used. There are two types of join operators: merge join and hash join. RDF-3X uses a merge join when the orderings of two input relations are the same with join variable. Otherwise RDF-3X uses a hash join.

In order to reduce the useless intermediate results, the authors of RDF-3X propose an RDF-specific *sideway information passing* technique called U-SIP [13]. It builds a sort of filters while processing an operator of an execution plan and passes the filters to other operators to avoid generating the useless intermediate results. U-SIP exploits the pipelined data flow of an execution plan to pass the filter information. However, it cannot transfer the filter information reversely to the data flow of the execution plan. As a result, the cases where U-SIP can be applied are limited by the pipeline-blocking operator like a hash join. Especially, the execution plan for long path patterns would use many hash joins and U-SIP could not be very effective in these cases.

3 Preliminary

In this section, we describe RDF data model and SPARQL query model. We do not cover the entire RDF and SPARQL specification. Rather we deal with a core fragment of RDF and SPARQL. We do not consider blank nodes, literals and data types in RDF. For SPARQL, we concentrate on the graph pattern

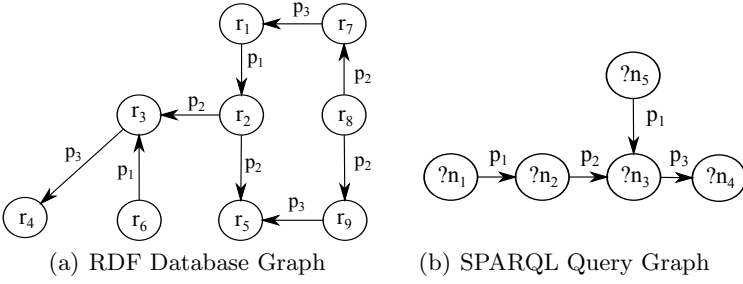


Fig. 2. RDF Database Graph and SPARQL Query Graph

matching of SPARQL, more specifically basic graph patterns [2] which consists of only conjunctive triple patterns. We also do not consider a join with predicate or a triple pattern having a variable predicate, as they are rarely used.

We assume the existence of two pairwise disjoint sets: a set U of URIs and a set VAR of variables. A variable symbol starts with $?$ to distinguish with URIs. A triple, $t \in U \times U \times U$ is called an RDF triple, and a triple, $tp \in (U \cup VAR) \times (U \cup VAR) \times (U \cup VAR)$ is called a triple pattern.

An RDF database D is a finite set of triples, and a SPARQL query Q is a finite set of triple patterns. We define three subsets of U such that $S = \{s | s \in U \wedge \exists t(s, p, o) \in D\}$, $P = \{p | p \in U \wedge \exists t(s, p, o) \in D\}$, $O = \{o | o \in U \wedge \exists t(s, p, o) \in D\}$.

We map RDF database D into a graph $G_D = (N_D, E_D, L_D)$ which consists of a node set N_D , a edge set E_D and a label set L_D , where $N_D = S \cup O$, $E_D = \{(s, p, o) | t(s, p, o) \in D\}$ and $L_D = P$. A SPARQL query Q is also mapped into a graph $G_Q = (N_Q, E_Q, L_Q)$, where $N_Q \subseteq S \cup O \cup VAR$, $E_Q = \{(s, p, o) | t(s, p, o) \in Q\}$ and $L_Q \subseteq P$. Both G_D and G_Q are edge labeled directed graphs. Fig. 2(a) shows an example RDF database graph and Fig. 2(b) shows an example SPARQL query graph.

A *path* on a graph $G = (N, E, L)$ is a sequence of connected edges in the graph. If the terminal node of a path is n , the path is an *incoming path* of n . For example, for the SPARQL query graph in Fig. 2(b), $pa = \langle (?n_1, p_1, ?n_2), (?n_2, p_2, ?n_3) \rangle$ is an incoming path of $?n_3$.

We define a *predicate path* as a sequence of predicates. The predicate path of a given path in $G = (N, E, L)$ is a sequence of edge labels of the path. For example, the predicate path of pa is $\langle p_1, p_2 \rangle$.

We use the notations $PPath(p)$ and $|PPath(p)|$ to denote the predicate path of a path p and its length, respectively. We also use $InPPath(n)$ to denote a set of all incoming predicate paths of $n \in N$. When the maximal path length l is given, a variant of the notation, $InPPath(n, l)$, is used to denote a subset of $InPPath(n)$ such that $InPPath(n, l) = \{ppath | ppath \in InPPath(n) \wedge |ppath| \leq l\}$.

Example 1 (Incoming Predicate Path). For the SPARQL query graph in Fig. 2(b), the incoming path set of $?n_4$ with the maximum length 2 is $InPPath(?n_4, 2) = \{\langle p_3 \rangle, \langle p_1, p_3 \rangle, \langle p_2, p_3 \rangle\}$.

4 RP-Filter

In this section, we present the definition of RP-Filter and the physical storage model of RP-Filter. To begin with, we discuss the requirement of RDF stores to use RP-filter.

4.1 Requirements of RP-Filter

In order to apply the RP-Filter technique into an RDF engine, the RDF engine should meet the following requirements.

1. URIs are mapped into integer IDs and the triples are stored using the IDs in a triple table with three column, S,P and O.
2. The execution plan has one scan operator for each triple pattern in the SPARQL query, which reads triples matching with the triple pattern from disks.
3. The scan operators read triples as sorted by the S or the O column.

Several RDF engines including RDF-3X utilize storage and query processing techniques which meet three conditions above for efficient query processing. Especially, the third condition is relatively strict. However, the condition is satisfied by several RDF store, e.g. RDF-3X and SW-store. This is because that the sorted materialization of triples provides a lot of efficiency, like the fast retrieval of matching triples and the usage of fast merge join. Therefore, RP-Filters can be adopted by various RDF engines including RDF-3X and SW-store.

4.2 Definition of RP-Filter

The RP-Filter for an RDF database is a set of node lists. A node list for a predicate path contains all the node IDs which have the predicate path as its incoming predicate path.

Definition 1 (Node List of $ppath$ N-List($ppath$)). *The node list for a predicate path $ppath$ is a sorted list of IDs of nodes n which satisfy that $ppath \in InPPath(n)$. We denote the node list of a predicate path $ppath$ as N-List($ppath$).*

By reading the node list for a predicate path, we can easily get all the node IDs which are reached by the path pattern which is described by the predicate path. Note that the node lists are sorted by the node IDs. The RP-Filter for an RDF database is defined as follows.

Definition 2 (RP-Filter of RDF database D with the maximum length $MaxL$). *Given an RDF database D and the maximum length $MaxL$, the RP-Filter of D is a set of all pairs $\langle ppath, N-List(ppath) \rangle$, for all $ppaths$ which exist in D and whose lengths are less than or equal to $MaxL$. $RP-Filter(D, MaxL) = \{ \langle ppath, N-List(ppath) \rangle | ppath \in \bigcup_{i=1}^{maxL} P^i \wedge ppath \text{ exists in } D \}$.*

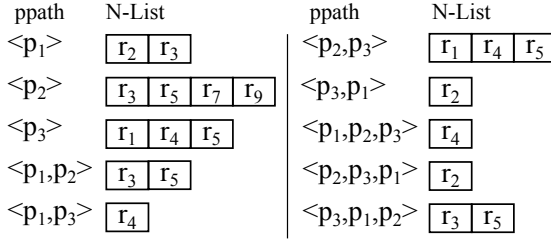


Fig. 3. RP-Filter (MaxL=3)

We say that a predicate path $ppath$ exists in D if and only if there exists a path whose predicate path is $ppath$ in D . We introduce $MaxL$ to limit the size of RP-Filters. As $MaxL$ increases, the number of predicate paths increases. As a result, the applicable RP-Filters also increase and the quality of RP-Filter improves but the size of RP-Filter also increases. In other words, there exists a tradeoff between the quality of RP-Filter and the space overhead of RP-Filter. We can control the tradeoff using the MaxL value.

Example 2 (RPFilter). Fig. 3 show RP-Filter($D, 3$) for the RDF database D in Fig. 2(a). The figure shows predicate paths and their node lists. There are 11 node lists in RP-Filter($D, 3$). And we can see that each node list is sorted by the node IDs.

4.3 Storage Model of RP-Filter

Each node list is stored in disk as sorted by node IDs so that the node list can be read in the sorted order while processing triple filtering. We use the delta based block compression technique used in RDF-3X [12] to alleviate the size overhead of RP-Filter and the disk I/O overhead for reading the node lists. In this method, the difference between two adjacent node IDs are stored. According to the size of the delta, the size of bytes for the writing of the delta is determined. We can use this compression method because the node IDs are sorted.

We organize the predicate paths in a trie(or prefix tree) called RP-Trie in order to search the node lists for a predicate path efficiently. RP-Trie is a trie built with all the predicate paths in RP-Filter. Each node in level l in RP-Trie has a pointer to the node list for its associated length- l predicate path. Fig. 4 shows RP-Trie for RP-Filter($D, 3$) in Fig. 3. We can find the location in disk of the node list for a predicate path by traversing RP-Trie using the predicate path. If there is no node for a predicate path whose length $\leq MaxL$, we can conclude that there exists no path which is matched to the predicate path in RDF database.

RP-Trie has the worst case space complexity, $O(\sum_{i=1}^{MaxL} |P|^i)$ and can grow exponentially to $MaxL$. But as we can see in section 6, for the real-life data sets and small $MaxL$ value, the size of RP-Trie is relatively of small size and the RP-Trie can be resident in the main memory.

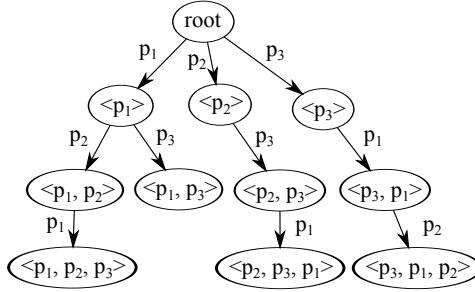


Fig. 4. RP-Trie for RP-Filter($D, 3$)

The structure of RP-Filter resembles the inverted index structure which is widely used in the information retrieval area. Each predicate path in RP-Filter can be considered a lexicon of an inverted index, while the node list is pretty much like a posting list. Just as a posting list of an inverted index keeps document IDs in sorted order, the node list keeps the node IDs in the sorted order. Note that the predicate paths are organized into RP-Trie to assist in locating the node list for a predicate path and finding relevant node IDs quickly.

5 Query Evaluation Using RP-Filter

In this section, we introduce a filtering operator RPFLT and then we discuss the query plan generation with the RPFLT operator.

5.1 RPFLT Operator

RP-Filter is used to filter the triples from scan operators in an execution plan. In order to use RP-Filter, the query compiler adds an operator called RPFLT to an execution plan. The RPFLT operator is a relational operator which gets triples from its child scan operator and outputs only the triples passing RP-Filter. An RPFLT operator is added to an execution plan as a parent operator of a scan operator.

Predicate Path Set of RPFLT. An RPFLT operator has a set of predicate paths called *PPS* (Predicate Path Set) assigned by the query compiler. To explain which predicate paths can be included in the *PPS* of an RPFLT, we define a property of a scan operator called *sortkey* as follows. The result triples of a scan operator are ordered by the *S* or the *O* column (not by the predicate column because we do not consider the predicate variable and the predicate join). We call the column by which the result triples are sorted a *sortkey column* of the scan operator. The *sortkey column* has a corresponding node in the mapped query graph. We also use the term *sortkey node* to indicate the *sortkey column*'s

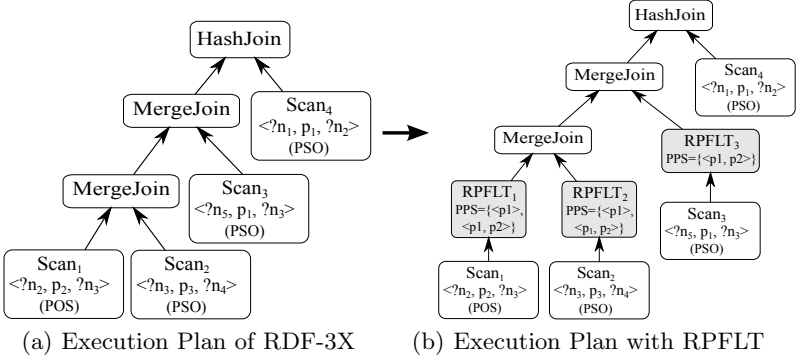


Fig. 5. Application of RPFLT Operators

corresponding node in the query graph. $Scan_i.sortkey$ is used to denote the sortkey column or the sortkey node of $Scan_i$ depending on context.

Fig. 5(a) shows an example execution plan of RDF-3X for the query graph in Fig. 2(b). The last item in each scan operator is the type of index to be used. For example, $Scan_1$ scans the POS index and so the scanned triples are ordered by (P,O,S). Since all the triples have the same predicate values ‘ p_2 ’, they are actually sorted by the O column. In the same way, the results of $Scan_2$ are ordered by the S column. Therefore, the sortkey column of $Scan_1$ is ‘O’ and the sortkey column of $Scan_2$ is ‘S’. And the sortkey nodes of $Scan_1$ and $Scan_2$ are both node $?n_3$ in Fig. 2(a), i.e., $Scan_1.sortkey = Scan_2.sortkey = ?n_3$.

The PPS of the RPFLT for $Scan_i$ should be a subset of $InPath(Scan_i.sortkey, MaxL)$. For example, $\langle p_1, p_2 \rangle$ can be included in the PPS of the RPFLT for $Scan_1$ because $Scan_1.sortkey = ?n_3$ and $\langle p_1, p_2 \rangle$ is in $InPath(?n_3, 3)$. Fig. 5(b) shows an execution plan which uses two RPFLT operators. The last item of an RPFLT operator lists the predicate paths in its PPS. The RPFLT operators for $Scan_1$ and $Scan_2$ have the same PPS because the sortkey for the two scan operators are same. The PPS is $\{\langle p_1, p_2 \rangle, \langle p_1 \rangle\}$ but $InPath(?n_3, 3)$ is $\{\langle p_1, p_2 \rangle, \langle p_1 \rangle, \langle p_2 \rangle\}$. The reason why only two of three predicate paths are included in the PPS is described in section 5.2. $Scan_3$ operator does not have an RPFLT operator because its sortkey node has no incoming predicate path.

Filtering Process of RPFLT. An RPFLT operator outputs the triples whose values of the sortkey column are include in all N-Lists for the predicate paths in its PPS. Fig. 6 illustrates the filtering process of $Scan_1$ in Fig. 5(b). The filtering process merges the input triples with all assigned N-Lists. In this example, $Scan_1$ outputs four triples but three of them are filtered out by $RPFLT_1$. $RPFLT_1$ outputs only one triple whose object is ‘ r_3 ’ because ‘ r_3 ’ is in both N-List($\langle p_1, p_2 \rangle$) and N-List($\langle p_1 \rangle$).

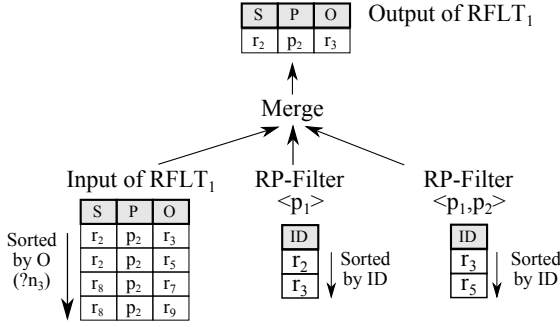


Fig. 6. Filtering in RPFLT Operator

The $?n_3$ node ($Scan_1.sortkey$) in the query graph has the two predicate paths in its InPPath. So the matching data nodes for $?n_3$ must have the two incoming predicate paths, too. The intersection of the two N-Lists gives us the matching data nodes which have both of the two incoming predicate paths. If the three filtered triples were not filtered out, they would be carried over to the next join operations - a *MergeJoin* and a *HashJoin* - and slow down the overall query processing without contributing to the final query result. In this manner, we attempt to filter triples out at the earliest possible stage, if they would not be included in the final results. Note that the filtering process involves reading the node lists and merging them with the input triples. We can filter the input triples simply by merging the node lists and the input triples because they share the same orderings. Also the N-Lists are usually of small length. Consequently, the reading and merging of the N-Lists incur little overhead and the RPFLT operator is very efficient and light operator.

Analysis of RPFLT. If we consider an node list as a table with the single column ID, the output of RPFLT can be described formally as following.

$$RPFLT(Scan_i, PPS) = \left(\bigcap_{ppath \in PPS} \text{N-List}(ppath) \right) \bowtie_{ID=Scan_i.sortkey} Scan_i \tag{1}$$

We use $Scan_i$ to denote the result relation of $Scan_i$, which have three columns and the $Scan_i.sortkey$ to denote the sortkey column of $Scan_i$. Note that we use the relational algebra only to describe the output of RPFLT not to describe the evaluation order of RPFLT.

The cost of $RPFLT(Scan_i, PPS)$ is $B \times \sum_{ppath \in PPS} \|\text{N-List}(ppath)\| + C \times \left(\sum_{ppath \in PPS} |\text{N-List}(ppath)| + |Scan_i| \right)$, where B is the cost of disk block I/O, C is the cpu cost for the merging, $|\text{N-List}(ppath)|$ and $\|\text{N-List}(ppath)\|$ are the number of nodes in $\text{N-List}(ppath)$ and the number of blocks of $\text{N-List}(ppath)$, respectively.

5.2 Generating an Execution Plan with RPFLT Operators

We use the 2-phase query optimization method to make execution plans with RPFLT operators. In the first phase, the query compiler generates an optimized execution plan through its normal query optimization process. Then, in the second phase, the query compiler adds RPFLT operators to the optimized execution plans. This 2-phase method uses heuristics that the optimized plan with no RPFLT also tends to be optimal when augmented by the RPFLT operators.

In the second phase, the query compiler examine the incoming predicate paths of the sortkey node of each scan operator in the execution plan. The query compiler makes decisions about which predicate paths are included in the PPS for the RPFLT operator. If the PPS is empty, no RPFLT operator is added to the scan operator.

When deciding the PPS, the query compiler should be careful not to choose redundant predicate paths. If a predicate path $ppath_1$ is a suffix of another predicate path $ppath_2$, $N\text{-List}(ppath_1) \supset N\text{-List}(ppath_2)$. Therefore, it is of no use to include both $ppath_1$ and $ppath_2$ in PPS. The query compiler should include only $ppath_2$ in PPS because $N\text{-List}(ppath_2)$ has less node IDs than $N\text{-List}(ppath_1)$ and is more effective filter. For example, let us take a look at Fig. 5 again. There exist three predicate paths in $\text{InPPath}(?n_3, 3) = \{\langle p_1, p_2 \rangle, \langle p_1 \rangle, \langle p_2 \rangle\}$. However, we do not include $\langle p_2 \rangle$ in the PPS of $Scan_1$ (or $Scan_2$), because $\langle p_2 \rangle$ is a suffix of $\langle p_1, p_2 \rangle$.

For another case of redundant predicate paths, if the triple pattern of $Scan_i$ is $\langle ?s, p_n, ?o \rangle$, we need not to include $\langle p_n \rangle$ in the PPS even though $\langle p_n \rangle$ is in InPPath . The reason is because $N\text{-List}(\langle p_n \rangle)$ could not filter any triple from $Scan_i$.

Note that the execution plan generated by this 2-phase method might not optimal. It is because RPFLT operators can change the cardinalities of the intermediate results. The join order of the original execution plan might be not optimal for the changed cardinalities of the intermediate results. To solve this problem, we should be able to decide the optimal join order for the execution plan with RPFLT. However, this requires a method to estimate the cardinalities of the filtered triples. Also, the additional costs of RPFLT operators could make the execution plan more expensive than the original execution plan. However, we leave this issue as future work and here we use this heuristic method.

6 Experimental Results

We implemented RP-Filter on the open-source system RDF-3X(0.3.5 version). The system was implemented with C++ and compiled by g++ with -O3 flag. We conducted all the experiments on an IBM machine having 8 3.0GHz Intel Xeon cores, 16GB memory and running 64bit 2.6.31-23 Linux Kernel. We used two datasets: YAGO2 [14] as a small real-life dataset and LUBM [15] as a large synthetic dataset. We generated the LUBM dataset with 10,000 universities [1].

¹ <http://swat.cse.lehigh.edu/projects/lubm/>

Table 2. Statistics about Datasets

	predicate	triples	RDF-3X size
YAGO2	94	195,048,649	7.6 GB
LUBM	18	1,334,681,192	70 GB

Table 2 shows statistics about the datasets. Note that YAGO2 has 94 predicates but LUBM has only 18 predicates. That is because YAGO2 is a combination of heterogeneous datasets(Wordnet and Wikipedia), while LUBM is about relatively homogeneous domain(university).

6.1 RP-Filter Size

For two datasets we built RP-Filters with MaxL=3. Table 3 shows the number of N-Lists and the total size of RP-Filters. As we can see, the size of RP-Filters for two datasets are much smaller than the input dataset sizes. The number of N-Lists for YAGO2 is higher than that of LUBM, although the data size of YAGO2 is smaller than the size of LUBM. This is because YAGO2 has more predicates than LUBM, there exists more distinct predicate paths in YAGO2.

6.2 Query Execution Time

To evaluate the performance of RP-Filter, we compared the query execution time of RDF-3X using RP-Filter with the original RDF-3X system. We measured the executions in the wall-clock time. RDF-3X converts node IDs in the final results into URIs to display the query results in the final stage of the query evaluation. The converting process is very time-consuming when there are large number of final results. Because the converting process is not relevant to the performance evaluation of RP-Filter, we excluded it from the execution time. We also counted the number of intermediate results for each evaluation. The number of intermediate results is the summation of the number of results of all the operators in the execution plan except the final operator. We exclude it because the number of final results is not changed by the filtering.

YAGO2 Dataset. For YAGO2 dataset, we generated several random path queries. We chose 15 predicates and made 3~7-length path queries using the chosen predicates. Each path queries has a single path and similar to the SPARQL query in Section 4.

Table 3. RP-Filter Size(MaxL=3)

	N-Lists	total size	avg. length
YAGO2	39,635	836MB	16,305
LUBM	122	1.3GB	2,571,504

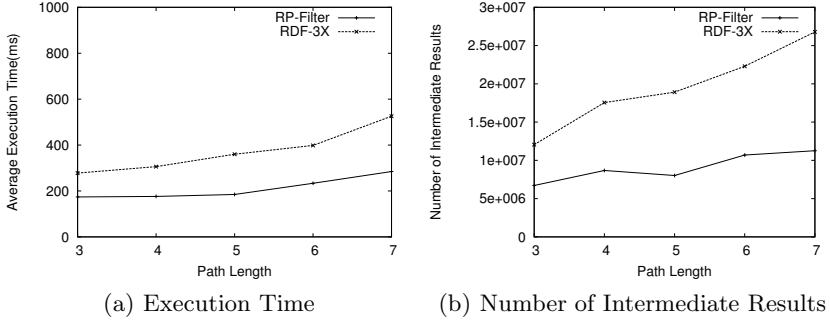


Fig. 7. Evaluation Results:YAGO

Fig. 7 shows the average execution time and the average number of the intermediate results. The average execution time of RP-Filter is lower than that of RDF-3X. Also the number of the total intermediate results is lower than that of RDF-3X. The growth rate of the execution time and the number of the intermediate results for RP-Filter is slower than those of RDF-3X.

LUBM Dataset. For LUBM datasets, we used two queries from 14 queries of the LUBM benchmark. We chose two of them because other queries have very simple structure and short paths in them. The queries we chose are Q2 and Q9. They have a 3-length path and complex structures. Q2 and Q9 are listed in Appendix. In order to execute the queries, we had to change the query Q9 slightly. In fact, LUBM is a benchmark for the RDF engines with the reasoning capability. However, currently RDF-3X does not support RDF reasoning. RDF-3X gives no answer for the query Q9 because the query asks about the instances of an inferred class. Therefore, we changed the class name so that no inference is needed, while leaving the structure of the queries unchanged.

Table 4 shows the execution time and the number of intermediate results for each query. For cold cache, the file system caches were dropped by `/bin/sync` and `echo 3 > /proc/sys/vm/drop_caches` commands. We evaluated the queries with U-SIP technique in RDF-3X. The results show that U-SIP is not very effective for the queries we used. The reason is that the execution plans involve many hash joins so the effect of U-SIP is limited. For RP-Filter, Q2 and Q9 showed different results. Q2 was improved by a factor of about 3 but for Q9 RP-Filter had little effect. That is because in Q2 there exists very selective path pattern but there is no such path pattern in Q9. And For Q2 the intermediate results are significantly reduced but for Q9, the intermediate results are not reduced much. For Q9, we can also observe that the query time is slightly longer when using RP-Filter. That is because the overhead for RP-Filter is more strong than the benefits of reduced intermediate results.

Table 4. Evaluation Results:LUMB (times in second)

	Warm cache		Cold cache		Intermediate Results	
	q2	q9	q2	q9	q2	q9
RDF-3X	26.5	37.0	28.7	43.4	424,747,108	659,968,158
NO U-SIP	22.9	31.9	25.0	38.4	424,785,330	662,615,314
NO U-SIP RP-Filter	7.4	32.7	9.6	40.2	308,143,082	620,276,418
RP-Filter	8.1	36.9	9.3	43.7	233,654,645	617,592,582

7 Conclusions and Future Work

In this paper, we propose a triple filtering method called RP-Filter. Based on the information about incoming paths of the query graph, RP-Filter prunes the scanned triples which would not be included in the final results. This triple filtering helps to reduce the useless intermediate results and reducing the intermediate results improves the query execution performance. Our experimental results shows that RP-Filter is very effective to reduce the useless intermediate results. For future work, we plan to explore how to reduce the overhead of RP-Filters and how to generate plans using RP-Filters based on the cost model.

Acknowledgements. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 20110017480) and the Brain Korea 21 Project.

Appendix

LUBM Queries PREFIX rdf:<<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>
PREFIX lubm:<<http://http://www.lehigh.edu#>>

Q2

```
SELECT ?x ?y ?z WHERE {
  ?x rdf:type lubm:GraduateStudent.
  ?y rdf:type lubm:University.
  ?z rdf:type lubm:Department.
  ?x lubm:memberOf ?z.
  ?z lubm:subOrganizationOf ?y.
  ?x lubm:undergraduateDegreeFrom ?y.}
```

Q9

```
SELECT ?x ?y ?z WHERE {
  ?x rdf:type lubm:Student.
  ?y rdf:type lubm:Faculty.
  ?z rdf:type lubm:Course.
  ?x lubm:advisor ?y.
  ?x lubm:takesCourse ?z.
  ?y lubm:teacherOf ?z.}
```

References

1. Klyne, G., Carroll, J.J.: Resource description framework (rdf): Concepts and abstract syntax. Technical report, W3C Recommendation (2004)
2. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Technical report, W3C Recommendation (2008)
3. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. *ACM Trans. Database Syst.* 34(3) (2009)
4. Belleau, F., Nolin, M.-A., 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. Redaschi, N., Consortium, U.: UniProt in RDF: Tackling Data Integration and Distributed Annotation with the Semantic Web. In: *Nature Precedings* (2009)
6. Sheridan, J.: Linking UK government data. In: *WWW Workshop on Linked Data*, pp. 1–4 (2010)
7. Mika, P.: Social Networks and the Semantic Web. In: *Proceedings of International Conference on Web Intelligence (WI 2004)*, pp. 285–291 (2004)
8. 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)
9. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) *ISWC 2002. LNCS*, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
10. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: *Proceedings of the 13th International World Wide Web Conference (WWW 2004)*, pp. 74–83 (2004)
11. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: SW-Store: a vertically partitioned DBMS for Semantic Web data management. *The VLDB Journal* 18(2), 385–406 (2009)
12. Neumann, T., Weikum, G.: Rdf-3x: a risc-style engine for rdf. *PVLDB* 1(1), 647–659 (2008)
13. Neumann, T., Weikum, G.: Scalable join processing on very large rdf graphs. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2009)*, pp. 627–640 (2009)
14. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: Yago2: Exploring and querying world knowledge in time, space, context, and many languages. In: *Proceedings of the 20th International Conference on World Wide Web (WWW 2011)*, pp. 229–232 (2011)
15. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* 3(2-3), 158–182 (2005)

Constructing Virtual Documents for Ontology Matching Using MapReduce

Hang Zhang, Wei Hu, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, China
hangzhang@smail.nju.edu.cn, {whu,yzqu}@nju.edu.cn

Abstract. Ontology matching is a crucial task for data integration and management on the Semantic Web. The ontology matching techniques today can solve many problems from heterogeneity of ontologies to some extent. However, for matching large ontologies, most ontology matchers take too long run time and have strong requirements on running environment. Based on the MapReduce framework and the virtual document technique, in this paper, we propose a 3-stage MapReduce-based approach called V-Doc+ for matching large ontologies, which significantly reduces the run time while keeping good precision and recall. Firstly, we establish four MapReduce processes to construct virtual document for each entity (class, property or instance), which consist of a simple process for the descriptions of entities, an iterative process for the descriptions of blank nodes and two processes for exchanging the descriptions with neighbors. Then, we use a word-weight-based partition method to calculate similarities between entities in the corresponding reducers. We report our results from two experiments on an OAEI dataset and a dataset from the biology domain. Its performance is assessed by comparing with existing ontology matchers. Additionally, we show how run time is reduced with increasing the size of cluster.

1 Introduction

The Semantic Web is an ongoing effort by the W3C community. To push traditional knowledge towards a common expression form, a number of data producers, such as MusicBrainz [12] and FMA [18], have published their data in the form of ontologies.

The wildly use of ontologies brings a practical problem. Due to the dispersion of Web data, there are multiple ontologies from different publishers over the world. Therefore, in the same or related domain, different ontologies may contain heterogeneous classes, properties and instances (all of them are uniformly called entities in this paper), which need ontology matching techniques to find those denoting the same thing in the real world [6,11].

To date, a number of ontology matching tools have been created to solve the problem of heterogeneity. Referring to the report of OAEI 2010 [4], some ontology matchers have good performance on real world datasets. However, a complex matching algorithm often leads to a long run time. According to our

investigation, most ontology matching tools suffer from unsatisfiable run time in large ontology matching despite of their high precision and recall. For example, in medicine and biology domains, two large ontologies (FMA [18,23] and GALEN¹) need to be matched. But most matchers spend hours even weeks on matching them. The main reason is their complex matching algorithms with limited CPU and memory environments [17]. Some researchers focused on the solution, such as ontology partition [8] and early pruning of dissimilar element pairs [15]. But they all fail to utilize the great power of modern parallel computing devices.

In this paper, we propose a parallel matching approach called V-Doc+ which is based on virtual document [16] and MapReduce [2]. The architecture of our approach is outlined in Fig. 1.

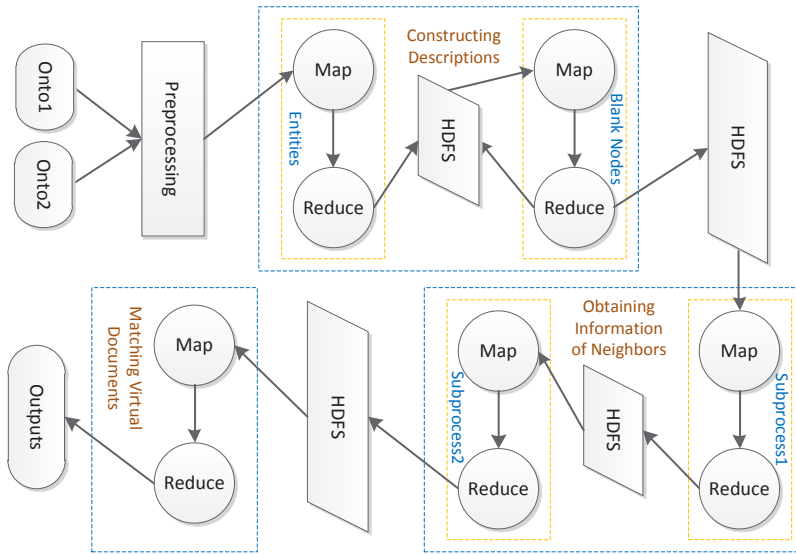


Fig. 1. Overview of the approach

Our approach contains three end-to-end MapReduce stages: constructing descriptions, obtaining information of neighbors and matching virtual documents. Before all start, in the stage of preprocessing, ontologies are splitted into the files which fit the requirement of the input format of MapReduce [2]. Based on the separated entities and RDF statements, the descriptions of entities and blank nodes will be calculated using several iterative MapReduce processes. Also, we optimize the description of each entity and blank node by annotating them with the descriptions of the neighbors. Finally, in the stage of matching virtual documents, we extract the high-weight words in the descriptions and partition the entities for reducing calculation space.

¹ <http://www.opengalen.org/>

We test V-Doc+ on the Food Ontology from OAEI 2007 and FMA vs. GALEN in medicine and biology domains. The experimental result shows a good run time of our approach with moderate precision and recall. The comparison of the efficiency in the environment with different number of nodes is also specified in the paper.

The rest of this paper is organized as follows. Sect. 2 presents the foundation of our approach and introduces the problem and Sect. 3 discusses related works. In Sect. 4 we give a MapReduce-based approach to construct the descriptions of entities and blank nodes. Information of neighbors is utilized in Sect. 5. Based on the virtual documents, similarities are calculated and the method is introduced in Sect. 6. Experimental results on two datasets are shown in Sect. 7. Finally, Sect. 8 concludes this paper with future work.

2 Preliminaries

In this work, all stages are established on the MapReduce framework. In the remainder of this section, we give the problem statement and introduce MapReduce briefly.

2.1 Problem Statement

There are a number of works that present different viewpoints on the ontology matching problem. In this paper, we define ontology matching as the process of finding mappings between entities from different ontologies. Each mapping consists of two entities and their confidence value. The definition is given as follows:

Definition 1 (Ontology Matching). *Let \mathcal{O} and \mathcal{O}' be two ontologies. The objective of ontology matching is to find a set of mappings defined as follows:*

$$\mathcal{M} = \{\mathcal{O}, \mathcal{O}', \mathcal{M}\} \quad (1)$$

where $\mathcal{M} = \{m_1, m_2, \dots, m_n\}$ denotes a set of mappings. A mapping m_i can be written as $m_i = (e, e', sim)$ where e and e' are two entities from \mathcal{O} and \mathcal{O}' respectively, and $sim \in (0, 1]$ denotes the similarity between e and e' .

2.2 MapReduce

MapReduce is the most popular framework for parallel computation on a number of computing nodes [2]. In MapReduce, the unit of computing process is named job, where each job consists of two main phases: map and reduce. The map inputs the data from the sources and splits each record into key/value pairs. These pairs are partitioned into different reducers according to the keys. Before a reducer handles the data, partitioned pairs are sorted by their keys and all values sharing the same key are clustered into the same set. The computation process is expressed in Fig. 2.

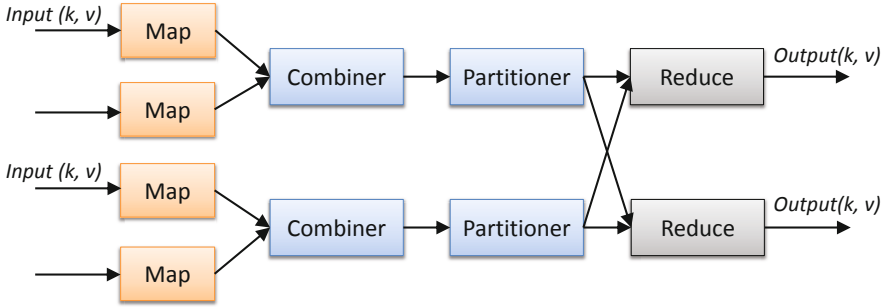


Fig. 2. Data flow in MapReduce

MapReduce also provides programmers with an extensible framework. Based on the interfaces, programmers are allowed to assign rules on how to partition key/value pairs and how to sort by keys, and these partitioning rules offers benefits for balancing the workload. Additionally, a combiner can also be rewritten to perform a local reducer to relieve the workload of the reducers.

MapReduce has the great power of parallel computing which makes used widely in a number of fields. In the ontology matching problem, one process which costs much time is to match every two entities. For some similarity-based matchers, calculating similarity on a large-scale dataset will cost too much time. As an example, calculating similarity for every pair of a n -size set of entities must be repeated $\frac{n*(n-1)}{2}$ times. Fortunately, MapReduce provides a parallel approach to partition data. According to the customized rules of partitioning, the n -size set can be partitioned into several subsets and different subset is calculated in different computing node. Thus, the run time is largely reduced.

Another usage of MapReduce for ontology matching is to solve the set-join problem. For example, for class c , finding all *rdfs:comment* values of c need to find all RDF statements that satisfy $(c, rdfs:comment, l)$ where l is a literal. Using MapReduce process can join c with its related RDF statements in a specific reducer.

3 Related Work

Although plenty of works have been proposed for the ontology matching problem, few approaches focus on matching large ontologies. In fact, some simple matching algorithms can deal with large ontologies with a good run time, such as edit distance [19]. However, the good efficiency of them mostly relies on the lightweight algorithms, which sometimes cannot achieve high precision and recall.

Rahm [17] summarized works towards large-scale schema and ontology matching. One solution is to reduce search space. Rewriting matching processes [15] could deal with different types of matching processes and use filter operators to

prune dissimilar element pairs. The work in [8] integrated a structure-based partitioning algorithm into Falcon-AO. This divide-and-conquer algorithm could calculate anchors and partition a large ontology into small clusters. Different from Falcon-AO, the work in [22] used the positive and negative reduction anchors but did not partition ontologies. However, these works still do not improved efficiency enough for a single compute node. Simplifying original algorithms to solve the ontology matching problem is also an option [11], but may not obtain a good recall.

There exists an approach that computes set-similarity joins with MapReduce [20]. This approach proposed both self-join and R-S join cases, and partitioned data in order to reduce matching space and balance the workload. The experiments showed a surprising run time and a good speedup on large-scale datasets. But the approach cannot be directly applied to matching ontologies as there are blank nodes existing.

For matching large ontologies, some researchers investigated and compared two kinds of parallelization on matching ontologies [7], and intra-matcher parallelization has been proved more versatile. The experiment also showed a feasibility of parallel matching. However, this kind of approaches is in essence a parallelizing matching workflow service consisting of a job queue, which is not easy to implement.

4 Constructing Descriptions

The construction of descriptions which are described by a collection of words with weights consists of two MapReduce sub-stages. We first preprocess to transform ontologies into records which can be sent to mappers directly. To minimize the network traffic, in all MapReduce processes, the real URIs are not transferred. So each URI in the RDF statements is replaced with an identifier based on unique name assumption. For example, we identify a class using a token c_i , a property using a token p_i and an RDF statement using s_i while w_i denotes a external URI involved.

In this section, we focus on how to construct descriptions in the MapReduce framework.

4.1 Descriptions of Entities

The first sub-stage is for named entities whose descriptions can be obtained by the local information. For an entity e , the description is defined as follows:

$$\begin{aligned}
 Desc(e) = & \alpha_1 * \text{collection of words in the local name of } e \\
 & + \alpha_2 * \text{collection of words in the rdfs:label of } e \\
 & + \alpha_3 * \text{collection of words in the rdfs:comment of } e \\
 & + \alpha_4 * \text{collection of words in other annotations of } e \quad (2)
 \end{aligned}$$

where $\alpha_1, \alpha_2, \alpha_3$ and α_4 are fixed rational numbers in $[0,1]$.

Fig. 3 presents an example data flow of constructing descriptions for entities. In the process, we ignore blank nodes. All the information of *rdfs:comment* and other annotations come from RDF statements. So the records from input involve classes, properties and RDF statements.

The map function extracts the identifiers to check if they are RDF statements. For each record of statement, the map function replaces its identifier with its subject. But for classes and properties, the keys are directly emitted. For example, in the figure, a record of RDF statement $(s1, (c2, p1, l1))$ is emitted by changing the key $s1$ to $c2$. However, a record of class $(c1, class1)$ is emitted without any changes.

After aggregating by keys, for each class c or property p , all related RDF statements are partitioned to the same reducer. The reduce function then calculates $Desc(c)$ or $Desc(p)$ according to Equation (2).

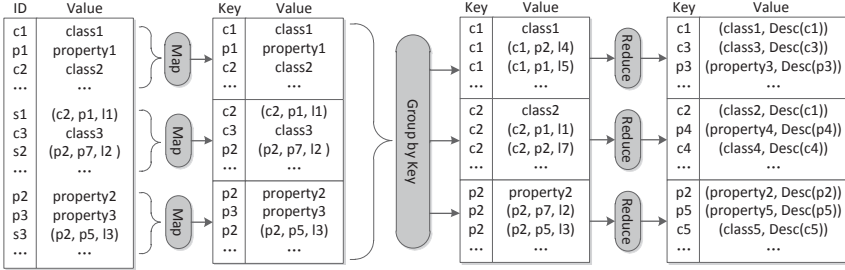


Fig. 3. Example data flow of constructing descriptions for entities

4.2 Descriptions of Blank Nodes

The second sub-stage is constructing descriptions for blank nodes. Having no local description, blank nodes get their information from neighbors, which may involve an iterative process. The following iteration equations give a convergence solution:

$$Desc_1(b) = \sum_{subj(s)=b} Desc(pred(s)) + \sum_{\substack{subj(s)=b \\ obj(s) \notin B}} Desc(obj(s)) \quad (3)$$

$$Desc_{k+1}(b) = \beta * (Desc_1(b) + \sum_{\substack{subj(s)=b \\ obj(s) \in B}} Desc_k(obj(s))) \quad (4)$$

where $subj(s)$, $pred(s)$ and $obj(s)$ denote subject, predicate and object of an RDF statement respectively. β is an attenuation coefficient in the $[0,1)$ range.

However, MapReduce is not designed to handle the recursive problem. So we transform each step of above equations into a MapReduce process. Firstly, we establish a blank node structure to implement WritableComparable interface, which extends the simple transmission unit for carrying the information of remaining nodes. Thus, the record of blank node b is extended from $Desc(b)$ to $(Desc(b), \{neb_1, neb_2, \dots, neb_n\})$, where neb_i denotes the remaining node waiting to be calculated.

We build a k -times-repeated MapReduce process to calculate descriptions for blank nodes. Before map function starts, every remaining nodes set is initialized with the current blank node. The input of map function derives from the output of the first sub-stage. Fig. 4 shows an example data flow of constructing descriptions for blank nodes. For each neb_i in $(Desc(b), \{neb_1, neb_2, \dots, neb_n\})$, a map function generates a new key-value pair where the key is neb_i and the value is $(Desc(b), \{\})$. Thus, a record of blank node may be replicated as many times as the number of remaining nodes. The treatment of RDF statements is similar with that in the first sub-stage.

The reduce function aggregates a blank node b with the related entities or RDF statements. For each related entity e , we update the description of blank node with $\beta^k * Desc(e)$ where k denotes the number of times the process has been repeated. For statement s whose subject is b , we add the object of s to the remaining nodes of b . For example, in Fig. 4, the record of blank node $b1$ has two remaining nodes $\{b1, c5\}$ and is thus partitioned to the reducers with $(b1, p6, c1)$ and $Desc(c5)$ respectively. In the reducer which loads $(b1, p6, c1)$, $p6$ and $c1$ is added to the remaining nodes of $b1$. Meanwhile, $Desc(b1)$ is updated by $\beta^k * Desc(c5)$ in another reducer. Notice that there exist duplicated records in the output. So, a combinator is needed to integrate results.

It also should be noticed that we do not consider the cycle descriptions in this process. We store the route of nodes which have been calculated and ignore all new nodes appearing in the route.

The process of constructing descriptions for blank nodes should be repeated k times. According to our experiments, five times of iteration is usually enough to converge.

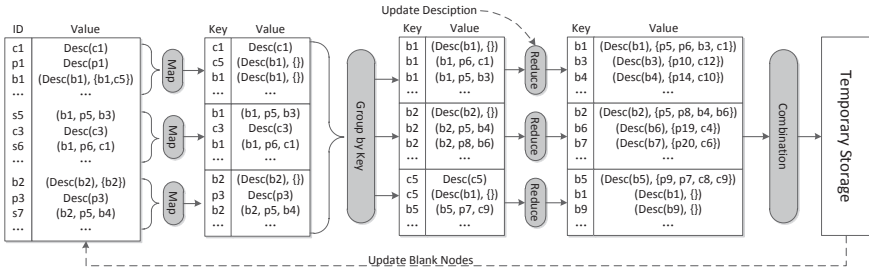


Fig. 4. Example data flow of construction of descriptions for blank nodes

5 Exchanging Information with Neighbors

The construction of virtual document needs both local descriptions and neighbor information. This section considers to use information of neighbors to update the description of each entity for constructing virtual document. The following two equations give the definition of virtual document:

$$VD(e) = Desc(e) + \gamma * Neigh(e) \quad (5)$$

$$Neigh(e) = \sum_{e' \in SN(e)} Desc(e') + \sum_{e' \in PN(e)} Desc(e') + \sum_{e' \in ON(e)} Desc(e') \quad (6)$$

where $SN(e)$ denotes the set of predicates and objects in the RDF statements whose subject is e , $PN(e)$ denotes the set of subjects and objects in the RDF statements whose predicate is e and $ON(e)$ stands for the set of subjects and predicates in the RDF statements whose object is e . We define γ as the repeat times of the MapReduce process for blank nodes and let $\gamma = 0.1$. But for some cases that most local information consist of trivial serial numbers or other random tokens, γ should be increased.

The calculation process contains two stages. The first stage is to notice each node with its neighbors and the second stage exchanges the descriptions between the neighbors.

Fig. 5 shows an example data flow. For each RDF statement (s, p, o) , the map function of the first stage generates three new key-value pairs: $(s, \{p, o\})$, $(p, \{s, o\})$ and $(o, \{s, p\})$. After that, for every entity e and blank node b , all neighbors can be obtained in the reduce function. For example, an RDF statement $(b1, p5, b3)$ is a record from input, after mapping, $(b1, \{p5, b3\})$, $(p5, \{b1, b3\})$ and $(b3, \{b1, p5\})$ are partitioned into three different reducers. In the reducer loading $(b1, \{p5, b3\})$, $b1$ gets to know that there exist neighbors $p5$ and $b3$ and adds them in a temporary structure. The outputs of reducers are stored in a temporary storage waiting the map function of the second stage to read.

With the locations of neighbors, every node sends its description in the second stage. For the output value $(Desc(c1), \{b2, p4, p1\})$ in the temporary storage, $c1$ sends its description $Desc(c1)$ to $b2$, $p4$ and $p1$. Thus, in the reduce function, every entity gets all the descriptions of the neighbors and updates its own description.

Because of frequency skew, some reducers may meet an unbalanced workload so that the whole calculation process is delayed heavily. Consequently, we count the appearing time of each entity in all RDF statements and find those with the highest frequency. Then we arrange some specific reducers to calculate the entities with the top-frequency.

6 Matching Virtual Documents

In the final stage, we calculate the similarities between virtual documents with the TF/IDF technique [14]. The value of TF can be easily calculated for a specific

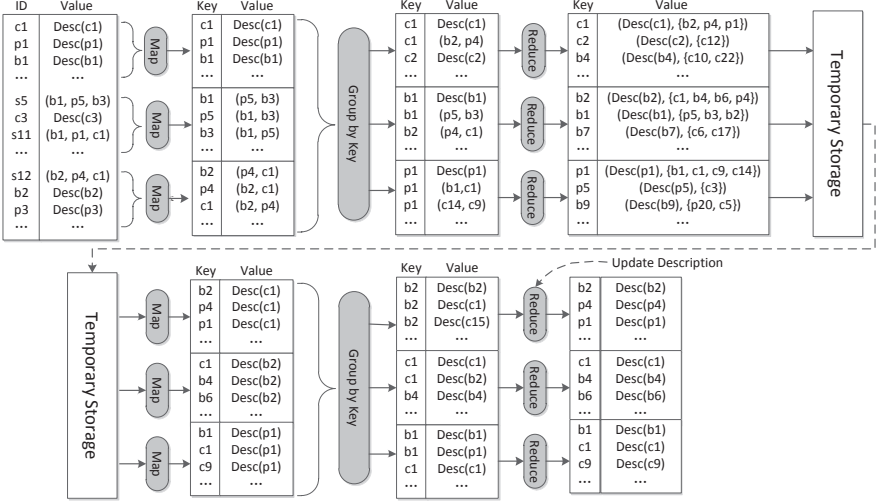


Fig. 5. Example data flow of exchanging information with neighbors

virtual document. To obtain the value of IDF, we build an additional MapReduce process for calculating the frequency of each word, and the number of virtual documents can be obtained in the preprocessing by counting the number of entities. Cosine similarity is used to measure the similarity. Equation (7) gives the function:

$$\text{sim}(e_1, e_2) = \frac{VD(e_1) \times VD(e_2)}{|VD(e_1)| * |VD(e_2)|} \quad (7)$$

Given a threshold θ , we define (e_1, e_2) be an ontology matching alignment where $\text{sim}(e_1, e_2) > \theta$.

Consider the objective of ontology matching, in the stage of matching virtual documents, we ignore the instances. However, if we calculate similarity for every two virtual documents respectively, lots of time would be wasted. So reducing the calculation space is necessary. In this stage, we propose a word-partition-based method to filter unnecessary matchings. For each description $\{(word_1, score_1), (word_2, score_2), \dots, (word_n, score_n)\}$, we normalize *scores* in $[0, 1]$ and rank $(word, score)$ pairs according to the value of *scores*. Thus, $score_1 \geq score_2 \geq \dots \geq score_n$. We define the important words as the set of words $\{word_1, word_2, \dots, word_i\}$, where i is the minimal integer which satisfy $score_1 + score_2 + \dots + score_i \geq \delta$. δ is a fixed rational number in $[0, 1]$.

Fig. 6 explains the process. Each mapper ranks the words and put the top words into the keys. For example, we select three words $word_1, word_2, word_3$ for entity c_1 in the map phase and generate new key-value pairs: $(word_1, Desc(c_1))$, $(word_2, Desc(c_1))$ and $(word_3, Desc(c_1))$. After partitioning, descriptions of entities c_1, c_8 and c_5 group into the same reducer with key $word_1$. Then we use Equation (7) to calculate the similarity between each two of them.

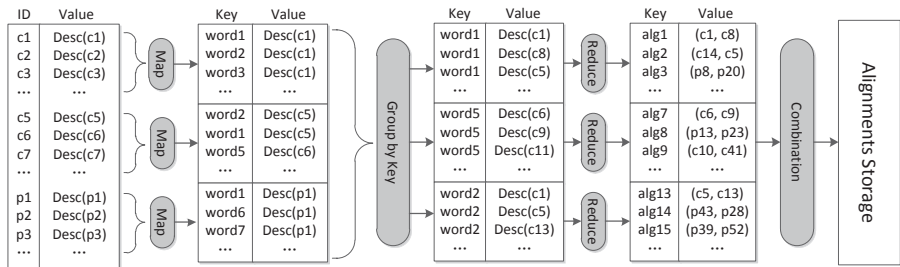


Fig. 6. Example data flow of matching virtual documents

Workload Balance. We load the frequency of words in the memory and construct a customized partitioner to choose the corresponding reducer. The words should be distributed in each node according to the frequency as average as possible. But for those with too high frequency, it is very hard to arrange or even split them. In this case, we assign one or more reducers to compute these high-frequency words while ignoring other keys. But we also allow them to choose reducers randomly if the number of computing nodes is too small.

7 Evaluation

We developed a parallel computing system, called V-Doc+, for our approach. V-Doc+ is based on the Hadoop framework², which provides an open-source software for scalable and distributed computing. Every mechanism of MapReduce corresponds to a process in Hadoop implementation. Given rich libraries, programmers are allowed to implement customized data structure, input/output record format, map/reduce function, and the way to partition.

In our program, each stage discussed above was implemented in one or more map/reduce functions. Particularly, some supporting functions, such as word statistics and combination, were added to the proper places in the whole implementation.

We ran our experiments on 10-node cluster and a Gigabit Ethernet interconnect. The NameNode equipped with an Intel processor with six 2.80GHz cores and 12M cache while the storage has 32GB memory and 2TB hard disk. For JobTracker and slave, the CPU is all Intel Quad Core and 2.4GHz/12M cache. The storage of JobTracker and slave is a little smaller than NameNode, which has 24GB memory and 2T hard disk. For compatibility consideration, we installed 0.21.0-version of Hadoop which is based on JDK v1.6.0 on Redhat Enterprise Linux Server 6.0 system. All stage of MapReduce process can be monitored in a Web browser.

According to experiments with varied parameters and the optimal result generated, we configured the parameters as follows: for calculating description stage, we set $\alpha_1 = 1$, $\alpha_2 = 0.5$, $\alpha_3 = 0.25$, $\alpha_4 = 0$ and $\beta = 0.5$. For exchanging information of neighbors stage and matching virtual documents stage, $\gamma = 0.1$ and

² <http://hadoop.apache.org/>

$\delta = 0.75$ respectively. In practice, α should be configured differently due to the fact of datasets. Particularly, if there is no differentiation between local names, α_1 should be lower or even be 0.

Preparation. Before starting experiments, some preprocessing were built. Firstly, we cached stopwords and entities/words frequency statistics in the memory. Also, we formatted the file system before each experiment starting and made sure that there was no other programs running on every computing node. For distinguishing the source of the record easily, the identifiers of entities, blank nodes and RDF statements were attached with the ontology name.

7.1 Datasets

According to our investigation, we chose two datasets according to their sizes: the Food Ontology in OAEI 2007 and FMA vs. GALEN. The reasons are as follows:

1. Our approach is designed to match large ontologies. Due to the network cost and repeated MapReduce job initialization, for small ontologies, it may perform worse than other tools. So, large ontologies are more suitable.
2. The last Food Ontology version and the results of the participants were published in OAEI 2007. So we cannot obtain this dataset after OAEI 2007.
3. Although OAEI publishes the campaign results of all tracks, such as Anatomy whose size is also suitable, we find no reference mappings for other size-suitable datasets so that we cannot evaluate their precisions and recalls.

Table 1 shows the statistical data of the Food Ontology which has lots of multi-lingual texts. It contains two ontologies: NALT and AGROVOC. NALT is developed by United Nations Food Organization and AGROVOC comes from Agriculture Organization. After the end of OAEI 2007, the results of the participants and the gold standard used to evaluate precision and recall are published on the OAEI website³. We downloaded them and used the gold standard to calculate precision, recall and value of F1 method in our experiment.

Table 1. Statistical data of the Food Ontology

Ontology	Classes and Properties	Statements
NALT	42,326	174,278
AGROVOC	28,439	319,662

Table 2. Statistical data of FMA vs. GALEN

Ontology	Classes and Properties	Statements
FMA	72,659	576,389
GALEN	9,596	59,753

For another dataset, we matched FMA ontology and GALEN ontology. FMA is more larger than GALEN. Unfortunately, we cannot find other ontology matching tools publishing their results of FMA vs. GALEN matching. Also,

³ <http://oei.ontologymatching.org/2007/>

no gold standard was found. Consequently, for FMA vs. GALEN, we only ran our experiment program on it and showed the run time and the speedup for different numbers of compute nodes. The statistical data of FMA vs. GALEN is showed in Table 2.

7.2 Experimental Results

The goal of our evaluation is threefold. Firstly, it is necessary to test precision, recall and F-Measure although the strength of V-Doc+ is its good efficiency. To test it, we investigated several ontology matchers and compared V-Doc+ with their performance. Secondly, we want to show how much the run time was reduced comparing with the non-parallel matchers. Thirdly, we showed the speedup on different computing nodes environment and presented the run time for each stage in our approach.

We evaluated V-Doc+ on precision, recall and F-Measure on the Food Ontology from OAEI 2007 [5] with some matchers: Falcon-AO, DSSim, RiMOM, Prior+ and COMA. One is Falcon-AO [8], which integrated three matchers: I-Sub, V-Doc and GMO, where V-Doc is a virtual-document-based technique which runs in a non-parallel way and GMO is a graph matching technique based on structural similarity. To match large ontologies with limited memory, Falcon-AO also constructed a divide-and-conquer approach which can partition entities into small clusters. DSSim [21] gave a multi-agent system to solve the ontology matching problem while considering uncertainty. The main approach of DSSim was to use different domains for finding ontology mappings. RiMOM [10] integrated multiple matchers to improve effectiveness by combining both literal and structure features. Another matcher Prior+ [13] is an adaptive ontology matching tool which was based on several different techniques, such as IR-based similarity and neural network. Like Falcon-AO, COMA [3] also considered the block matching and provided a fragment-based matcher to solve the large ontology matching problem. Differently, it partitioned data represented as trees.

F-Measure. We gave the performance of our experiment on the Food Ontology using F-Measure to assess the results. Firstly, we calculated precision and recall according to the gold standard that OAEI provides. Then, the value of F-Measure was calculated using the following equation:

$$F\text{-Measure} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (8)$$

Fig. 7 shows the result of comparison between V-Doc+ and other five matchers on the Food Ontology. From the figure, we observe that the precision, recall and f1-measure of V-Doc+ is no better than some of matchers. It mostly dues to the combination of varied algorithms for other tools while one in V-Doc+. However, they are all lower than Falcon-AO's. The main reason is that Falcon-AO combined several matchers, including I-Sub, V-Doc and GMO, where V-Doc is a non-parallel implementation of the virtual document technique which has the similar precision and recall with V-Doc+.

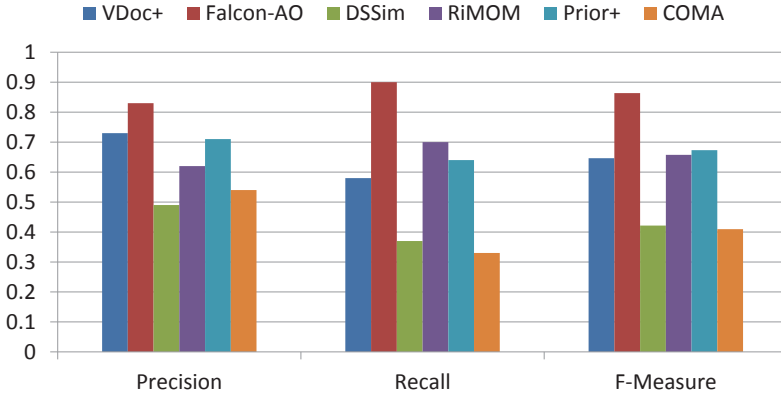


Fig. 7. Precision, recall and F1-Measure on the Food Ontology

Run Time. The strength of our approach is its efficiency. Table 3 shows the run times for 10-node cluster on the Food Ontology. V-Doc+ only spent ten minutes on running. Among other matchers, the one cost the least is Prior+ which spent 1.5 hours while DSSim cost one week which is the slowest. Consequently, although V-Doc+ does not achieve the best F-Measure, the parallelization makes it much faster than others.

Table 3. Run times comparison among V-Doc+, Falcon-AO, DSSim, RiMOM and Prior+ on the Food Ontology

Run time	V-Doc+	Falcon-AO	DSSim	RiMOM	Prior+
	10 min	6 h	1 week	4 h	1.5 h

To analyze each stage of the whole approach, we calculated the run time for details presented in Fig. 8, where 10-node cluster is used. For both the Food Ontology and FMA vs. GALEN, constructing descriptions and calculating similarities spent the most of time.

In order to evaluate the speedup, we calculated the run time for varied cluster sizes. For each dataset, we ran our program on 2, 4, 8, 10 nodes environment. Fig. 9 shows the result. In the figure, we see that the run time keeps reducing while increasing the number of compute nodes. But we also notice that the growth trend of efficiency is reducing. Fig. 10 shows the speedup which also reflects the reducing growth of efficiency with increased cluster size. From 8-node to 10-node, the speedup tends to be unchanged. An interesting thing is that the speedup on the Food Ontology is smaller than that on FMA vs. GALEN. The main reason is that, for any dataset, our approach must repeat the MapReduce process several times on constructing the descriptions for blank nodes. Although other MapReduce process stages on the Food Ontology cost less time than that on FMA vs. GALEN, they share the similar run time on the blank node stage, which leads to a low speedup.

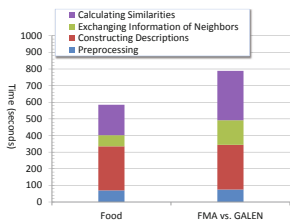


Fig. 8. Run time of each component on a 10-node cluster

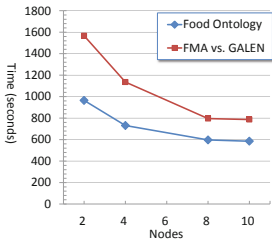


Fig. 9. Run time on different cluster sizes

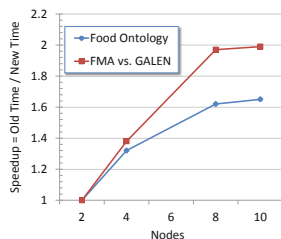


Fig. 10. Speedup on different cluster sizes

Some small scale datasets (less than 1,000 classes and properties and less than 3,000 statements) have also been tested for the run time and the speedup, but the result is not so good. Due to the network cost and the repeated reading of files, the whole process cost as much time as other tools.

8 Conclusion

Matching large ontologies is an inevitable obstacle for data fusion and only a few ontology matchers can finish matching task in a satisfiable run time. MapReduce is a wildly used computing framework for parallel computation, and it has been used in a number of fields. However, there is few studies on matching ontologies using MapReduce. In this paper, we proposed a 3-stage parallelized ontology matching method, called V-Doc+, using virtual document technique based on the MapReduce framework, which largely reduces the run time from hours to minutes.

Each stage of our approach establishes several MapReduce processes. For blank nodes, the descriptions are updated iteratively by emitting neighbors to the reducers. For similarities, we calculate the high-weight words in the descriptions of entities by ranking the frequency and emit the entities to reducers. Also, we considered the workload balance. The frequency of entities and words in RDF statements is calculated in the preprocessing. Then the statistical data of frequency is loaded in the memory to assist automatically partition.

For performance test, we conducted experiment on two large real datasets and the results showed a good efficiency and moderate precision and recall. For the Food Ontology from OAEI 2007, V-Doc+ used ten minutes to finish the task while other tools spent hours even weeks. The speedup with increased computing nodes is also illustrated.

Currently, a number of matchers obtain good precision and recall by combining multiple matchers. However, our approach is restricted to a certain linguistic matching algorithm. In the future work, we look forward to integrating a new parallelized algorithm based on ontology structures to improve precision and recall.

Acknowledgements. This work is supported in part by the National Natural Science Foundation of China under Grant Nos. 61003018 and 61021062, in part by the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20100091120041, and also in part by the National Science Foundation of Jiangsu Province under Grant No. BK2011189. We appreciate Jianfeng Chen for conduction on MapReduce programming.

References

1. Bleiholder, J., Naumann, F.: Data Fusion. *ACM Computing Surveys* 41(1), 1–41 (2008)
2. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM* 51(1), 107–113 (2008)
3. Do, H., Rahm, E.: Matching Large Schemas: Approaches and Evaluation. *Information Systems* 32(6), 857–885 (2007)
4. Euzenat, J., Ferrara, A., Meilicke, C., Nikolov, A., Pane, J., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Trojahn, C.: Results of the Ontology Alignment Evaluation Initiative 2010. In: *ISWC Workshop on Ontology Matching* (2010)
5. Euzenat, J., Isaac, A., Meilicke, C., Shvaiko, P., Stuckenschmidt, H., Šváb, O., Svátek, V., Hage, W., Yatskevich, M.: First Results of the Ontology Alignment Evaluation Initiative 2007. In: *ISWC Workshop on Ontology Matching* (2007)
6. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
7. Gross, A., Hartung, M., Kirsten, T., Rahm, E.: On Matching Large Life Science Ontologies in Parallel. In: Lambrix, P., Kemp, G. (eds.) *DILS 2010*. LNCS, vol. 6254, pp. 35–49. Springer, Heidelberg (2010)
8. Hu, W., Qu, Y., Cheng, G.: Matching Large Ontologies: A Divide-and-Conquer Approach. *Data & Knowledge Engineering*, 140–160 (2008)
9. Jean-Mary, Y., Shironoshita, E., Kabuka, M.: Ontology Matching with Semantic Verification. *Journal of Web Semantics* 7(3), 235–251 (2009)
10. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering* 21(8), 1218–1232 (2009)
11. Mork, P., Bernstein, P.: Adapting a Generic Match Algorithm to Align Ontologies of Human Anatomy. In: *Proceedings of the 20th International Conference on Data Engineering*, pp. 787–790 (2004)
12. Moutselakis, E., Karakos, A.: Semantic Web Multimedia Metadata Retrieval: A Music Approach. In: *13th Panhellenic Conference on Informatics*, pp. 43–47 (2009)
13. Mao, M., Peng, Y., Spring, M.: An Adaptive Ontology Mapping Approach with Neural Network Based Constraint Satisfaction. *Web Semantics: Science. Services and Agents on the World Wide Web* 8(1), 14–25 (2010)
14. McGill, M., Salton, G.: *Introduction to Modern Information Retrieval*. McGraw-Hill (1983)
15. Peukert, E., Berthold, H., Rahm, E.: Rewrite Techniques for Performance Optimization of Schema Matching Processes. In: *Proceedings of 13th International Conference on Extending Database Technology*, pp. 453–464. ACM Press, New York (2010)
16. Qu, Y., Hu, W., Cheng, G.: Constructing Virtual Documents for Ontology Matching. In: *15th International World Wide Web Conference*, pp. 23–31. ACM Press, New York (2006)

17. Rahm, E.: Towards Large-Scale Schema and Ontology Matching. *Data-Centric Systems and Applications, Part I*, 3–27 (2011)
18. Rosse, C., Mejino, L.: The Foundational Model of Anatomy Ontology. In: Burger, A., Davidson, D., Baldock, R. (eds.) *Anatomy Ontologies for Bioinformatics: Principles and Practice*, vol. 6, Part I, pp. 59–117. Springer, Heidelberg (2008)
19. Stoilos, G., Stamou, G., Kollias, S.: A String Metric for Ontology Alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) *ISWC 2005. LNCS*, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
20. Vernica, R., Carey, M., Li, D.: Efficient Parallel Set-Similarity Joins Using MapReduce. In: *SIGMOD 2010 Proceedings of the 2010 International Conference on Management of Data*, pp. 495–506. ACM Press, New York (2010)
21. Vargas-Vera, M., Nagy, M.: Towards Intelligent Ontology Alignment Systems for Question Answering: Challenges and Roadblocks. *Journal of Emerging Technologies in Web Intelligence* 2(3), 244–257 (2010)
22. Wang, P., Zhou, Y., Xu, B.: Matching Large Ontologies Based on Reduction Anchors. In: *Proceedings of International Joint Conferences on Artificial Intelligence*, pp. 2343–2348 (2011)
23. Zhang, S., Bodenreider, O.: Hybrid Alignment Strategy for Anatomical Ontologies: Results of the 2007 Ontology Alignment Contest. In: *ISWC Workshop on Ontology Matching* (2007)

Semantic Flow Networks: Semantic Interoperability in Networks of Ontologies

Valeria Fionda and Giuseppe Pirr6

KRDB, Free University of Bolzano-Bozen
{fionda,pirro}@inf.unibz.it

Abstract. In an open context such as the Semantic Web, information providers usually rely on different ontologies to semantically characterize contents. In order to enable interoperability at a semantic level, ontologies underlying information sources must be linked by discovering alignments, that is, set of correspondences or mappings. The aim of this paper is to provide a formal model (i.e., *Semantic Flow Networks*) to represent networks of ontologies and alignments with the aim to investigate the problem of composite mapping discovery. *Semantic Flow Networks (SFN)* differ from other models of networks of ontologies for two main aspects. *SFN* consider constraints over mappings that are necessary to take into account their dependencies. Moreover, a different notion of mapping, that is, compound mapping is considered. Complexity results and a CSP formulation for composite mapping discovery are provided.

1 Introduction

The Semantic Web is an extension of the current Web and it is aimed at providing a new class of services through which software agents can meaningfully access data, interact and interoperate in order to fulfil complex tasks. The main pillar of this new class of intelligent services and applications are ontologies. An ontology is a formal representation of a particular knowledge domain in terms of concepts, relations among concepts and axioms [8]. Ontologies provide the necessary semantic underpinning for endowing data with machine-processable annotations. An ontology inherently contains some degree of subjectivity since its definition depends on different factors among which the purpose of its usage and the background of the ontology engineer. In an open context such as the Semantic Web, this will inevitably raise issues when applications relying on different ontologies need to interact. Ontology matching [6] aims at providing, by ontology alignments (i.e., sets of correspondences or mappings), interpretations of how a particular piece of knowledge in an ontology can be linked to another one in a different ontology. Hence, interoperability among applications will occur at semantic level since it will be possible to automatically interpret pieces of knowledge in a new and denser space of semantic links enabled by alignments. Alignments can be exploited for different purposes as, for instance, ontology merging [13], translating queries among different information sources or routing queries in Peer-to-Peer networks [10].

Despite the potential huge number of online available ontologies, it is likely that some of them will emerge and bring a certain level of agreement on particular pieces of knowledge. This is the case, for instance, of the Friend of a Friend (FOAF) ontology,

which is largely adopted when there is the need to express information about people, the links among them and the things they create and do. From an ontology matching perspective, discovering alignments with popular ontologies can result in having a higher interoperability in a network of ontologies. Although ontology matching [6] is a very relevant problem, there is also the need to study and trace on a more abstract perspective the boundary of the (re)use of ontology alignments. There exists some initiative in this direction such as the work by Jiang et al. [9] where the problem of information fluidity in a network of alignments has been studied. Other initiatives (e.g., [219]) consider network of ontologies and alignments from a logical perspective. In [18] an empirical study on the precision of mapping composition has been conducted. In a more recent work [5], an algebra for ontology alignment relations has been defined which, on one side, introduces a rich set of relations between ontology entities and, on the other side, also investigates how to compose alignments.

This paper introduces the notion of *Semantic Flow Network (SFN)*. A *SFN* is used to formally define network of ontologies interlinked by alignments. *SFN* differ from related work for two main aspects. First, we consider constraints over mappings (i.e., XOR and AND) that are necessary to take into account their dependencies. Second, we consider a different notion of mapping, that is, compound mapping. Building up on the *SFN* framework, two main discovery tasks, that is *Free Mapping* and *Perfect Mapping*, will be introduced. The former is the task of finding a sequence of mappings so that entities in a given ontology can be linked to entities in other ontologies. The latter aims at linking input entities to a specific set of entities in other ontologies. These tasks introduce two interesting problems in a *SFN*, that is, *mapping existence* and *mapping selection*. With mapping existence it is intended the process of deciding whether there exists a solution (i.e., a sequence of mappings) for the *Free Mapping* (resp., *Perfect Mapping*) problem. On the other hand, mapping selection is concerned in finding the *optimal* solution to these problems. The contributions of this paper are as follows:

- The idea of *Semantic Flow Network* is introduced to represent networks of ontology entities linked by mappings. Besides, *Mapping Existence* and *Mapping Selection* for the tasks of *Free Mapping* and *Perfect Mapping* are also introduced.
- The classical representation of a mapping will be generalized to the case in which input and output are sets of entities.
- Some complexity results for *Mapping Existence* and *Mapping Selection* are provided by considering XOR and AND constraints among mappings.
- A Constraint Satisfaction Problem (CSP) formulation for the existence problems is provided.

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. Relevant notions along with their formal definitions are introduced in Section 3. Section 4 introduces the notion of *Semantic Flow Network* and describes *Free Mapping* and *Perfect Mapping* upon which the problems of mapping existence and selection will be defined. Some complexity results to these problems are given in Section 5. Section 6 provides a CSP formulation for the mapping existence problems. Section 7 draws some conclusions and sketches future work.

2 Related Work

This section reports on some recent work that dealt with networks of ontologies linked by alignments. In Jiang et al. [9], information fluidity has been tackled as a graph connectivity problem. Authors investigated the total amount of information flowing in a network of ontologies as a function of the number of alignments in the network. The NEON¹ project investigated some aspects related to networks of ontologies such as large-scale semantic applications in distributed organizations. In [19] three different semantics for distributed systems, defined as sets of ontologies interconnected by alignments, have been introduced. Upon these three semantics a mapping composition operator has been defined. Authors distinguish between syntactic composition and semantic composition, which requires semantic consistency to be preserved when indirectly linking two ontologies. Another interesting model of distributed knowledge-based system are Distributed Description Logics [2], which consist in ontologies linked by bridge rules. This approach tackles the problem of preserving the consistency of the distributed knowledge base by interpreting each ontology within a context. In [5], an algebra for ontology alignment relations has been introduced with some considerations on how mapping composition can be performed. In [18], chains of (near) equivalence relations of length two have been analysed with the aim to investigate if transitivity of these relations is preserved. An empirical analysis has been conducted in three different domains (i.e., Biomedicine, Cultural Heritage and Library Subject Headings) where human experts have been asked to evaluate composite mappings. It has been observed that the quality of the composition depends on the content of the ontologies and the quality of the basic mappings upon which the composition is built. Interestingly, in some cases (e.g., Bioportal) the precision of the composition exceeds the precision of the basic alignments. In [14], network of ontologies and alignments have been investigated by considering repositories of ontologies. In particular, the notion of hyperontology has been introduced, which serves to study (in)consistency propagation in connected alignments. The problem of reusing ontology alignments in a Peer-to-Peer network has been investigated in [10]. Mapping composition [7][12] is a relevant problem in database theory and in particular in schema mapping where it enables data integration and exchange [11]. In [1] an algorithm for implementing mapping composition has been defined.

The aim of this paper is to propose a formal model (i.e., *Semantic Flow Networks*) to represent networks of ontologies and alignments with the aim to investigate the problem of composite mapping discovery. Our formulation of network of ontologies and alignments differs from related work for two main aspects. First, we consider constraints over mappings (i.e., XOR and AND) that are necessary to take into account their dependencies. Second, we consider a different notion of mapping, that is, compound mapping. Compound mapping discovery has received little attention so far, even though it is a more powerful notion of mapping. Besides, the study of the complexity presented in this paper, traces the intrinsic difficulty of composite mapping discovery, which on its turn bounds, in the worst case, the efficiency of algorithms that can be designed. However, we provided a CSP formulation, which can benefit from the efficiency of existing solvers.

¹ <http://www.neon-project.org/>

3 Preliminaries

This section introduces general notions that provide the necessary background to introduce formal definitions and subsequent results.

We consider ontologies expressed in OWL, which is the W3C standard language for ontologies. In the following the term entity will be used to generically denote an ontology concept or relation. The generic set of entities defined in an ontology ω will be denoted by C_ω . In order to enable interoperability among ontologies it is necessary to discover alignments i.e., sets of mappings. To cope with this problem several ontology matching systems have been proposed (e.g., [16,17]). The reader can refer to Euzenat & Shvaiko 2007 [6] for a comprehensive discussion about the topic. We assume that alignments already exist and are stored in some repository such as the *Alignment Server* [3].

Definition 1 (Atomic mapping). *An atomic mapping is a tuple $\langle (c_i, \omega_i), (c_j, \omega_j), r, s \rangle$ where c_i (resp., c_j) is an entity belonging to the ontology ω_i (resp., ω_j), r is a semantic relation holding between c_i and c_j and $s \in [0, 1]$ is the confidence value associated to the relation² r .*

As discussed in Euzenat 2008 [5], r is a subset of relations $\mathcal{R} \in 2^\Gamma$, with $\Gamma = \{=, <, >, \bowtie, \perp\}$. The symbols $=, <, >$ express equivalence, more general and less general relations, respectively. The symbol \bowtie expresses an overlap relation between entities, which has to be interpreted as the case in which some instances are shared but no other relation can hold. The symbol \perp states that two entities are disjoint. Finally, Γ represents total uncertainty about the relation holding between entities.

Given a set of ontologies Ω , let $C = \bigcup_{\omega \in \Omega} C_\omega$ be the set of all entities appearing in these ontologies. Let $\Psi = \{(c, \omega) \mid \omega \in \Omega \wedge c \in C_\omega\}$ be the set of entities along with their ontologies. Given a subset $\mathcal{W} \subseteq \Psi$, $\mathcal{W}(\omega_j)$ denotes the set of elements $\{(c, \omega_j) \mid (c, \omega_j) \in \mathcal{W}\}$.

Definition 2 (Compound mapping). *Let C be a set of entities and ω_i, ω_j two ontologies, a compound mapping μ is a tuple $\langle \mathcal{I}, \mathcal{O}, \mathcal{R}, s \rangle$ where $\mathcal{I} \subseteq \mathcal{W}(\omega_i)$ and $\mathcal{O} \subseteq \mathcal{W}(\omega_j)$ are the set of entities that are taken in input and given as output, respectively. Besides, \mathcal{R} and s have the same meaning as in Definition 1.*

Example 3. *Let $\Omega = \{\omega_1, \omega_2\}$ and $C = C_{\omega_1} \cup C_{\omega_2} = \{c_1, c_2\} \cup \{c_3, c_4\} = \{c_1, c_2, c_3, c_4\}$ be the set of ontologies and entities, respectively. The tuple $\langle \mathcal{I}, \mathcal{O}, =, 1 \rangle$ with: $\mathcal{I} = \{(c_1, \omega_1), (c_2, \omega_1)\}$ and $\mathcal{O} = \{(c_3, \omega_2), (c_4, \omega_2)\}$, represents the mapping between the entities $\{c_1, c_2\}$ in ω_1 and the entities $\{c_3, c_4\}$ in ω_2 , characterized by an equivalence semantic relation and 1 as a confidence value.*

Note that an atomic mapping is a special case of compound mapping where \mathcal{I} and \mathcal{O} contain only one entity. Fig. 1 depicts a mapping, with round nodes representing input and output entities and the rectangle the relations \mathcal{R} and the confidence score s . Besides, edges model the direction in which information flows through μ . The mapping depicted in Fig. 1 expresses the fact that whenever the concept $c_2 \in \omega_1$ is used to interact with the information source relying on ω_2 , it is seen as $c_3 \in \omega_2$.

² The closer the score to 1 the higher the confidence that r holds.

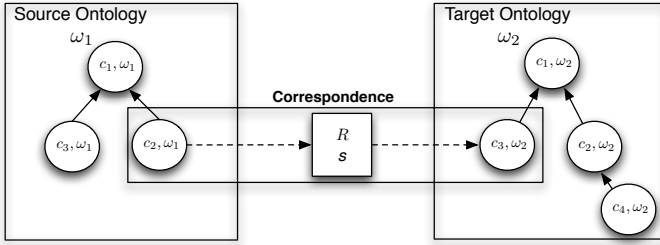


Fig. 1. Graphical representation of a mapping

In a network of ontologies interlinked by alignments it is also possible to perform some manipulation over alignments, and then over the mappings defined within, with the aim to discover new ones. In particular, in [5] the operations of inverse and composition of alignments have been introduced. The reader can refer to [5] for a comprehensive discussion about these notions.

3.1 Processing Ontology Mappings

Since a mapping, as previously defined, is just a declarative statement about the relation holding between entities in different ontologies, it is necessary from an operational point of view to provide a way to "process" mappings. Indeed, mapping systems are used to discover mappings and provide them for subsequent usage, which means that these declarative statements have to be somehow processed in targeted applications (e.g., ontology merging, query translation). We consider a function $\text{process}(\mu)$, which takes in input a mapping $\mu = \langle \mathcal{I}, \mathcal{O}, \mathcal{R}, s \rangle$ and processes it, that is, starting from the input entities \mathcal{I} of the mapping, returns the output entities \mathcal{O} . The function $\text{process}(\cdot)$ is crucial for exploiting alignments to preform composition since in order to discover composite mappings it is necessary to process already existing ones.

Definition 4 (Mapping composition). Let $\mathcal{M} = \{\mu_1, \mu_2, \dots, \mu_k\}$ be a set of mappings. A composition \mathcal{Y} over \mathcal{M} (indicated as $\mathcal{Y}_{\mathcal{M}}$) defines the order in which $\{\mu_1, \mu_2, \dots, \mu_k\}$ are processed by the function $\text{process}(\cdot)$.

For example, the composition $\mathcal{Y}_{\mathcal{M}} = \mu_2 \circ \mu_4 \circ \mu_1 \circ \dots \circ \mu_k$, corresponds to process in order the mappings $\mu_2, \mu_4, \mu_1, \dots, \mu_k$ (i.e., $\text{process}(\mu_2), \text{process}(\mu_4), \text{process}(\mu_1), \dots, \text{process}(\mu_k)$).

Given a set $\{\mathcal{Y}_{\mathcal{M}_1}, \dots, \mathcal{Y}_{\mathcal{M}_l}\}$ of compositions, it is possible to assign a score in order to rank different sets of compositions, by introducing a generic function *value*. As an example, *value* can take into account the confidence scores and/or the relations in the compositions. For instance, it can be: $\text{value}(\{\mathcal{Y}_{\mathcal{M}_1}, \dots, \mathcal{Y}_{\mathcal{M}_l}\}) = s_{\{\mathcal{M}_1, \dots, \mathcal{M}_l\}} = \prod_{i=1}^l s_{\mathcal{M}_i}$, with $s_{\mathcal{M}_i}$ being the *cumulative confidence score* obtained by applying an operator (e.g., multiplication) to the confidence scores of the compositions in \mathcal{M}_i .

3.2 Constraints over Mappings

It is unrealistic to assume that mappings in an alignment are all independent from one another. In general, mappings are OR-constrained, that is, whatever combination of them can be processed. However, in some circumstances, discussed below, the way mappings are processed has to respect some criteria. Here, we provide some hints about how constraints can be imposed for the particular scenario considered in the present work, that is, networks of ontologies in which new mappings can be deduced starting from the existing ones. Indeed, an interesting line of future research is to investigate how to discover constraints in ontology alignments. This is because constraints are something that is not generally given as a result of executing a matching algorithm but could be discovered leveraging both alignment information and axioms defined in the ontologies. For the purposes of this paper, to encode constraints among mappings, the operators XOR and AND will be used.

Mappings AND-constrained. An AND constraint among a set of mappings $\{\mu_1, \mu_2, \dots, \mu_k\}$ indicated as $\text{AND}(\mu_1, \mu_2, \dots, \mu_k)$ implies that if one mapping is processed then all of them have to be processed. This constraint is useful to decompose compound mappings. In particular, when a mapping has n inputs and m outputs, it can be decomposed in a set of $n \times m$ atomic mappings AND-constrained. In Fig. 2 it is reported the decomposition for n and m ranging from 1 to 2. The case $n = m = 1$ is reported in Fig. 1.

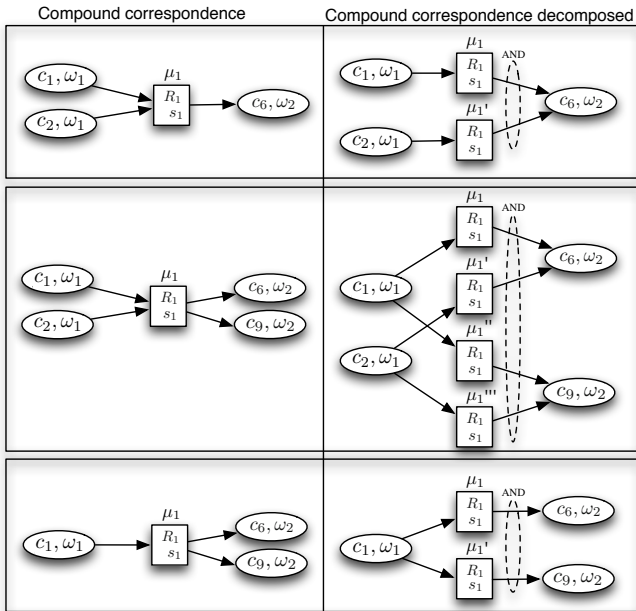


Fig. 2. Decomposition of a compound mapping

Mappings XOR-constrained. A XOR constraint over the set of mappings $\{\mu_1, \mu_2, \dots, \mu_k\}$ indicated as $\text{XOR}(\mu_1, \mu_2, \dots, \mu_k)$ implies that at most one of them can be processed. An example of XOR constraint is reported in Fig. 3. XOR constraints can be discovered, for instance, by looking at disjointness relations (available in OWL) between concepts in a target ontology (ω_2 in Fig. 3). As can be observed, there exist two mappings both taking as input (c_2, ω_1) and returning (c_9, ω_2) and (c_6, ω_2) , respectively. However, c_6 and c_9 are defined as disjoint in ω_2 . In a dynamic scenario, as for instance when composing alignments, only one of the two mappings should be processed. Note that, XOR constraints can also be defined over sets of mappings AND-constrained when dealing with decomposed compound mappings.

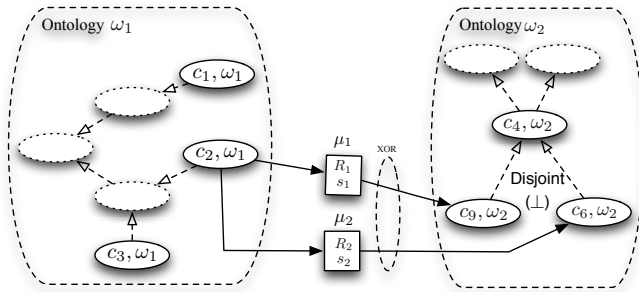


Fig. 3. An example of XOR constraint

Operations over existing alignments and in particular alignment composition are at the basis for the subsequent reasoning. In particular, given a network of ontologies it is interesting to decide whether certain kinds of problems such as: “there exist a composition, or a sequence of compositions, such that a set of concepts in an ontology ω_i can be mapped into a set of concepts in another ontology ω_j ?” can be solved.

4 Semantic Flow Networks

This section introduces the notion of *Semantic Flow Network (SFN)* that can be seen as a network of ontologies interconnected via alignments and augmented with constraints about how mappings contained in an alignment can be processed.

Definition 5 (Semantic Flow Network). A *Semantic Flow Network (SFN)* \mathcal{F} is a tuple $\langle C, \Omega, \mathcal{M}, \vartheta \rangle$ where Ω is a set of ontologies and C is the set of entities defined in the ontologies in Ω . \mathcal{M} is a set of mappings over C and Ω , and ϑ is a set of constraints over the mappings in \mathcal{M} .

Note that \mathcal{M} contains all the mappings contained in all the alignments in the network of ontologies.

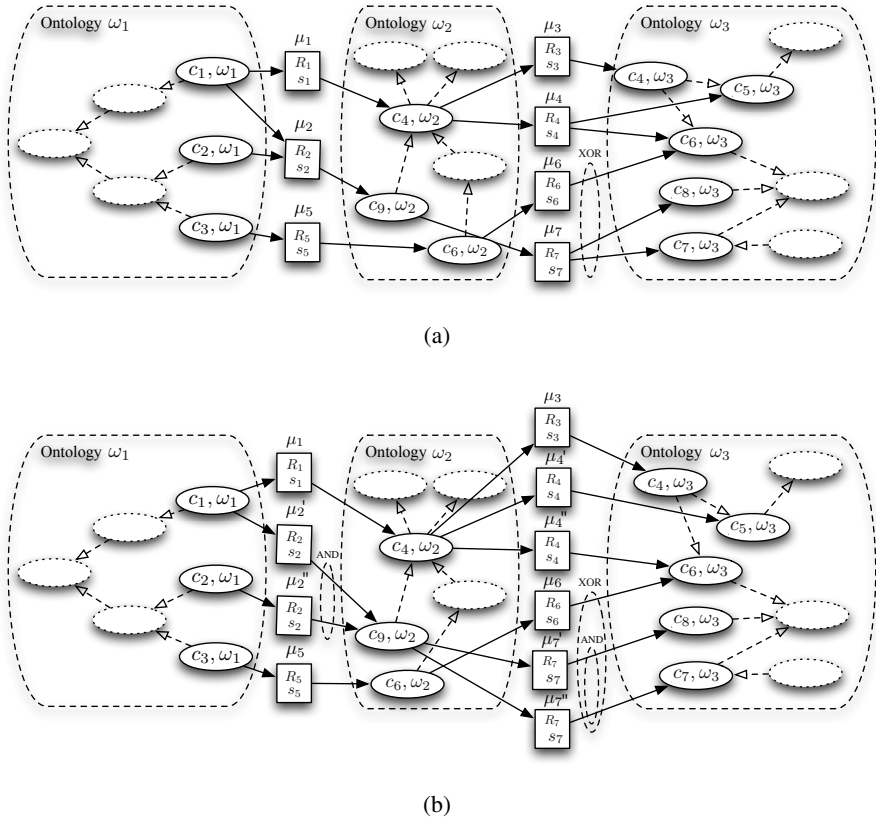


Fig. 4. A SFN (a) with compound mappings and (b) with decomposed mappings

Example 6. An example of SFN $\mathcal{F} = \langle C, \Omega, \mathcal{M}, \vartheta \rangle$ is depicted in Fig. 4(a) where $C = \{c_1, \dots, c_8\}$, $\Omega = \{\omega_1, \omega_2, \omega_3\}$, $\mathcal{M} = \{\mu_1, \mu_2, \dots, \mu_7\}$ and $\vartheta = \{XOR(\mu_6, \mu_7)\}$. Since \mathcal{F} involves compound mappings it can be represented using their decompositions. This leads to $\mathcal{F}' = \langle C, \Omega, \mathcal{M}', \vartheta' \rangle$ (see Fig. 4(b)) where the mappings μ_2, μ_4 and μ_7 are substituted with their decompositions. Moreover, the new set of mapping constraints is $\vartheta' = \{XOR(\mu_6, AND(\mu_7', \mu_7'')), AND(\mu_2', \mu_2''), AND(\mu_4', \mu_4''), AND(\mu_7', \mu_7'')\}$. \triangleleft

For sake of presentation, in the above example and in the remainder of this paper we only deal with direct alignments. However, for each alignment it is possible to compute its inverse, which in practice means including in the network additional edges from target to source entities.

4.1 Mapping Discovery Strategies in a SFN

The aim of a SFN is to enable interlinking entities in a network of ontologies by discovering new mappings. Note that simple path discovery algorithms are not enough since SFNs are not mere Directed Graphs as they also include constraints over mappings.

In a *SFN* two main discovery strategies have been identified. The first one is about discovering and composing mappings through which, a given set of entities can be linked to entities belonging to a given set of target ontologies. This task is useful when, for instance, there is no a-priori knowledge about the structure of the target ontologies but there are clues that these ontologies are somehow relevant. This leads to the notion of *Free Mapping*.

Definition 7 (Free Mapping). A *Free Mapping* is a triple $FM = \langle \mathcal{F}, \mathcal{C}_{in}, \Omega_{out} \rangle$, where $\mathcal{F} = \langle C, \Omega, \mathcal{M}, \vartheta \rangle$ is a *SFN* instance, $\mathcal{C}_{in} \subseteq \Psi$ is the set of input entities and $\Omega_{out} \subseteq \Omega$ is a set of target ontologies. *FM* encodes the fact that given \mathcal{C}_{in} and Ω_{out} , a set of mappings $\overline{\mathcal{M}} \subseteq \mathcal{M}$ and a set of compositions over them $\{\Upsilon_{\mathcal{M}_1}, \dots, \Upsilon_{\mathcal{M}_k}\}$, with $\mathcal{M}_i \subseteq \overline{\mathcal{M}}$, $\forall i \in \{1, \dots, k\}$ and $\overline{\mathcal{M}} = \bigcup_i \mathcal{M}_i$, have to be processed to obtain a set of entities $E \subseteq \Psi$, s.t., $(c, \omega_{out}) \in E$ iff $\omega_{out} \in \Omega_{out}$.

Another problem we consider concerns the discovery of mappings through which a given set of entities can be mapped into a specific set of entities in specific target ontologies. For instance, let's consider the FOAF ontology, which is largely adopted to express relations among people. One can be interested in discovering mappings between entities in her ontology with some FOAF entities such as *Person*. This will certainly enable, via FOAF, to discover additional mappings with other ontologies already linked to FOAF. Here the notion of *Perfect Mapping* comes into play.

Definition 8 (Perfect Mapping). A *Perfect Mapping* is a triple $PM = \langle \mathcal{F}, \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$, where \mathcal{F} and \mathcal{C}_{in} have the same meaning as in Definition 7 and $\mathcal{C}_{out} \subseteq \Psi$ is the set of specific target entities. *PM* encodes the fact that given \mathcal{C}_{in} and \mathcal{C}_{out} a set of mappings $\overline{\mathcal{M}} \subseteq \mathcal{M}$ and a set of compositions over them $\{\Upsilon_{\mathcal{M}_1}, \dots, \Upsilon_{\mathcal{M}_k}\}$ have to be processed to obtain entities in \mathcal{C}_{out} .

Given a *Free Mapping* instance *FM* (resp., *Perfect Mapping* instance *PM*), a relevant problem is to determine if there exists a solution i.e., a subset of mappings $\overline{\mathcal{M}} \subseteq \mathcal{M}$ and a set of compositions over these mappings $\{\Upsilon_{\mathcal{M}_1}, \dots, \Upsilon_{\mathcal{M}_k}\}$ able to satisfy the requirements of *FM* (resp., *PM*). Besides, another relevant problem concerns the selection of the optimal subset of mappings $\mathcal{M}^* \subseteq \mathcal{M}$ and their corresponding set of compositions $\{\Upsilon_{\mathcal{M}_1}^*, \dots, \Upsilon_{\mathcal{M}_k}^*\}$. The former problem will be described in detail in Section 4.2, while the latter will be analyzed in Section 4.3.

4.2 Mapping Existence

Mapping existence is the problem of deciding whether a given *Free Mapping* instance or *Perfect Mapping* instance admits a solution.

Consider the set of mappings $\overline{\mathcal{M}} = \bigcup_i \mathcal{M}_i$ and a set $\{\Upsilon_{\mathcal{M}_1}, \dots, \Upsilon_{\mathcal{M}_k}\}$ of compositions, and let $\sigma_{\overline{\mathcal{M}}}$ be a *processing sequence* defining the order in which mappings in $\overline{\mathcal{M}}$ are processed by the function `process(.)`. Note that to each set of compositions can correspond more than one processing sequence but it is always possible, starting from the processing sequence, to construct the corresponding set of compositions.

Let $\mathcal{T}_0 = \mathcal{C}_{in}$ denote the set of entities at the beginning and $\mu_i = \langle \mathcal{I}_i, \mathcal{O}_i, R_i, s_i \rangle$ be the *i*-th mapping in $\sigma_{\overline{\mathcal{M}}}$. Then, $\mathcal{T}_i \subseteq \Psi$ denote the set of entities obtained after processing μ_i , that is:

$$\mathcal{T}_i = (\mathcal{T}_{i-1} \setminus \mathcal{I}_i) \cup \mathcal{O}_i. \quad (1)$$

Moreover, let the set $\mathcal{D}_i \subseteq \Psi$ (with $\mathcal{D}_0 = \emptyset$) denote the set of entities used as input in some mappings in the previous $i-1$ steps, that is:

$$\mathcal{D}_i = \mathcal{D}_{i-1} \cup \mathcal{I}_i. \quad (2)$$

The sequence $\sigma_{\overline{\mathcal{M}}} = \mu_1, \dots, \mu_k$ is *legal* w.r.t. *FM* (*PM*, resp.) if:

- (i) $\mathcal{I}_i \subseteq \mathcal{T}_{i-1} \cup \mathcal{D}_{i-1}, \forall i \in \{1, \dots, k\}$.
- (ii) all the XOR and AND constraints defined in ϑ are satisfied, which means that for each XOR constraint over a set of mappings, at most one of them is processed (i.e., belongs to $\sigma_{\overline{\mathcal{M}}}$) and for each AND constraint over a set of mappings, none or all of them are processed (i.e., belongs to $\sigma_{\overline{\mathcal{M}}}$).

A legal processing sequence $\sigma_{\overline{\mathcal{M}}} = \mu_1, \dots, \mu_k$ is a *solution sequence* to *FM* (resp., *PM*) iff $(c, \omega) \in \mathcal{T}_k \Rightarrow (\omega \in \Omega_{out})$ (resp., $\mathcal{C}_{out} \subseteq \mathcal{T}_k$) holds. A set $\{\mathcal{Y}_{\mathcal{M}_1}, \dots, \mathcal{Y}_{\mathcal{M}_k}\}$ of compositions is a *solution set* to *FM* (resp., *PM*) iff there exists at least one *solution sequence* over the associated $\overline{\mathcal{M}}$.

Definition 9 (Mapping Existence Problem). *Given a Free Mapping instance FM (a Perfect mapping instance PM, resp.), the Mapping Existence Problem (MEP) amounts at deciding if FM (PM, resp.) is feasible, that is, if a solution sequence (and, consequently, a set of compositions) for FM (PM, resp.) exists.*

Example 10. *Given the SFN instance \mathcal{F} depicted in Fig. 5 consider the Free Mapping instance $FM = \langle \mathcal{F}, \mathcal{C}_{in}, \omega_4 \rangle$, with $\mathcal{C}_{in} = \{(Citizen, \omega_1)\}$. Possible solutions to *FM* are the sequences:*

- $\sigma_{\{\mu_2, \mu_5\}} = \mu_2, \mu_5$ (and the set of compositions $\{\mathcal{Y}_{\{\mu_2, \mu_5\}} = \mu_2 \circ \mu_5\}$).
- $\sigma_{\{\mu_1, \mu_2, \mu_3, \mu_5\}} = \mu_2, \mu_5, \mu_1, \mu_3$ (and the set of compositions $\{\mathcal{Y}_{\{\mu_2, \mu_5\}} = \mu_2 \circ \mu_5, \mathcal{Y}_{\{\mu_1, \mu_3\}} = \mu_1 \circ \mu_3\}$) \triangleleft .

A solution sequence $\sigma_{\overline{\mathcal{M}}}$ to a Perfect Mapping instance is said to be *exact* iff $\mathcal{C}_{out} = \mathcal{T}_k$.

Definition 11 (Exact Mapping Existence Problem). *Given a Perfect Mapping instance *PM*, the Exact Mapping Existence Problem (EMEP) amounts at deciding if *PM* is exact-feasible, that is, if an exact solution to *PM* exists.*

Example 12. *Given the SFN instance \mathcal{F} depicted in Fig. 5 consider the Perfect Mapping instance $PM = \langle \mathcal{F}, \mathcal{C}_{in}, \mathcal{C}_{out} \rangle$, with $\mathcal{C}_{in} = \{(Citizen, \omega_1)\}$ and $\mathcal{C}_{out} = \{(Female, \omega_4)\}$. A possible solution to *PM* is the sequence $\sigma_{\{\mu_1, \mu_3\}} = \mu_1, \mu_3$ (and the set of compositions $\{\mathcal{Y}_{\{\mu_1, \mu_3\}} = \mu_1 \circ \mu_3\}$). Note that, this sequence is also an exact solution to *PM*. \triangleleft*

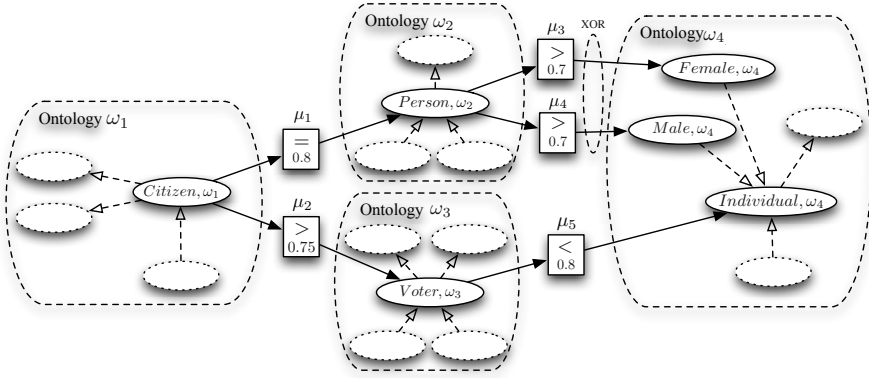


Fig. 5. An example of *Semantic Flow Network*

4.3 Optimal Mapping Selection

Optimal Mapping Selection is the problem of finding the optimal solution to a *Free Mapping* instance FM or a *Perfect Mapping* instance PM . The optimal solution to FM aims at finding the set of mappings $\mathcal{M}^* \subseteq \mathcal{M}$ and a set of compositions $\{\Upsilon_{\mathcal{M}_1^*}, \dots, \Upsilon_{\mathcal{M}_k^*}\}$ over \mathcal{M}^* , s.t., there does not exist $\hat{\mathcal{M}} \subseteq \mathcal{M}$ and a set of compositions $\{\hat{\Upsilon}_{\hat{\mathcal{M}}_1}, \dots, \hat{\Upsilon}_{\hat{\mathcal{M}}_k}\}$ over $\hat{\mathcal{M}}$ for which $value(\{\hat{\Upsilon}_{\hat{\mathcal{M}}_1}, \dots, \hat{\Upsilon}_{\hat{\mathcal{M}}_k}\}) > value(\{\Upsilon_{\mathcal{M}_1^*}, \dots, \Upsilon_{\mathcal{M}_k^*}\})$. Even in this case, two relevant problems can be defined.

Definition 13 (Optimal Mapping Selection Problem). *Given a FM instance (a PM instance, resp.), the Optimal Mapping Selection Problem (OMSP) amounts at finding the optimal solution set $\{\Upsilon_{\mathcal{M}_1^*}, \dots, \Upsilon_{\mathcal{M}_k^*}\}$, along with one of the possible solution sequences, to FM (resp., PM).*

Example 14. *If we admit the function value to be the cumulative confidence score over a mapping composition, by considering the Free Mapping instance FM reported in Example 10 the solution to the OMSP over FM is $\sigma_{\{\mu_2, \mu_5\}} = \mu_2, \mu_5$ along with the solution set $\{\Upsilon_{\{\mu_2, \mu_5\}} = \mu_2 \circ \mu_5\}$. This is because this solution set has the maximum cumulative confidence (i.e., $0.75 \times 0.8 = 0.6$) among all the possible solutions. \triangleleft*

An exact optimal solution for a *Perfect Mapping* instance PM is an exact solution providing the maximum value over all the possible exact solutions.

Definition 15 (Exact OMSP). *Given a Perfect Mapping instance PM , the Exact Optimal Mapping Selection Problem (EOMSP) amounts at finding the exact optimal solution set $\{\Upsilon_{\mathcal{M}_1^*}, \dots, \Upsilon_{\mathcal{M}_k^*}\}$, along with one of its solution sequences, to PM .*

Example 16. *Consider the Perfect Mapping instance PM reported in Example 12 the solution of the OMSP and of the EOMSP over PM is $\sigma_{\{\mu_1, \mu_3\}} = \mu_1, \mu_3$ and $\{\Upsilon_{\{\mu_1, \mu_3\}} = \mu_1 \circ \mu_3\}$ having the maximum cumulative confidence (i.e., 0.56). \triangleleft*

The existence and optimal selection problems are crucial to enable large scale reuse of alignments and then mappings defined within. Therefore, in Section 5 some complexity results concerning these problems will be sketched.

5 Some Complexity Results

This section provides some complexity results for the mapping existence and mapping selection problems.

Theorem 17. *MEP and EMEP, built using XOR and AND operators to constraint mappings in alignments, belong to the NP class of decision problems.*

Proof. The certificate for all the three problems is a processing sequence $\sigma_{\overline{\mathcal{M}}}$ (see Section 4.2). Mappings in $\overline{\mathcal{M}}$ are a subset of the mappings in the input *SFN* thus the size of the certificate is polynomial in the total number of mappings. The satisfaction of XOR and AND constraints can be checked in polynomial time by scanning the sequence and it can also be checked in polynomial time (by using formulas 1 and 2) that the sequence is legal and that at the end of the processing sequence:

- in the case of MEP, only entities of the target ontologies or a superset of the required entities are obtained;
- in the case of EMEP, exactly the set of required entities is obtained.

Moreover, if we consider the class of *SFN* instances in which both XOR and AND constraints are used, the following results hold:

Theorem 18. *MEP and EMEP for the class of FM and PM instances obtained from SNF instances built using both XOR and AND operators to constraint mappings are NP-complete.*

In the following, hardness results are provided by reductions from the Satisfiability of Boolean formulas in conjunctive normal form (CNF). In particular, recall that deciding whether a Boolean formula in CNF is satisfiable is NP-hard [15]. Since we are interested in the existence of a solution we omit information about semantic relations and confidence values (we can consider without loss of generality all relations as equivalence and all confidence values as unitary).

Proof. (Sketch)

NP-Hardness. Let $\phi = c_1 \wedge \dots \wedge c_m$ be a generic boolean formula in CNF over the clauses c_1, \dots, c_m and the variables X_1, \dots, X_n . Starting from ϕ we build the *SFN* $\mathcal{F} = \langle C, \Omega, \mathcal{M}, \vartheta \rangle$ reported in Fig. 6. Note that, in Fig. 6 mappings are represented by edges. \mathcal{F} is build such that:

- $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$.
- $C = C_{\omega_1} \cup C_{\omega_2} \cup C_{\omega_3} \cup C_{\omega_4} = \bigcup_{X_i} \{x_i\} \cup \bigcup_{X_i} \{x_i^T, x_i^F\} \cup \bigcup_{c_j} \{c_j\} \cup \{\phi\}$. Thus, for each variable X_i we consider the three entities $x_i, x_i^T, x_i^F \in C$ and for each clause c_j an entity $c_j \in C$.
- Each variable X_i in ϕ is associated to the two mappings $\langle \{(x_i, \omega_1)\}, \{(x_i^T, \omega_2)\} \rangle$ and $\langle \{(x_i, \omega_1)\}, \{(x_i^F, \omega_2)\} \rangle$.

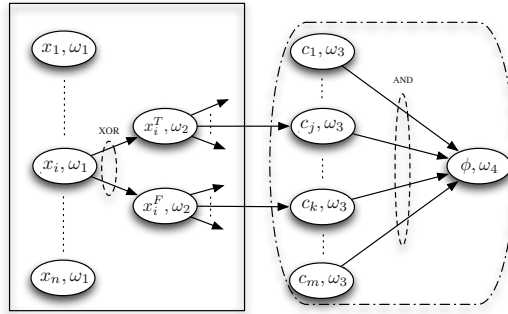


Fig. 6. Hardness results

- For each variable X_i occurring positively (negatively, resp.) in c_j we consider the mapping $\langle \{(x_i^T, \omega_2)\}, \{(c_j, \omega_3)\} \rangle$ ($\langle \{(x_i^F, \omega_2)\}, \{(c_j, \omega_3)\} \rangle$, resp.).
- For each clause c_j we add the mapping $\langle \{(c_j, \omega_3)\}, \{(\phi, \omega_4)\} \rangle$.
- ϑ contains the constraint $\text{AND}(\langle \{(c_1, \omega_3)\}, \{(\phi, \omega_4)\} \rangle, \dots, \langle \{(c_m, \omega_3)\}, \{(\phi, \omega_4)\} \rangle)$.
- For each variable X_i , the constraint $\text{XOR}(\langle \{(x_i, \omega_1)\}, \{(x_i^T, \omega_2)\} \rangle, \langle \{(x_i, \omega_1)\}, \{(x_i^F, \omega_2)\} \rangle)$ is added to ϑ .

The two mappings $\langle \{(x_i, \omega_1)\}, \{(x_i^T, \omega_2)\} \rangle$ and $\langle \{(x_i, \omega_1)\}, \{(x_i^F, \omega_2)\} \rangle$ are meant to encode the selection of a truth value assignment to the variable X_i . Indeed, they are composed using a XOR constraint and at most one of them can be selected in any solution. Moreover, since all the mappings in the set $\bigcup_{j=1}^m \langle \{(c_j, \omega_3)\}, \{(\phi, \omega_4)\} \rangle$ are composed using an AND constraint, all of them must be selected in any solution and, thus, the required entity ϕ can be obtained if and only if all the m clauses can be satisfied. Upon \mathcal{F} we define:

- the Simple Mapping instance: $\text{FM}(\phi) = \langle \mathcal{F}, \bigcup_i \{(x_i, \omega_1)\}, \{(\phi, \omega_4)\} \rangle$;
- the Perfect Mapping instance: $\text{PM}(\phi) = \langle \mathcal{F}, \bigcup_i \{(x_i, \omega_1)\}, \{(\phi, \omega_4)\} \rangle$.

Eventually, one may check that as for MEP it is possible to obtain only entities of the target ontology ω_4 (i.e., the entity ϕ) iff all the various selections encode an assignment that satisfies ϕ . The same reasoning holds for EMEP over $\text{PM}(\phi)$. Thus, ϕ is satisfiable \Leftrightarrow the MEP or EMEP problems over $\text{FM}(\phi)$ and $\text{PM}(\phi)$ has a solution. \diamond

Clearly enough, the hardness results we have derived for the Mapping Existence problems are inherited by the Mapping Selection ones (i.e., OMSP and EOMSP). This is because, determining the optimal solution implies to check the existence of at least one solution.

6 A Constraint Satisfaction Problem Formulation

In this section a CSP formulation is introduced for encoding the existence problems discussed so far. A constraint satisfaction problem (CSP) consists of a set of n variables

$\{x_1, \dots, x_n\}$; a domain D_i of possible values for each variable x_i , $i \in \{1, \dots, n\}$; a set of m constraints $\{C_1, \dots, C_m\}$, where each constraint C_i , $i \in \{1, \dots, m\}$ defines the valid values for the set of variables to which apply [4]. The choice of variables defines the search space whereas the choice of constraints defines how the search space can be reduced so that it can be effectively searched using backtracking search. The advantage in using CSP formulations for our problems is that powerful algorithms that have been designed to solve CSP can be exploited. In more detail, encoding a problem in CSP consists in choosing variables, domains and constraints. The formulations of Mapping Existence (MEP), Perfect Mapping Existence (PMEP) and Exact Perfect Mapping Existence (EPMEP) Problems described in the following rely on binary variables. This way both CSP and SAT solvers can be exploited. Therefore, in the following, the sets used to represent input entities (i.e., \mathcal{C}_{in}), target entities (i.e., \mathcal{C}_{out}), target ontologies (i.e., Ω_{out}), input \mathcal{I} and output \mathcal{O} entities of each mapping will be encoded as sets of binary variables. As an example, the set \mathcal{C}_{in} is represented by the following $|\mathcal{C}| \cdot |\Omega|$ binary variables: $\mathcal{C}_{in}(c, \omega) \in \{0, 1\}$, $\forall c \in \mathcal{C}$ and $\forall \omega \in \Omega$. In particular, the variable $\mathcal{C}_{in}(c, \omega) = 1$ iff the pair $(c, \omega) \in \mathcal{C}_{in}$.

In each of the three problems (MEP, PMEP and EPMEP) the size of any solution (in term of solution sequence) is at most equal to the number of mappings in the *SFN*. Thus, for each mapping, it has to be decided if it belongs to the solution sequence and, if so, to which position of the solution sequence it belongs. Let $|\mathcal{M}| = n$ be the number of mappings in the *SFN*. We introduce a set of binary variables $x_i^j \in \{0, 1\}$, $i, j \in \{1, \dots, n\}$, where $x_i^j = 1$ iff the i -th mapping is placed in the position j of the solution sequence, and $x_i^j = 0$ otherwise. By setting all these variables we can encode the fact that both a set of mappings and the order in which they have to be processed, have been decided. However, many illegal solutions may be obtained and, thus, some constraints have to be introduced:

- *Unary mapping execution constraints*. These constraints force each mapping to be selected at most once, $\sum_j x_i^j \leq 1, \forall i \in \{1, \dots, n\}$.
- *Concurrency constraints*. These constraints force that for each position of the solution sequence at most one mapping is selected, $\sum_i x_i^j \leq 1, \forall j \in \{1, \dots, n\}$.
- *No gap constraints*. These constraints enforce that if a mapping has been selected at position j then a mapping must be selected for all previous $j-1$ positions, $\bigvee_i x_i^j \leq \bigvee_i x_i^k, \forall k \in \{1, \dots, j-1\}$.
- *Update constraints*. These constraints encode the conditions expressed by formulas (1) and (2) in Section 4.2. These constraints must hold for each mapping selected at each position $m \in \{1, \dots, n\}$. For encoding these constraints, the variables $\mathcal{T}_m(c, \omega) \in \{0, 1\}$ and $\mathcal{D}_m(c, \omega) \in \{0, 1\}$, $m \in \{1, \dots, n\}$, $c \in \mathcal{C}$ and $\omega \in \Omega$ are introduced. $\mathcal{T}_m(c, \omega) = 1$ iff the entity c of the ontology ω belongs to the set of entities obtained after processing the mapping selected at position m . $\mathcal{D}_m(c, \omega) = 1$ iff the entity (c, ω) has been used as input in some mappings in the previous $m-1$ steps. Note that $\mathcal{T}_0(c, \omega) = \mathcal{C}_{in}(c, \omega)$ and $\mathcal{D}_0(c, \omega) = 0, \forall c \in \mathcal{C}$ and $\omega \in \Omega$.
 - $\mathcal{T}_m(c, \omega) = (\mathcal{T}_{m-1}(c, \omega) \wedge (\neg(\bigvee_{i=1}^n (x_i^m \wedge \mathcal{I}_i(c, \omega)))) \vee (\bigvee_{i=1}^n (x_i^m \wedge \mathcal{O}_i(c, \omega))))$.
 - $\mathcal{D}_m(c, \omega) = \bigvee_{j=1}^m \bigvee_{i=1}^n (x_i^j \wedge \mathcal{I}_i(c, \omega))$.

- *Enough Input constraints.* These constraints enforce that the entities in input at the mapping selected at position m must be available, $\bigvee_{i=1}^n (x_i^m \wedge \mathcal{I}_i(c, \omega)) \Rightarrow (\mathcal{T}_{m-1}(c, \omega) \vee \mathcal{D}_{m-1}(c, \omega)), \forall c \in C, \forall \omega \in \Omega$.

Additional binary variables and constraints are necessary to take into account the AND and XOR constraints imposed over the mappings. We introduce a new binary variable $a_k \in \{0, 1\}$ for each AND constraint in the *SFN* and also consider the following constraints in the CSP formulation:

- *AND constraints.* These constraints enforce that for each AND-constrained set of mappings all or none of them are selected. If the i -th mapping belongs to the k set of AND-constrained mappings, we add the constraint $a_k = \bigvee_{j=1}^n x_i^j$
- *XOR constraints.* These constraints ensure that for each pair of mappings (or, possibly, sets of AND-constrained mappings) composed using a XOR constraint at most one of them is selected. If the i -th and q -th mappings are XOR-constrained we add the constraint $(\bigvee_{j=1}^n x_i^j = 1) \Rightarrow (\bigvee_{j=1}^n x_q^j = 0)$. Note that, if the XOR constraint involve the k -th set of AND-constrained mappings the term $\bigvee_{j=1}^n x_i^j$ is substituted with a_k .

Finally, the last set of constraints is necessary to ensure the achievement of the requirements at the end of the solution sequence:

- *Requirement constraints.*
 - $\mathcal{T}_n(c, \omega) \Rightarrow \Omega_{out}(\omega)$ for the MEP;
 - $\mathcal{C}_{out}(c, \omega) \Rightarrow \mathcal{T}_n(c, \omega)$ for the PMEP;
 - $\mathcal{C}_{out}(c, \omega) \Leftrightarrow \mathcal{T}_n(c, \omega)$ for the EPMEP.

To conclude, note that the number of variable is quadratic in the number of mappings and linear in the number of entities and ontologies involved in the *SFN*.

7 Conclusions

This paper introduced the notion of *Semantic Flow Network*, upon which two main tasks in mapping discovery, that is, *Free Mapping* and *Perfect Mapping* have been identified. The problems of *mapping existence*, which is concerned to establish if the *Free* or *Perfect* mapping admit a solution, and the *mapping selection* that, on the other hand, aims at finding the optimal solution, have also been introduced. Some complexity results for both the mapping existence and selection problems have been provided. We plan to investigate classes of instances of these problems for which mapping existence and selection become tractable. Also checking the logical consistence of the solutions to the problems we introduced is another interesting line of future research.

References

1. Bernstein, P.A., Green, T.J., Melnik, S., Nash, A.: Implementing Mapping Composition. *VLDB Journal* 17(2), 333–353 (2008)
2. Borgida, A., Serafini, L.: Distributed Description Logics: Assimilating Information from Peer Sources. In: Spaccapietra, S., March, S., Aberer, K. (eds.) *Journal on Data Semantics I*. LNCS, vol. 2800, pp. 153–184. Springer, Heidelberg (2003)

3. David, J., Euzenat, J., Scharffe, F., dos Santos, C.T.: The Alignment API 4.0. *Semantic Web Journal* 2 (2011)
4. Dechter, R.: *Constraint Processing*. Morgan Kaufmann (2003)
5. Euzenat, J.: Algebras of Ontology Alignment Relations. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 387–402. Springer, Heidelberg (2008)
6. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
7. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing Schema Mappings: Second-order Dependencies to the Rescue. *ACM Trans. on Database Systems* 30(4), 994–1055 (2005)
8. Guarino, N., Giaretta, P.: *Ontologies and Knowledge Bases - Towards a Terminological Clarification*. In: *Knowledge Building and Knowledge Sharing*, pp. 25–32. IOS Press, Amsterdam (1995)
9. Jiang, G., Cybenko, G., Hendler, J.A.: Semantic Interoperability and Information Fluidity. *Int. Journal of Cooperative Information Systems* 15(1), 1–22 (2006)
10. Jung, J.J.: Reusing Ontology Mappings for Query Routing in Semantic Peer-to-Peer Environment. *Information Sciences* 180(17), 3248–3257 (2010)
11. Kolaitis, P.G.: Schema Mappings, Data Exchange, and Metadata Management. In: *PODS*, pp. 61–75 (2005)
12. Madhavan, J., Halevy, A.Y.: Composing Mappings among Data Sources. In: *VLDB*, pp. 572–583 (2003)
13. Noy, N.F., Musen, M.A.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: *Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, Austin, Texas (2000)
14. Kutz, O., Normann, I., Mossakowski, T., Walther, D.: Chinese whispers and connected alignments. In: *Proc. of the 5th International Workshop on Ontology Matching, OM 2010* (2010)
15. Papadimitriou, C.H.: *Computational complexity*. Addison-Wesley (1994)
16. Pirrò, G., Talia, D.: LOM: a linguistic ontology matcher based on information retrieval. *J. Information Science* 34(6), 845–860 (2008)
17. Pirrò, G., Ruffolo, M., Talia, D.: SECCO: On Building Semantic Links in Peer-to-Peer Networks. In: Spaccapietra, S. (ed.) *Journal on Data Semantics XII*. LNCS, vol. 5480, pp. 1–36. Springer, Heidelberg (2009)
18. Tordai, A., Ghazvinian, A., van Ossenbruggen, J., Musen, M.A., Noy, N.F.: Lost in translation? empirical analysis of mapping compositions for large ontologies. In: *Proc. of the 9th International Workshop on Ontology Matching, OM 2010* (2010)
19. Zimmermann, A., Euzenat, J.: Three Semantics for Distributed Systems and their Relations with Alignment Composition. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006*. LNCS, vol. 4273, pp. 16–29. Springer, Heidelberg (2006)

Building a Large Scale Knowledge Base from Chinese Wiki Encyclopedia

Zhichun Wang¹, Zhigang Wang¹, Juanzi Li¹, and Jeff Z. Pan²

¹ Department of Computer Science and Technology, Tsinghua University
{zchwang, wzhighang, ljz}@keg.cs.tsinghua.edu.cn

² Department of Computer Science, The University of Aberdeen
Jeff.z.pan@abdn.ac.uk

Abstract. DBpedia has been proved to be a successful structured knowledge base, and large scale Semantic Web data has been built by using DBpedia as the central interlinking-hubs of the Web of Data in English. But in Chinese, due to the heavily imbalance in size (no more than one tenth) between English and Chinese in Wikipedia, there are few Chinese linked data are published and linked to DBpedia, which hinders the structured knowledge sharing both within Chinese resources and cross-lingual resources. This paper aims at building large scale Chinese structured knowledge base from Hudong, which is one of the largest Chinese Wiki Encyclopedia websites. In this paper, an upper-level ontology schema in Chinese is first learned based on the category system and Infobox information in Hudong. Totally, there are 19542 concepts are inferred, which are organized in hierarchy with maximally 20 levels. 2381 properties with domain and range information are learned according to the attributes in the Hudong Infoboxes. Then, 802593 instances are extracted and described using the concepts and properties in the learned ontology. These extracted instances cover a wide range of things, including persons, organizations, places and so on. Among all the instances, 62679 of them are linked to identical instances in DBpedia. Moreover, the paper provides RDF dump or SPARQL to access the established Chinese knowledge base. The general upper-level ontology and wide coverage makes the knowledge base a valuable Chinese semantic resource. It not only can be used in Chinese linked data building, the fundamental work for building multi lingual knowledge base across heterogeneous resources of different languages, but also can largely facilitate many useful applications of large-scale knowledge base such as knowledge question-answering and semantic search.

Keywords: Semantic Web, Linked Data, Ontology, Knowledge base.

1 Introduction

The vision of Semantic Web is to build a "web of data" that enables machines to understand the semantics of information on the Web [1]. In order to achieving the goal of Semantic Web, datasets in various domains have been published and

interlinked on the Web, such as DBLP¹ in the domain of scientific publication, Myspace² in the domain of social networks, Linked MDB³ and Music Brains⁴ in the domain of entertainment. Besides these domain dependent datasets, several large-scale domain independent knowledge bases covering various things have also been proposed, including YAGO [2, 3], DBpedia [4, 5] and Freebase [6]. These knowledge bases typically integrate information from different resources, and provide structured information of various objects. Take DBpedia as an example, it extracts structural information from Wikipedia, provides approximately 1.2 billion triples of information and covers various domains such as geographic information, people, companies, films, music, etc. Because of well-defined ontologies and wide coverage of things, DBpedia and YAGO have the works as the core of linked data [7], and have been used in applications such as music recommendation [8], tag disambiguation [9], information extraction [10, 11].

With various knowledge of different languages are used on the web, multilinguality of semantic web is increasingly evident. Currently, DBpedia provides several versions in non-English languages, including German, French and Japanese etc. However, there is not a Chinese knowledge base with wide coverage in the cloud of linked data. This is mainly because English Wikipedia has 359 thousand articles with inter-language links to Chinese Wikipedia that is only no more than one tenth comparing to 3.64 million English articles in Wikipedia. Also, both DBpedia and YAGO only builds upper-level ontology in English, there is not a Chinese domain independent ontology for the linked data. These problems hinder the structured knowledge sharing both within Chinese resources and cross-lingual resources in Semantic Web.

In this paper, we aim at building a large-scale domain independent Chinese knowledge base on an online Chinese Wiki encyclopedia website, Hudong. As a Chinese encyclopedia, Hudong has much more articles than Chinese Wikipedia; the categories in Hudong are organized in a hierarchy like a tree, which is more suitable for building concept taxonomy. Based on our method, the extracted knowledge base consists of Chinese ontological schema and covers a large number of instances. Specifically, our work makes the following contributions:

- 1) We propose a method to learn an ontology from the category system and Infobox schema in Hudong. Concepts and its hierarchy are extracted from the category system by eliminating inconsistent relations and too specific categories. Properties are extracted from the Infoboxes, and their domains and ranges are properly defined according to their associated concepts. Based on the proposed method, 52404 concepts are extracted and organized in hierarchy with maximally 20 levels are extracted from Hudong. At the same time, 2381 properties with domain and range are learned to describe various relations between different concepts;

¹ <http://dblp.rkbexplorer.com/>

² <http://dbtune.org/myspace/>

³ <http://linkedmdb.org/>

⁴ <http://dbtune.org/musicbrainz/>

- 2) Based on the extracted ontology, 802,593 instances are extracted from Hudong. Three kinds of properties including *General-properties*, *Infobox-properties* and *Person-Relation-properties* are used to describe various attributes of entities and their relationships, resulting in more than 5.2 million RDF triples. And among which, 62679 entities are linked to their identical entities in DBpedia to make our knowledge base linked with others.
- 3) Both RDF dump and SPARQL endpoint are provided to access our knowledge base.

The rest of this paper is organized as follows: Section 2 introduces the Hudong which is the one of the most largest Chinese Wiki encyclopedia websites, Section 3 presents our approach of ontology extraction from Hudong, Section 4 describes how the entities are extracted; Section 5 shows the results of established knowledge base; Section 6 gives the conclusion and future work.

2 Preliminary

This section first gives some related definitions, and then briefly introduces the Hudong encyclopedia.

2.1 Related Definitions

An knowledge base consist of a ontology which model the schema information, and a set of instance defined and described under the ontology constitutes the main information in the knowledge base. We formally introduce some related notions as follows.

Definition 1. An ontology is a formal specification of a shared conceptualization, which provides a vocabulary describing a domain of interest [12]. An ontology can be described as a 4-tuple:

$$O = \{C, P, H^C, H^P\}$$

where C and P are the sets of concepts and properties, respectively. H^C and H^P represents the hierarchical relationships of concepts and properties, respectively.

Definition 2. Let I be a set of instances of concepts in ontology O , the ontology O together with instances I constitute a Knowledge Base $KB = \{O, I\}$.

Definition 3. In an ontology O , properties stating relationships from instances to data values are called datatype properties; properties describing relationships between instances are called object properties.

Definition 4. In an ontology O , the concepts that a property P describes, are called the domain of property P , denote as $dom(P)$; the allowed concepts that the value of an object property P can linked to, are called the range of property P , denote as $rag(P)$.

Definition 5. Given a set of concepts $C = \{C_1, C_2, \dots, C_n\}$, the Minimum General Set (MGS) of C is a set of concepts C^g that satisfies:

- For each concept $C_i \in C$, $C_i \in C^g$ or $\exists C'_i \in C^g, C_i \prec C'_i$ ($A \prec B$ means A is a sub-concept of B);
- For each concept $C_i \in C^g$, $C_i \in C$;
- For each concept $C_i \in C^g$, $\neg \exists C_j \in C \setminus \{C_i\}$ that $C_i \prec C_j$.

The MGS in Definition 5 will be used to derive the domains and ranges of properties in the following Section. Given a set of concept C , Algorithm 1 shows how to transform it to its Minimum General Set.

Algorithm 1. *Minimum General Set Transformation*

Input:

- A concept set $C = \{c_1, c_2, \dots, c_n\}$

Output:

- The *Minimum General Set* C^g of C

Begin:

$C^g \leftarrow \emptyset$;

For each concept $c_i \in C$

If $\neg \exists c_j \in C^g$ that $c_i \prec c_j$
 $C^g \leftarrow C^g \cup \{c_i\}$

EndIf

For each concept $c_j \in C^g$

If $c_j \prec c_i$
 $C^g \leftarrow C^g \setminus \{c_j\}$

EndIf

EndFor

EndFor

Return C^g

End

2.2 Hudong

Our knowledge base is built on Hudong⁵, one of the worlds's largest Chinese encyclopedia website. This section gives a brief introduction to Hudong. Hudong is found in 2005, and it has more than 5 million pages created by 3 million users in May 2011. Basic entries of Hudong are article pages; each describes a specific concept or thing. Typically, each article page contains the following elements:

- Title: Every article in Hudong has a unique title, which denotes the name of the article's subject. We call the title of the article entities in this paper. It is can be words or phrase.

⁵ <http://www.hudong.com/>

- Content: Content of an article is a long text which describes information in various aspect of the article’s subject.
- Links: An article consists of a hypertext document with hyperlinks to other pages within or outside Hudong. The hyperlinks guide readers to the pages that provide related information about the article’s subject.
- Infobox: Infobox offers structured information about the article’s subject in table format. It summarizes the key aspects of the article’s subject by attribute-value pairs.
- Category: An article may have category tags that reflect the topic of the article’s subject. One article may have several or none category tags.

There is a classification tree in Hudong to organize its articles. Nodes in the classification tree are categories, and articles are associated with their corresponding categories. There are 12 upper categories under the root of classification tree, including 社会 (Social), 地理 (Geography), 科学 (Science), 人物 (Person), 文化 (Culture), 经济 (Economics), 艺术 (Art), 自然 (Nature), 技术 (Technology), 历史 (History), 体育 (Sport), 生活 (Life). We define the 13th categories of 组织 (Organization) in our defined categories which play an important role in knowledge base.



Fig. 1. Classification Tree in Hudong

3 Ontology Extraction

We first build an upper level ontology to model the schema information of the extracted knowledge base. This section presents our approach to automatically build the ontology from the category system and Infobox templates in Hudong.

3.1 Concept Extraction

A concept in ontology defines a group of instances that belong to the same type and share some common properties. Concepts can be organized in a hierarchy by specifying the *subclass-of* relation between them. The concepts and their hierarchy comprise the backbone of the ontology, which benefit the sharing and querying information of the extracted entities.

Here, we explore hudong's category system and transformed it to a taxonomy of concepts. Hudong's category system uses a classification tree to organize its articles. Articles describing the same type of things are grouped into one category, and categories have sub-categories and super-categories. For each category, there is a page in Hudong lists its sub-categories, super-categories, and articles belong to it. The categories' names and their hierarchical relations are built by collaborate editions of large number of users. In general, most hierarchical category relationships defined in the classification tree of Hudong is consistent and high qualified.

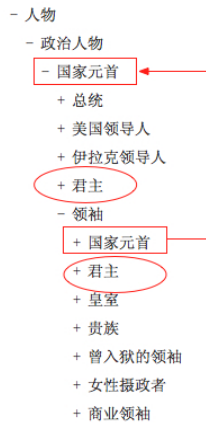


Fig. 2. A snap of the classification tree to illustrate the inconsistency in Hudong category system

We summarize three problems with Hudong categories for defining a concept hierarchy. First, there are some inconsistent sub-class links in the tree; some concepts' sub-classes may also be the super-class of it, or be the brother of its super-classes. Such as shown in Fig 2, the sub-categories of 国家元首 (Head of State) contains a node 国家元首 (Head of State), which causes a circle in the tree. Second, one

category may have several super-categories. Such as shown in Fig 2, the category 君主 (Sovereign) has two sup-categories, 国家元首 (Head of State) and 领袖 (Leader). Third, some categories are too specific that only contains one or two articles, these over specific categories cannot represent a group of instances, therefore is not suitable to be extracted as concepts. In order to build a concept hierarchy, we first use the following methods to refine the category system of Hudong:

- (1) Delete the inconsistent sub-category relations. Enumerate all sub-category links in the classification tree, delete the links from a category on lower level to categories on higher level. By this step, the circles in the classification tree are eliminated without destroying other category relations.
- (2) Delete multiple super-categories, keep the super-category closest to the root category. In this way, only the general definitions of categories are kept.
- (3) Delete specific categories that contain less than two entities.

After refining the category system of Hudong, we define concepts and concepts' hierarchy based on the refined category system. For each category, we define a concept and assign a unique URI to it. The URI of a concept is created by concatenating the namespace prefix *http://CKB.org/ontology/* and the name of the category. The hierarchy of concepts is extracted from the sub-category links in Hudong. If a concept's corresponding category has sub-categories, then concepts corresponding to these sub-categories are specified as its sub-concepts. All the defined concepts and hierarchical relations are recorded using the OWL⁶ language. Fig 3 is a snap of our extracted concept hierarchy, all these concepts belongs to the 人物 (Person) concept.



Fig. 3. A snap of the concept hierarchy extracted from Hudong

⁶ <http://www.w3.org/TR/owl-features/>

3.2 Property Extraction

Properties are used to describe the relationships between instances or from instances to data values. Properties in this paper are divided into two types: datatype properties, relations between instances of classes and RDF literals and XML Schema datatypes; object properties, relations between instances of two classes. We define three groups of properties for Hudong entities: *general-properties*, *Infobox-properties*, and *person-relation-properties*.

(1) *General-properties*

The *general-properties* include label, abstract and url, they are all datatype properties. These properties describe basic information of instances. The label property specifies the name of an instance; the abstract property represents the first paragraph of the text in the instance's Hudong page; the url property gives the url of the Hudong page of an instance.

(2) *Infobox-properties*

Infobox-properties are defined based on the attributes in the Infobox, such as 姓名 (name), 年龄 (age), 籍贯 (native place) in a person's Infobox. All the attributes are defined as properties with a unique URI. Here, we concatenate the namespace prefix *http://CKB.org/ontology/* and the attribute's name as the URI of the defined property. In order to determine the type of properties, i.e. object and datatype, the values of Infobox attributes need to be processed first. If the values of an attribute are plain texts, then this attribute can be defined as a datatype property. For example, the attribute 姓名 (name) can be defined as a datatype property. If the values of an attribute contain links to other entities' pages, then this attribute is an object property. For example, the attribute 校长 (president) of a university is usually a link to a person; therefore the attribute president is defined as a object property, its range is concept 人物 (person).

For each defined property, we also specify its domain and range. For the three general properties, their domain is the most general concept "*Thing*", and their range is defined as "*xsd:string*"⁷. The domains and ranges of Infobox properties are determined by as follows.

a. *Domain*

For each *Infobox-property* P , we enumerate all the wiki pages $W_p = \{w_1, w_2, \dots, w_k\}$ that it appears; record the category tags $T_p = \{t_1, t_2, \dots, t_m\}$ in the wiki pages W_p . Let $D_p = \{C_1, C_2, \dots, C_m\}$ be the set of defined concepts corresponding to categories $T_p = \{t_1, t_2, \dots, t_m\}$, the MGS of D_p is defined as $dom(P)$.

b. *Range*

For all the datatype properties, their ranges are defined as "*xsd:string*."

For each object property P , enumerate all the wiki pages $W_p = \{w_1, w_2, \dots, w_k\}$ that it appears; record all the wiki pages $W_{p_i} = \{w'_1, w'_2, \dots, w'_m\}$ that the values of the

property link to; enumerate pages in W_{p_i} and record the category tags $T_p = \{t_1, t_2, \dots, t_n\}$ in these wiki pages. Let $R_p = \{C_1, C_2, \dots, C_n\}$ be the set of defined concepts corresponding to categories $T_p = \{t_1, t_2, \dots, t_n\}$, the MGS of R_p is defined as $rag(P)$.

(3) *Person relation properties*

In most Hudong pages belong to person category, there is usually a person relation graph describing the relations between the person and other persons. For example, Fig. 4 is an example of person relation of Yao Ming (姚明). The graph shows other persons that related to 姚明 (Yao Ming), including his father, coach, daughter and so on. We extract these relations between persons and define object properties from them. Because all these relations are between persons, the domains and ranges of person relation properties are all set as 人物 (person) concept.

4 Instance Extraction

4.1 Extract Instances and Descriptions

After the ontology being defined, entities in Hudong are extracted as the instances in the ontology. A unique URI is assigned to each instance, which its namespace prefix *http://CKB.org/ontology/* is connected with the instance's name. Concept types are assigned to instances according to their category tags in Hudong. There are three groups of properties to describe information of instances. *General-properties* including title, abstract and url are extracted for every instance. *Infobox-properties* are extracted if there is an Infobox in the instance's Hudong page. For instances belonging to 人物 (person) concept, if there are person relation graphs in their pages, *person-relation* properties will be used to describe the relationships between the instance and other instances.

When extracting the values of object properties from the Infoboxes, we have to handle the problem of missing links. A lot of properties' values should be supposed to have links to other entities, but sometimes they only have instance' name without links. For example, the president of Tsinghua University is “顾秉林” (Binglin Gu), the text “顾秉林” (Binglin Gu)” is not linked to the page of instance “顾秉林” (Binglin Gu)”. Therefore, we have to find these missing links so that we can use object properties to establish RDF links between them. Here we use the method of name matching to add the missing links. The value of object property is matched with the names of all the entities. If there is an exactly matched name with the property value, then the property value is replaced with the link to the matched instance.



Fig. 4. Person relation graph of 姚明(Yao Ming)

4.2 Link Entities to DBpedia Entities

In order to make our knowledge base linked with other linked data, we also create the *owl:sameAs* links with DBpedia. Identical instances' URIs are found by the following method:

- (1) Given an instance e extracted from Hudong, find the entity e' in Chinese Wikipedia with the same title.
- (2) Find whether there is an inter-language links between e' and an entity e'' in English Wikipedia; if e'' is exist, get its url.
- (3) Search the DBpedia URI of e'' by looking for the url.
- (4) Declare $URI(e) owl:sameAs URL(e'')$.

5 Results

5.1 Dataset

We wrote a web crawler that starts from the root of the classification tree in Hudong, and downloads all the articles attached to the nodes in the classification tree. Finally, we are able to download 687 thousand articles. Although the number of extracted articles is relative small comparing to the total number of articles in Hudong, these

downloaded articles have high quality than the rest of articles. These 687 thousand articles usually have rich information including Infoboxes, categories, etc. Table 1 shows the number of articles in each upper category. It should be explained that each Hudong article may appear in multiple upper categories, therefore the total number of pages of 12 categories is much larger than 687 thousand.

5.2 Extracted Knowledge Base

The extracted ontology contains 19542 concepts, 2079 object properties, 302 data type properties. There are 13 upper level concepts in the ontology corresponding to the 13 categories in Hudong, including 社会 (Social), 地理 (Geography), 科学 (Science), 人物 (Person), 文化 (Culture), 组织 (Organization), 经济 (Economics), 艺术 (Art), 自然 (Nature), 技术 (Technology), 历史 (History), 体育 (Sport), 生活 (Life). As we noticed, there is not a upper level category 组织 (Organization) in Hudong category system. The categories belong to organizations appear in all the other upper level categories, such as 经济组织 (Economic Organization) category belongs to 经济 (Economics), 科研机构 (Scientific Organization) belongs to 科学 (Science), etc. Because organization is an important concept, we manually aggregate all the related categories and build “Organization” concept in our ontology. Table 2 shows the number of concepts, associated properties and hierarchy levels for each upper level concept.

Table 1. Number of articles in Hudong’s upper categories

Category	#Hudong articles	Percentage
社会 (Social)	538576	15.45%
地理 (Geography)	520869	14.94%
科学 (Science)	471083	13.52%
人物 (Person)	111899	3.21%
文化 (Culture)	292680	8.40%
生活 (Life)	314047	9.01%
经济 (Economics)	211229	6.06%
艺术 (Art)	261794	7.51%
自然 (Nature)	531240	15.24%
技术 (Technology)	143537	4.12%
历史 (History)	54658	1.57%
体育 (Sport)	33657	0.97%
Total	3485269	100%

Table 2. Ontology information

Concept	#Concepts	#Related properties	#Hierarchy Levels
社会 (Social)	13515	1897	15
地理 (Geography)	11468	1482	18
科学 (Science)	5044	964	19
人物 (Person)	4345	2177	9
生活 (Life)	3379	895	10
文化 (Culture)	1947	963	10
组织 (Organization)	1845	626	10
经济 (Economics)	2346	594	10
艺术 (Art)	1536	816	10
自然 (Nature)	7035	481	17
技术 (Technology)	776	446	11
历史 (History)	1826	672	10
体育 (Sport)	694	588	8
文化 (Culture)	1947	963	10

Based on the extracted ontology, 802593 instances are defined. These instances are described by various properties resulting in 5237520 RDF triples. Table 3 shows the number of instances and RDF triples for each upper level concept.

Our knowledge base is recorded in an RDF file, and we also provide a SPARQL endpoint for querying the knowledge base. Applications can send queries by the SPARQL protocol to endpoint to get instances' structured information. Fig 5 shows the SPARQL query interface of our knowledge base. There is a sample query as follows:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ckb: <http://cbk.org#>

SELECT ?people ?property ?value
WHERE
{ ?people rdfs:label "姚明".
  ?people ?property ?value }
```

This query looks up information about a person 姚明 (Yao Ming), a famous NBA Chinese player. After submitting this query, 38 triples are returned from the knowledge base. Table 4 shows part of the query results, including Yao Ming's birth data, English name, height, etc.

Both the RDF file and SPARQL endpoint can be assessed in our project's homepage: <http://keg.cs.tsinghua.edu.cn/project/ChineseKB/>.

Table 3. Instances Information

	#Instances	#RDF tripples
社会 (Social)	326774	2447922
地理 (Geography)	311952	1999392
科学 (Science)	236187	1589840
人物 (Person)	144254	1153841
生活 (Life)	159252	1088034
文化 (Culture)	120965	879674
组织 (Organization)	107103	602378
经济 (Economics)	99927	637539
艺术 (Art)	98219	726341
自然 (Nature)	94672	1043903
技术 (Technology)	51822	294569
历史 (History)	36979	271885
体育 (Sport)	18701	177989

SPARQL Endpoint of Knowledge Base

Currently, the endpoint provides queries of the whole ontology and instances belonging to "人物" concept.

SELECT - get variables (apply XSLT stylesheet)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ckb: <http://cbk.org#>
```

```
SELECT ?people ?property ?value
WHERE
{ ?people rdfs:label "姚明".
  ?people ?property ?value }
```

Output XML: with XSLT style sheet (leave blank for none):

or JSON output:

or text output:

or CSV output:

or TSV output:

Force the accept header to text/plain regardless

Fig. 5. The SPARQL query interface of our knowledge base

Table 4. Sample query results from the SPAQRL endpoint

people	property	value
<http://ckb.org/ontology#姚明>	<http://www.w3.org/2000/01/rdf-schema#label>	"姚明"
<http://ckb.org/ontology#姚明>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://ckb.org/ontology#慈善家>
<http://ckb.org/ontology#姚明>	<http://ckb.org/ontology#出生年月>	"1980年9月12日"
<http://ckb.org/ontology#姚明>	<http://ckb.org/ontology#英文名>	"Yao Ming"
<http://ckb.org/ontology#姚明>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://ckb.org/ontology#球类运动员>
<http://ckb.org/ontology#姚明>	<http://ckb.org/ontology#身高>	"226 厘米"
<http://ckb.org/ontology#姚明>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://ckb.org/ontology#上海人>
<http://ckb.org/ontology#姚明>	<http://ckb.org/ontology#身材>	"140 公斤"
<http://ckb.org/ontology#姚明>	<http://ckb.org/ontology#别名>	"小巨人、移动长城"
<http://ckb.org/ontology#姚明>	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>	<http://ckb.org/ontology#男演员>
<http://ckb.org/ontology#姚明>	<http://ckb.org/ontology#重要事件>	"2009.7, 收购上海篮球队, 成为新老板 2009 年带领火箭队进入季后赛第二轮 1998 年入选中国篮球明星队"

6 Related Work

In this section, we review some work related to our paper.

YAGO [2] is a large ontology built based on Wikipedia⁷ and WordNet [13]. It extracts more than 1.7 million entities and 14 relationships from Wikipedia. The category system and the redirect pages are used to establish a hierarchy of concepts. In order to improve the quality of the concepts' hierarchy, YAGO links leaf categories of Wikipedia into the WordNet hierarchy. Different from our work, YAGO does not extract various properties in the Wikipedia's Infoboxes, instead, we extract 2079 properties to describe the characteristics of the concepts in this paper.

DBpedia [4] is a knowledge base which extracts structured information from Wikipedia and to make this information available on the Web. DBpedia extracts entities from Wikipedia and describes entities by a set of general properties and a set of Infobox-specific properties. The extracted entities are also mapped into four classification schemata, including DBpedia ontology⁸, SKOS⁹, YAGO [2] and UMBEL¹⁰. In our paper, we propose a framework to extract schema ontology and entities information according to the features of Chinese wiki Encyclopedia Hudong, and also generate a Chinese structured knowledge base, and 62679 of them are linked to DBpedia. These links provide the knowledge of English-Chinese languages that can be used in the application of cross-lingual knowledge base.

⁷ <http://www.wikipedia.org/>

⁸ <http://wiki.dbpedia.org/Ontology>

⁹ <http://www.w3.org/2004/02/skos/>

¹⁰ <http://www.umbel.org/>

Freebase [6] is an open repository of structured data of almost 22 million entities. Users of Freebase can edit the data in a similar way as they edit Wikipedia articles. Free-base extracts knowledge from Wikipedia as initial content for their database, which is then edited by Freebase users. In Chinese, currently there is no such kind of the information.

Ponzetto et.al [14] proposed an approach for deriving a large-scale taxonomy from Wikipeda. They took the category system in Wikipedia as a conceptual network, and created subsumption hierarchy of concepts. In order to determine the *isa* relation between concepts, they used methods based on the connectivity of the network and on applying lexico-syntactic patterns to Wikipedia articles. They mainly focused on building the subsumption relations between concepts and did not include the instances and their Infoboxes' information in the taxonomy.

Melo et.al [15] explored the multilingual nature of Wikipedia, and built a large multilingual entity taxonomy MENTA, which describes 5.4 million entities in various languages. They integrated entities from all editions of Wikipedia and WordNet to a single coherent taxonomic class hierarchy. Categories are extracted as candidates of classes; categories denoting genuine classes and topic labels are distinguished by the singular/plural heuristic proposed for YAGO [2]. Only categories denoting genuine classes are defined as classes. The *subclass* relations between classes are established by making use of parent categories, category-WordNet subclass relationships and WordNet Hyponymy. Instances are extracted based on the Infoboxes and categories in articles.

To summarize the related work, in this paper we propose a framework to extract schema ontology of knowledge base and generate the structured knowledge base from one of the largest wiki Encyclopedia website-Hudong. Currently, there is no RDF DBpedia like knowledge base in Chinese and few Chinese data sets are linked to DBpedia. Besides, we provide links to DBpedia, which lay the foundation for cross lingual structured knowledge base sharing by integrating structured knowledge base across existing wiki Encyclopedia websites of different languages.

7 Conclusion

This paper presents a Chinese knowledge base built from a Chinese Wiki websites, Hudong. An upper level Chinese ontology is first built based on the category system and Infobox schema of Hudong. Then more than 800 thousand entities in various domains are extracted and classified according to the defined ontology. Structured information of entities is described by the defined properties. As the development of semantic techniques and applications, our knowledge base can be used as a useful Chinese semantic resource. Currently, both RDF dump and SPARQL endpoints are provided to access our knowledge base.

As our future work, we will concentrate on improving the quality of the Chinese structured knowledge base including the refinement of the schema ontology and the process of the entities learning. We want to build links from other data sets such as in news domain and academic domain to this structured knowledge base to build Chinese linked data. Also, linking Chinese structured knowledge with DBpedia or other knowledge bases of different languages could fulfill structured knowledge sharing across heterogeneous knowledge bases of different languages.

Acknowledgement. The work is supported by the National Natural Science Foundation of China (No. 6 61035004, 60973102), the National Basic Research Program of China (973 Program) (No. 2007CB310803), the China Postdoctoral Science Foundation (No. 20110490390), it is also supported by THU-NUS Next research center.

References

- [1] Berners-Lee, T.: Semantic Web Road map (1998), <http://www.w3.org/DesignIssues/Semantic.html>
- [2] Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semantics: Science. Services and Agents on the World Wide Web* 6(3), 203–217 (2008)
- [3] Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: *Proceedings of the 16th International Conference on World Wide Web 2007*, pp. 697–706. ACM, Banff (2007)
- [4] Bizer, C., et al.: 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)
- [5] Auer, S., et al.: DBpedia: A Nucleus for a Web of Open Data The Semantic Web. In: Aberer, K., et al. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
- [6] Bollacker, K., et al.: Freebase: a collaboratively created graph database for structuring human knowledge. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data 2008*, pp. 1247–1250. ACM, Vancouver (2008)
- [7] Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - the story so far. *International Journal on Semantic Web and Information Systems* 5(3) (2009)
- [8] Passant, A.: dbrec - Music Recommendations Using DBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part II*. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
- [9] García-Silva, et al.: Preliminary Results in Tag Disambiguation using DBpedia. In: *First International Workshop Collective Knowledge Capturing and Representation CKCaR 2009*, Redondo Beach, California, USA (2009)
- [10] Wu, F., Weld, D.S.: Automatically refining the wikipedia infobox ontology. In: *Proceeding of the 17th International Conference on World Wide Web 2008*, pp. 635–644. ACM, Beijing (2008)
- [11] Kasneci, G., et al.: The YAGO-NAGA approach to knowledge discovery. *SIGMOD Record* (2008)
- [12] Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
- [13] Fellbaum, C.: *WordNet: An Electronic Lexical Database*. In: Fellbaum, C. (ed.) *WordNet: An Electronic Lexical Database*. MIT Press (1998)
- [14] Ponzetto, S.P., Strube, M.: Deriving a large scale taxonomy from Wikipedia. In: *Proceedings of the 22nd National Conference on Artificial Intelligence*, vol. 2, pp. 1440–1445. AAAI Press, Vancouver (2007)
- [15] Melo, G., Weikum, G.: MENTA: inducing multilingual taxonomies from wikipedia. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 1099–1108. ACM, Toronto (2010)

Dynamic *Is-a* Hierarchy Generation System Based on User's Viewpoint

Kouji Kozaki, Keisuke Hihara, and Riiciro Mizoguchi

The Institute of Scientific and Industrial Research, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
{kozaki, hihara, miz}@ei.sanken.osaka-u.ac.jp

Abstract. In ontological theories, *is-a* hierarchy must represent the essential property of things and hence should be single-inheritance, since the essential property of things cannot exist in multiple. However, we cannot avoid multi-perspective issues when we build an ontology because the user often want to understand things from their own viewpoints. Especially, in the Semantic Web, the variety of users causes the variety of viewpoints to capture target domains. In order to tackle this multi-perspective issue, we adopt an approach of dynamically generating *is-a* hierarchies according to the viewpoints of users from an ontology using single-inheritance. This article discusses a framework for dynamic *is-a* hierarchy generation with ontological consideration on *is-a* hierarchies generated by it. Then, the author shows its implementation as a new function of Hozo and its applications to a medical ontology for dynamically generation of *is-a* hierarchies of disease. Through the function, users can understand an ontology from a variety of viewpoints. As a result, it could contribute to comprehensive understanding of the ontology and its target world.

Keywords: ontology, dynamic *is-a* hierarchy generation, viewpoint, disease ontology.

1 Introduction

Ontologies are designed to provide systematized knowledge and machine readable vocabularies of domains for Semantic Web applications. The competences of semantic technologies strongly depend on the ontology which they use. Ontology is defined as “An explicit specification of conceptualization” [1], and it clearly represents how the target world is captured by people and systems.

Semantics of concepts (classes) are defined clearly through the description of their relationships between other concepts in an ontology. In particular, the most important relationship is an *is-a* (sub-class-of) relationship which represents a relation between a generalized concept and a specialized concept. Class hierarchies according to *is-a* relationships are called *is-a* hierarchies, and they form the foundation of ontologies. That is, *is-a* hierarchies in an ontology reflect how the ontology captures the essential conceptual structure of the target world.

Therefore, in ontological theories, an *is-a* hierarchy should be single-inheritance because the essential property of things cannot exist in multiple. Imagine that objects,

processes, attributes, all of them have their own unique and essential properties. The use of multiple-inheritance for organizing things necessarily blurs what the essential property of things is. This observation is strongly supported by the fact that both of the well-known upper ontologies: DOLCE and BFO use single-inheritance hierarchies.

Nicola Gunarino criticizes the careless usage of *is-a* relationships without enough ontological consideration as *is-a* overloading [2] and propose an ontology development methodology, called OntoClean, which defines concepts based on meta-properties such as rigidity and anti-rigidity. DOLCE is developed based on the OntoClean methodology using single-inheritance *is-a* hierarchy. BFO is the upper ontology used by the OBO Foundry¹ which aims to create a suite of orthogonal interoperable reference ontologies in the biomedical domain. BFO also uses single-inheritance hierarchy, and it is recommended in the guideline of OBO Foundry to avoid careless usage of multiple-inheritance.

However, we cannot avoid multi-perspective issues when we build an ontology across multiple domains. It is because domain experts often want to understand the target world from their own domain-specific viewpoints. In many cases, their interests are different even if they are experts in the same domain. In some domains, there are many ways of categorization of the same kinds of concepts and different taxonomies are used depending on the purpose and situation.

For example, in the medical domain, a disease is interpreted from various viewpoints. Consider diabetes as an example. Clinician may pay attention to the body parts with the abnormalities and classify diabetes as diseases which have abnormality in blood. On the other hand, certain specialists may pay attention to the main pathological condition and may classify diabetes as an abnormality in metabolism, and other specialists may classify diabetes as a lifestyle related disease. Staffs administering medical care implicitly understand which *is-a* hierarchy should be used for disease interpretation in correlation with their respective interpretations. This suggests that one *is-a* hierarchy of diseases cannot cope with such a diversity of viewpoints, since a single-inheritance hierarchy necessarily represents one viewpoint.

Many efforts are under taken to solve these multi-perspective issues. The OBO Foundry proposes a guideline for ontology development stating that we should build only one ontology in each domain [3]. This is an effort to exclude the multi-perspective nature of domains from ontologies. Ontology mapping is used as an approach to acceptance of multiple ontologies based on the different perspectives in a domain. It aims to make clear the relationships between different ontologies. Someone may consider that multiple-inheritance is an easy way to solve these multi-perspective issues. Because multiple-inheritance causes some ontological problems as mentioned above, our ontology development tool, named Hozo[4]², allows the user to use a multiple-inheritance only when he/she represents clearly from which upper-concepts the essential property is inherited³. However, if we define every possible *is-a* hierarchy using multiple-inheritances, they would be very verbose and the user's viewpoints would become implicit.

¹ <http://www.obofoundry.org/>

² <http://www.hozo.jp>

³ It is represented by two kinds of *is-a* relationships: (essential) *is-a* and (non-essential) *IS-A*.

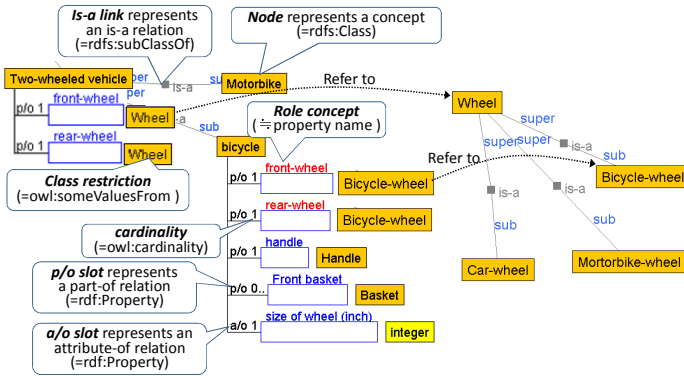


Fig. 1. An example of ontology defined using Hozo

In order to tackle these multi-perspective issues, the authors take an approach based on ontological viewpoint management. It is dynamically generation of *is-a* hierarchies according to the viewpoint of users from an ontology using single-inheritance. The main strategy is composed of: (1) fixing the conceptual structure of an ontology using single-inheritance based on ontological theories and (2) reorganizing some conceptual structures from the ontology on the fly as visualizations to cope with various viewpoints. Based on this strategy, the authors consider a framework for dynamic *is-a* hierarchy generation according to the interests of the user and implement the framework as an extended function of the ontology development tool Hozo. In this article, we discuss the framework for dynamic *is-a* hierarchy generation and its application to a medical ontology. It would solve the conflicting requirements of multi-perspective and single-inheritance in a good ontology, and it could contribute to deep understanding of the ontology.

The rest of this paper is organized as follows: In section 2, we introduce dynamic *is-a* hierarchy generation according to viewpoints. Next, we consider ontological characteristics of the generated *is-a* hierarchies in section 3. In section 4, we discuss implementation of the framework as an extended function of Hozo. In Section 5, we show its application to a medical ontology for dynamic *is-a* hierarchy generation of disease. In section 5, we discuss related work. Finally, we present concluding remarks with future work.

2 Dynamic *Is-a* Hierarchy Generation According to Viewpoints

2.1 Ontology Representation in Hozo

We implement the dynamic *is-a* hierarchy generation system as an additional function of Hozo [4]. Fig.1 shows an example of ontology defined using Hozo. Ontologies are represented by nodes, slots and links. The nodes represent concepts (classes), *is-a* links represent *is-a* (subclass-of) relations, and slots represents *part-of* (denoted by “p/o”) or *attribute-of* (denoted by “a/o”) relations. A slot consists of its kind (“p/o” or “a/o”), *role concept*, *class restriction*, *cardinality*. Roughly speaking, a slot corresponds to *property* in OWL and its role name represents name of the property. Its class restriction and cardinality correspond to owl: someValuesFrom and owl:cardinality respectively. Their restrictions refer to other concepts which are

defined elsewhere. However, semantics of Hozo's ontology includes some concepts related to role which are not supported in OWL because it is designed based on an ontological theory of role [5]. While we have designed three levels of role representation model in OWL to capture the semantics level-wise [6], we use the simplest model described above in this paper because it is almost compatible with OWL. That is, the proposed method for dynamic *is-a* generation is also applicable to not only ontologies in Hozo's format but also ontologies in OWL.

In the target ontologies, concepts (classes) are defined by several slots which represent properties and restrictions for them. These definitions are inherited from super-concepts (super-classes) to their sub-concepts (sub-classes) along with *is-a* links. Furthermore, in some sub-concepts, some inherited definitions are specialized according to *is-a* hierarchies of concepts which are referred to by their restrictions. For example, *bicycle* in Fig.1 inherit *front-wheel* from *Two-wheeled vehicle* and its class-restriction could be specialized from *Wheel* to *Bicycle-wheel*. This research focuses on these characteristics of *is-a* hierarchies and considers an approach to reorganize *is-a* hierarchies of concepts based on *is-a* hierarchies of concepts referred to by their definitions.

2.2 Dynamic *Is-a* Hierarchy Generation through Transcription of a Hierarchical Structure

Fig.2 outlines a framework for dynamic *is-a* generation. It generates *is-a* hierarchies by reorganizing the conceptual structures of the target concept selected by a user according to the user's viewpoint. The viewpoint is represented by an *aspect* and a *base hierarchy*. By aspect, we mean something which the user is interested in and selects from the definition of the target concept to generate an *is-a* hierarchy. By base hierarchy, we mean a conceptual structure of concepts which are referred to by the definition selected as the aspect. Because sub-concepts of the target concept could be defined by specializing their inherited definitions according to the base hierarchy, we could reorganize the *is-a* hierarchy of the target concepts according to the following steps:

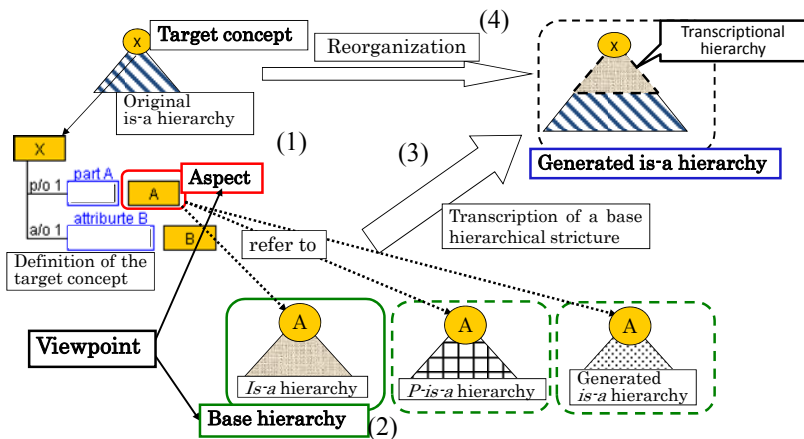


Fig. 2. A framework for dynamic *is-a* generation

Step 1: Selection of an Aspect

The user selects something as an aspect from the definition of the target concept for dynamic *is-a* hierarchy generation (see. Fig.2-(1)). Because any concept is defined in terms of slots each of which consists of a role-concept, a role-holder [5] and a class-restriction, he/she can select one of them as an aspect. In this paper, we consider only a case where the user selects a class restriction as an aspect for simplicity.

Step 2: Selection of a Base Hierarchy

The user selects a base hierarchy from hierarchies of concepts which the aspect is referring to (see. Fig.2-(2)). In Hozo, three kinds of conceptual hierarchies could be the base hierarchy as follows: the *is-a* hierarchy of concepts referred to by the aspect, the *p-is-a* hierarchy which is generated by the system according to part-whole relationships of the concepts referred to and dynamically generated *is-a* hierarchies using the proposed method. A *p-is-a* hierarchy is obtained by abstracting parts from a part-of hierarchy [7]. The detail of the *p-is-a* hierarchy is discussed in section 2.3.2.

Step 3: Transcription of a Hierarchical Structure

The system defines new sub-concepts of the target concept by specializing the definition of it according to the class restriction selected as an aspect and base hierarchy (see. Fig.2-(3)). Then, their concept names are automatically determined by the system using a template such as “<the target concept name> with <the specialized aspect> as <the role name of the aspect>”. As a result, an *is-a* hierarchy which has the same conceptual structure with the base hierarchy is generated. We call the generated hierarchy a *transcriptional hierarchy* and the operations to generate it a *transcription of a hierarchical structure*.

The scope of a transcription of the base hierarchy could be managed by specifying the number of the target layers rather than to use all concepts of the base hierarchy for transcription.

Step 4: Reorganization of *is-a* Hierarchy Based on a Transcriptional Hierarchy

The system reorganizes the *is-a* hierarchy by comparing the original *is-a* hierarchy and the transcriptional hierarchy generated in step 3. The system compares the sub-concepts of the target concept (we call them *existing sub-concepts*) with the concepts on the transcriptional hierarchy (we call them *generated sub-concepts*) according to the aspect and the base hierarchy. When an existing sub-concept’s definition is subsumed by the definition of a generated sub-concept, the existing sub-concept is classified into a sub-concept of the generated sub-concept. If an existing concept is classified into sub-concepts of multiple generated sub-concepts, the existing concept is classified into the lowest sub-concepts. As a result, all existing concepts are classified into sub-concepts of the generated concepts in the transcriptional hierarchy according to the aspect and the base hierarchy⁴.

Through the above four steps, the system can dynamically generate *is-a* hierarchies by reorganizing existing sub-concepts according to the transcriptional hierarchies of base hierarchies.

⁴ The result of reorganization corresponds to the result of classification using DL-reasoner while it is implemented by procedural ways in Hozo.

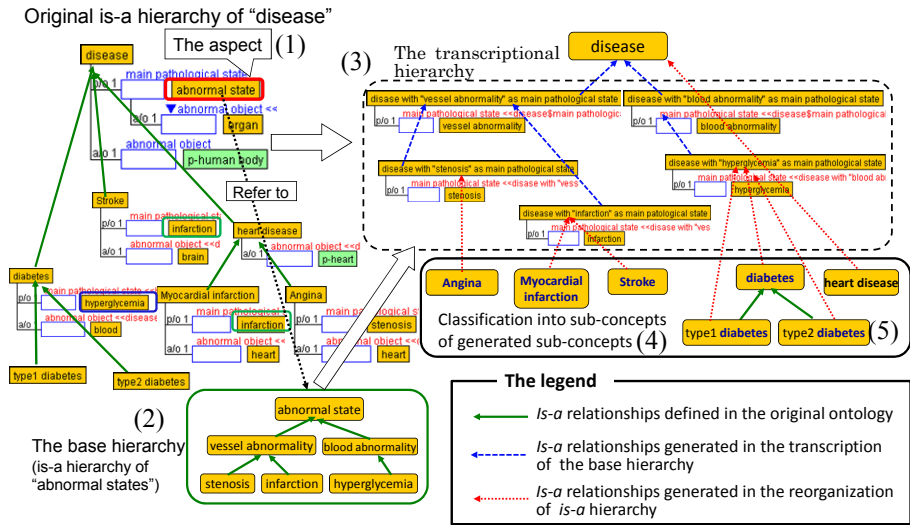


Fig. 3. An example of dynamic *is-a* generation of disease in the case that *is-a* hierarchy of abnormal state is selected as the base hierarchy

Although DL-reasoners can classify classes (concepts) automatically by reasoning, the result of classification is only an *is-a* hierarchy which is determined uniquely according to the definitions of the classes. Therefore, it is different from our dynamic reorganization according to the users' view points. DL-reasoners can generate a different *is-a* hierarchy only when class definitions in the ontology have changed.

2.3 Examples of Dynamic *Is-a* Hierarchy Generation

§ 1 In the Case of that an *Is-a* Hierarchy is Selected as a Base Hierarchy

As an example, we consider a dynamic *is-a* hierarchy generation of diseases which is defined in terms of several slots such as "main pathological state", "abnormal object" and so on (see. Fig.3). Here, we suppose the user selects the class-restriction of "main pathological state" as an aspect (Fig.3-(1)) and the *is-a* hierarchy of "abnormal state" as a base hierarchy (Fig.3-(2)).

First, sub-concepts of "disease" such as "disease with vessel abnormality as main pathological state" and "disease with blood abnormality as main pathological state" are dynamically generated by specializing the definition of "disease" according to the class restriction selected as the aspect and the base hierarchy. After repetitions of generations of sub-concepts, the transcriptional hierarchy of "disease" is obtained (Fig.3-(3)). Then, existing sub-concepts of "disease", such as "myocardial infarction" and "angina pectoris" are classified into sub-concepts of the generated sub-concepts on the transcriptional hierarchy through comparisons between definitions of them (Fig.3-(4)). When more than one existing sub-concepts are classified into the same generated sub-concept, they could be organized according to original *is-a* relationships between them. In the case shown in Fig.3-(5), because *is-a* relationships

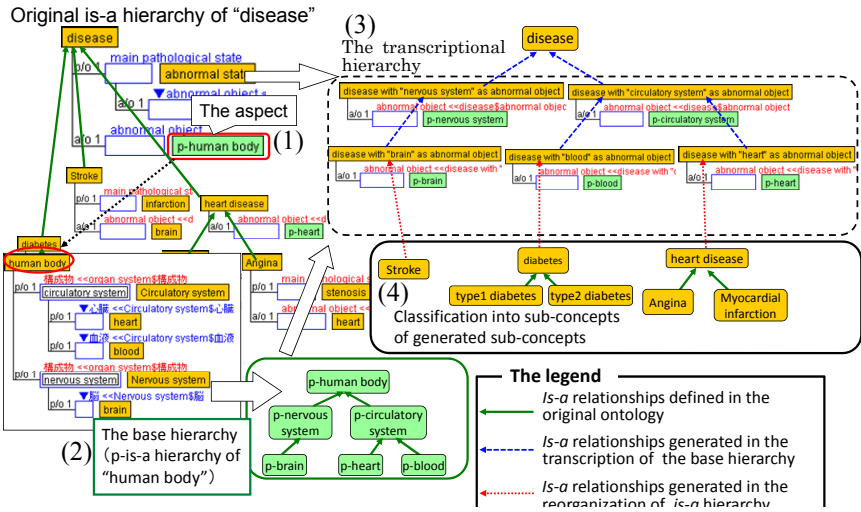


Fig. 4. An example of dynamic *is-a* generation of disease in the case that *p-is-a* hierarchy of human body is selected as the base hierarchy

between “disease with hyperglycemia as main pathological state” and “type1/type2 diabetes” can be identified by reasoning, “type1/type2 diabetes” are classified into sub-concepts of diabetes according to the original *is-a* relationships.

§2 In the Case of that an *p-is-a* Hierarchy is Selected as a Base Hierarchy

In the next example, we suppose the user selects the class-restriction of “abnormal object” as the aspect and the *p-is-a* hierarchy of “human body” as the base hierarchy for a dynamic *is-a* generation of disease in the same ontology with the previous example (Fig.4-(1),(2)).

In the property inheritance mechanism of ordinary *is-a* relationship, when a super class and its sub-class have the same slot, the class restriction of the sub-class’s slot must be a sub-class of the super-class’s one as well. However, in some case, the class restriction of the sub-class’s slot must be a part of the super-class’s. For example, when <disease of a pulmonary valve *is-a* disease of heart>, both “disease of a pulmonary valve” and “disease of a heart” have a slot of “site of the disease” and the class restriction of the former must be a part of the latter, that is <pulmonary valve *part-of* heart>.

To cope with such cases, on the basis of our latest theory of roles, we introduced “*p-*” operator in Hozo which automatically generates a generic concept representing all the parts of the entity to which the operator is attached. The operator enables parts to be inherited by ordinary property inheritance mechanism⁵. In the case of Fig.5, for example, we write “*p-heart*” instead of “heart”, and then the slot of its subclass

⁵ To deal with *p-is-a* hierarchies in OWL, we can represent them by some design pattern of ontologies such as SEP triple proposed by Udo Hahn and his group [8].

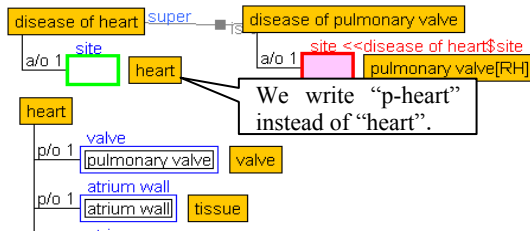


Fig. 5. An example of usage of p-operator

inherits not subclass of “heart” but its parts. When p-X is used, Hozo automatically generates a generic concept representing all of the defined parts of X including all parts which have X as their ancestor. This is valid because each part *is-a* subclass of “X’s parts class” which coincides with p-X. According to mereology, the theory of parts, p-X includes itself which is not the very X as an entity but X as its part.

Based on this theory, Hozo automatically generates *is-a* relationships between p-X such as \langle p-pulmonary valve *is-a* p-heart \rangle . As a result, an *is-a* hierarchy of p-X is generated according to part-of hierarchy of X. The *is-a* hierarchy of p-X is called p-*is-a* hierarchy and could be selected as a base hierarchy for a dynamic *is-a* generation.

In the case of Fig.4, since the class restriction of “abnormal object” is “p-human body”, we can select it as an aspect and p-*is-a* hierarchy as a base hierarchy for dynamic *is-a* hierarchy generation. Then, sub-concepts of “disease” such as “disease with p-nervous system as abnormal object” and “disease with p-circulatory system as abnormal object” are dynamically generated according to the aspect and the base hierarchy. As a result, the transcriptional hierarchy of “disease” based on p-*is-a* hierarchy of “p-human body” is obtained (Fig.4(3)). Then, the existing sub-concepts of “disease” are classified into the transcriptional hierarchy like Fig.4(4).

In addition to these examples, we can select *is-a* hierarchies which are generated using the proposed method as a base hierarchy to generate another *is-a* hierarchies. That is, our dynamic *is-a* generation could be executed recursively.

The dynamic *is-a* hierarchy generation is applicable to reorganizations of a portion of an *is-a* hierarchy. For example, we can select a middle-level concept (e.g. “disease of heart” as the target concept for the dynamic *is-a* generation.

In these ways, we can dynamically generate *is-a* hierarchies of diseases according to the selected aspects and base *is-a* hierarchies from various viewpoints.

3 Ontological Consideration on Generated *Is-a* Hierarchies

3.1 Three kinds of *Is-a* Relationship

We need to classify *is-a* relationships which appear in *is-a* hierarchies dynamically generated by the proposed method into the following three kinds according to their characteristics.

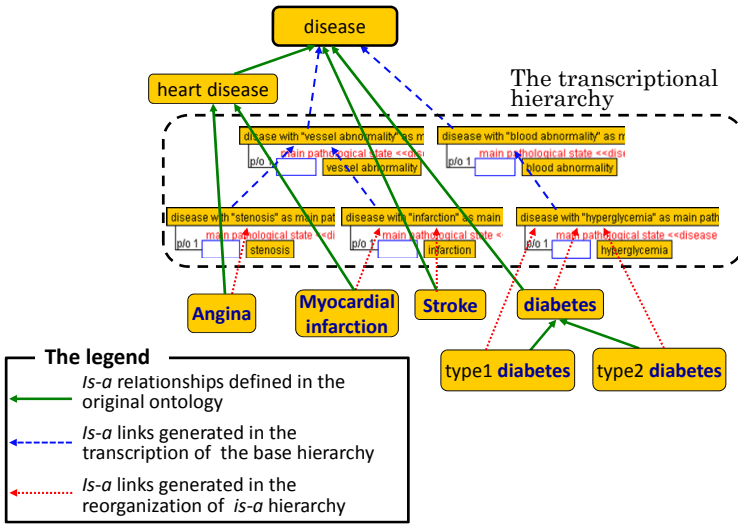


Fig. 6. Three kinds of *is-a* relationships in a generated *is-a* hierarchy

(A) *Is-a* relationships defined in the original ontology

They are *is-a* relationships which are defined in the original ontology before reorganizations. They are based on single-inheritance principle, and hence multiple-inheritance is not allowed following the ontological theory

(B) *Is-a* relationships generated in a transcription of a base hierarchy

They are automatically generated by the system when transcriptional hierarchies are generated according to the selected aspect and base hierarchy.

(C) *Is-a* relationships generated in a reorganization of *is-a* hierarchy

They are automatically generated by the system between existing sub-concepts and generated sub-concepts.

Note here that (B) and (C) are automatically generated by the system when dynamic *is-a* hierarchy generation is executed, whereas (A) are originally defined in the ontology by its developer.

For instance, Fig.6 shows these three kinds of *is-a* relationships in the generated *is-a* hierarchy illustrated in section 2.3.1. All the existing sub-concepts (e.g. Angina, diabetes) have *is-a* relationships of type (A). Therefore, when the existing concepts are classified into the generated concepts using *is-a* relationships of type (C), they always have multiple-inheritance according to two kinds of relationships of type (A) and (C). In order to identify from which concept an essential property is inherited to each generated concept, we have to distinguish these two kinds of *is-a* relationship in the generated *is-a* hierarchy. In the case of OWL, *is-a* relationships of type (B) correspond to sub-class-of relationships which are determined through classifications by reasoning using a DL reasoner. In the case of Hozo, such kind of *is-a* relationships

are represented by *IS-A* relationships⁶ which allows inheritance of only properties of super-concepts without identity criterion.

3.2 Consistency of *Is-a* Relationships

Since *is-a* links of type (B) and (C) are automatically generated by the system, there would be a concern about inconsistency of *is-a* relationships in the new *is-a* hierarchy. We investigate whether our method of automatic generation of *is-a* hierarchy causes inconsistencies or not.

§1 Consistency of *is-a* relationships of type (B)

Is-a links of type (B) are automatically generated by the system when generated sub-concepts are defined by specializing the definition of a target concept according to the selected aspect and base hierarchy. That is, only the target concept's definition specified by the aspect is specialized in the generated sub-concepts according to the base hierarchy. It also means that all the generated sub-concepts do not have any inconsistencies with its super-concepts as long as *is-a* relationships in the base hierarchy are consistent. Therefore, *is-a* relationships of type (B) are consistent in the generated *is-a* hierarchy as well.

§2 Consistency of *is-a* relationships of type (C)

Is-a links of type (C) are automatically generated by the system between existing sub-concepts and generated sub-concepts. Both of existing sub-concepts and generated sub-concepts inherit all properties of the target concept. Only when an existing sub-concepts' definition is subsumed by the definition of a generated sub-concept, the existing sub-concept is classified into a sub-concept of the generated sub-concept and an *is-a* link of type (C) is generated between them. Therefore, there is not any inconsistency such as inheritances of unintended properties or undefined properties between them.

The above discussion shows that *is-a* relationships of type (B) and (C) do not cause any inconsistency between the original *is-a* hierarchy and dynamically generated ones nor any change of definitions of existing sub-concepts while they are automatically generated by the system. Furthermore, because the proposed method does not generate *is-a* links between concepts defined in the original ontology, the original *is-a* hierarchy remains after reorganization.

That is, the proposed method enables us to dynamically generate *is-a* hierarchies without causing any inconsistency with the original ontology and changes of original definitions of concepts.

While many redundant concepts could be generated by the method, the user can manage them by specifying the number of the target layers rather than to use all concepts of the base hierarchy for transcription.

⁶ In Hozo, multi-inheritance is represented by distinction between an *is-a* relationship and *IS-A* relationship in order to identify from which concept an essential property is inherited.

4 Implementation

We implemented a prototype of dynamic *is-a* hierarchy generation system as an extended function of Hozo. The system was developed as a Java client application using HozoCore, which is Java API for ontologies built using Hozo, and Hozo OAT (Ontology Application Toolkit), which is Java library for GUI of ontology-based applications, developed using HozoCore.

The new function consists of three modules: *is-a* hierarchy viewer, viewpoint setting dialog, and dynamic *is-a hierarchy* generation module (Fig.7). The *is-a* hierarchy viewer shows an *is-a* hierarchy of an ontology in a tree representation. The user selects a target concept⁷ on the *is-a* hierarchy for a dynamic *is-a* hierarchy generation. The definition of the selected target concept is shown on the viewpoint setting dialog. In the dialog, the user selects a viewpoint for the dynamic *is-a* hierarchy generation by choosing class restriction of a slot as an aspect, a kind of base hierarchy and the number of target layers for a transcription of a base hierarchy according to his/her interests. The dynamic *is-a* hierarchy generation module generates *is-a* hierarchy according to the specified viewpoint. The generated *is-a* hierarchy is shown on the *is-a* hierarchy viewer and could be saved as an ontology file if required.

While the target of the system is an ontology in Hozo's format, it also can support an ontology in OWL because Hozo can import/export OWL ontologies. When the generated *is-a* hierarchy is exported in the OWL format, its generated sub-concepts in the transcriptional hierarchy are represented by *owl:equivalentClass* which have

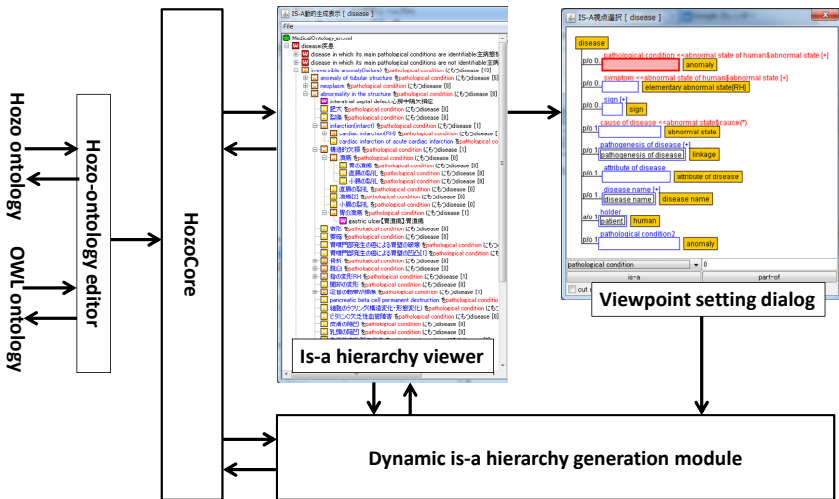


Fig. 7. The architecture of the dynamic *is-a* hierarchy generation system

⁷ When the user uses it as the additional function of Hozo, he/she selects a target concept on several GUIs of HOZO for ontology representations.

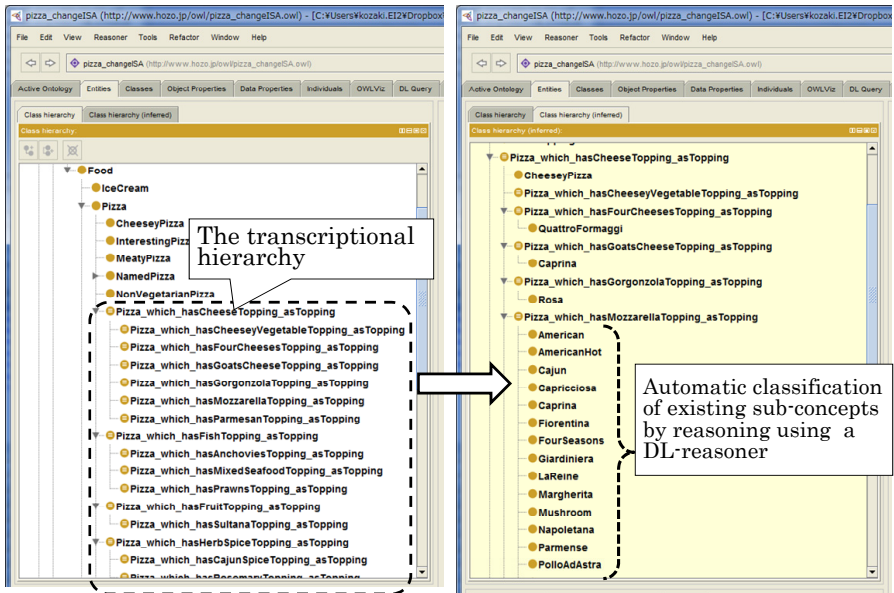


Fig. 8. The dynamically generated *is-a* hierarchy from Pizza Ontology exported in OWL format

restriction on properties selected as the aspect. On the other hand, *is-a* relationships of type (C) in the generated *is-a* hierarchy are not exported because they can be identified by reasoning using a DL reasoner. For example, Fig.8 shows a dynamically generated *is-a* hierarchy from Pizza Ontology⁸. As shown in the left of the figure, its transcriptional hierarchy is generated by selecting the restriction on *hasTopping* property as the aspect and the *is-a* hierarchy of *PizzaTopping* as the base hierarchy. The existing sub-concepts of *Pizza* are automatically classified by reasoning using a DL-reasoner like the right of Fig.8.

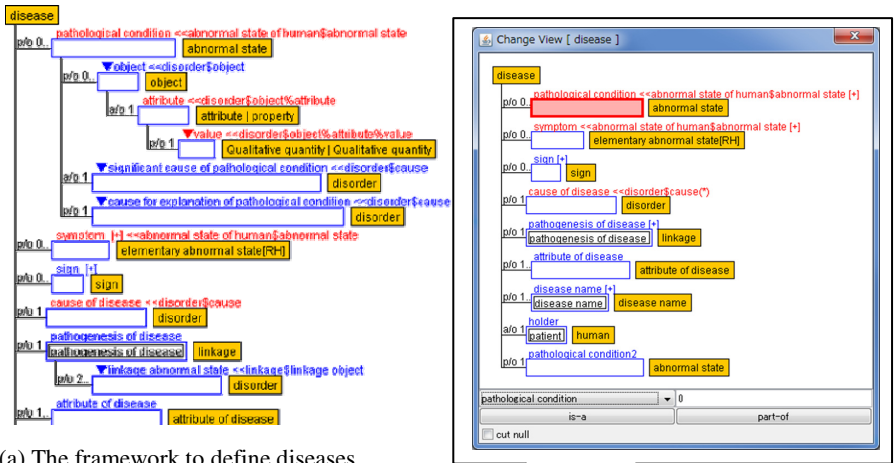
5 Application of Dynamic *Is-a* Generation to a Medical Ontology

We applied dynamic *is-a* hierarchy generation system to a medical ontology which we are developing in a project supported by Japanese government [7, 9]. In our medical ontology, diseases are defined by specifying typical *disorder roles*, such as *pathological condition*, *symptom*, played by *abnormal state*. Fig.9-(a) shows the framework to define diseases. Its *disorder roles* are represented as slots with class-restrictions for constraining slot values. These slots are used as aspects for dynamic generation of *is-a* hierarchies of diseases.

For example, when we select the *pathological condition* of disease as an aspect and the *is-a* hierarchy of abnormal state as the base hierarchy, the *is-a* hierarchy of disease is generated (Fig.9-(c)). In the generated *is-a* hierarchy, concepts which have names

⁸ <http://www.co-ode.org/ontologies/pizza/2007/02/12/>

represented by “disease which has X as pathological condition” (e.g. *disease which has abnormality in the structure as pathological condition*) are sub-concepts generated through the dynamic *is-a* hierarchy generation. Their concept names are automatically determined by the system using a template. Existing sub-concepts are reorganized as sub-concepts of them. For instance, *acute cardiac infarction* is classified into a sub-concept of *disease which has cardiac infarction as pathological condition*. From the generated *is-a* hierarchy, we can understand diseases according to the classification of pathological conditions.



The original is-a hierarchy of “disease” is generated to a hierarchy

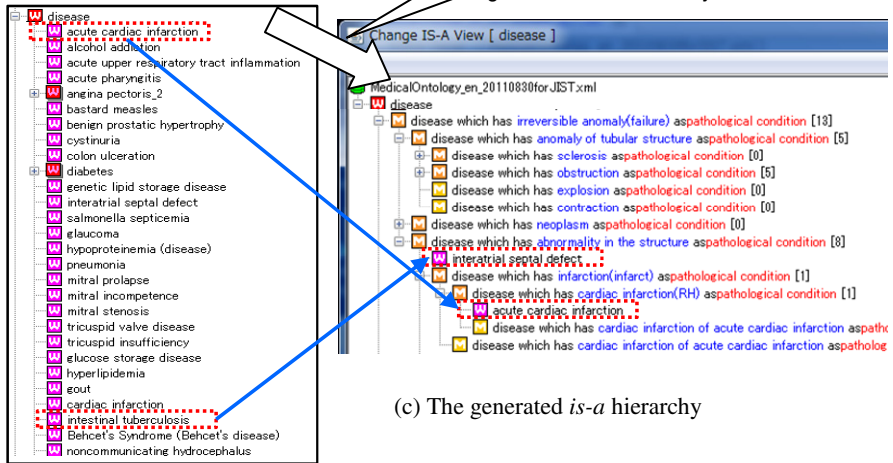


Fig. 9. Application of dynamic *is-a* generation to a medical ontology

On the other hand, when we select the *object of pathological condition* as an aspect and *p-is-a* hierarchy of the *human body* as a base hierarchy, the system generates the *is-a* hierarchy of disease which is similar to the part-whole hierarchy of the human body. For instance, *acute cardiac infarction* is classified into a sub-concepts of *disease which has a pathological condition in the myocardium*.

Although we use a prototype of the medical ontology which includes about 200 diseases in the above examples, about 6,000 diseases have been defined in the current version medical ontology by 12 clinicians. We already have applied dynamic *is-a* hierarchy generation to the latest version and confirmed it could reorganize *is-a* hierarchy of 6,000 diseases. Through dynamic *is-a* hierarchy generation according to the users' viewpoint, they can understand diseases from a variety of perspective. We believe it could contribute to deeper understanding of them.

Moreover, we have developed a medical information system to consider how the dynamic *is-a* hierarchy generation function can be used in other systems [10]. It is used as an index for semantic navigation in the system. We also performed an informal evaluation of the implemented system in a workshop⁹ and received favorable comments from medical experts. They especially liked the dynamic *is-a* hierarchy reorganization, which is the first solution to the multi-perspective issues of medical knowledge in the world.

Currently, we are refining the medical ontology based on a new disease model which captures disease as causal chains of disorders [9]. While the dynamic *is-a* hierarchy generation is applicable to the new medical ontology, we need to extend the proposed framework to cope with more sophisticated *is-a* hierarchy generation because we introduce a new kind of *is-a* relationships in the disease model based on an ontological consideration of causal chains.

We also plan to make more formal evaluation of the proposed method. Now, we are collecting several kinds of classification hierarchies of disease, and analyzing what kinds of viewpoints are considered in these classifications. Then, we are evaluating whether our method and medical ontology could support all kinds of them. We also plan applications of the dynamic *is-a* generation to other ontologies.

6 Related Work

In order to avoid multiple-inheritance, some researchers took an approach that they developed ontologies using single-inheritance and reorganized them by reasoning using a DL-reasoner [11]. It corresponds to reorganization of *is-a* hierarchy based on a transcriptional hierarchy in step 4 of the proposed method. However, the approach needs that the transcriptional hierarchy is developed in advance while it is dynamically generated by the system in the case of the proposed method.

Faceted Classification is used to represent classifications from multiple-perspectives. In the Semantic Web, some researchers proposed Faceted Search for semantic portals [12, 13]. They use Faceted Classification according to the user's

⁹ The number of participants was about 25. It includes not only the members of the medical ontology development but also others who work in the medical domain.

choose of facets from the definition of ontologies to provide user-centric semantic search. In order to formalize the Faceted Classification, Bene Rodriguez-Castro proposed an ontology design pattern to represent Faceted Classification in OWL [14]. Although the proposed method use a similar technique to Faceted Classification for transcription of a hierarchical structure, it is different from Faceted Classification since we focus on considerations of ontological meaning of generated *is-a* hierarchies. Introduction of a *p-is-a* hierarchy is one of the results of the ontological investigations.

However, there are some rooms to ontological investigate on a method of dynamic *is-a* hierarchy generation. For instance, we need to investigate *is-a* hierarchies of role-concepts and role-holders [5] while this paper concentrated on *is-a* hierarchies of basic concept (normal type). Dynamic *is-a* hierarchy generation based on more complicated viewpoints is also important subject to be considered. For example, we are considering viewpoints to cope with a new disease model based on an ontological consideration of causal chains [9]. Because the latest version of our medical ontology based on the new disease model has more rich definitions than previous one, it would support more complicated viewpoints for dynamic *is-a* hierarchy generation based on causal chains in diseases. We believe these ontological considerations would clarify the feature of the proposed method.

7 Concluding Remarks

In this paper, we discussed multi-perspective issues of *is-a* hierarchy construction in ontologies and proposed a method of dynamic generation of *is-a* hierarchies. The main idea is reorganization of *is-a* hierarchies from the original ontology according to viewpoints of users. Then, we made ontological consideration on *is-a* hierarchies which are developed by the proposed method, and we showed that dynamic *is-a* hierarchy generation does not cause any inconsistency between the original ontology and the generated one. Moreover, we developed a dynamic *is-a* hierarchy generation system as new function of Hozo and applied it to a medical ontology. It enables the users to understand an ontology from various viewpoints according to their intentions.

We plan to investigate further about some issues of the proposed framework for dynamic *is-a* hierarchy generation. Although this paper concentrated on *is-a* hierarchies of basic concept (normal type), we need to consider *is-a* hierarchies of role-concepts and role-holders [5]. Because they can be selected as an aspect for dynamic *is-a* hierarchy generation as well, we also have to consider about such cases. An extension of the proposed frame work to cope with the new disease model based on an ontological consideration of causal chains is another important topic should be considered.

Evaluation of the proposed method is also very important thing which we have to do. We plan to evaluate it through comparison of dynamically generated *is-a* hierarchies of disease and existing classification of disease.

The demonstration of the dynamic *is-a* hierarchy generation is available at <http://www.hozo.jp/demo/>. The function is also supported by the latest version of

Hozo. Currently, we are also developing the dynamic *is-a* hierarchy generation system for OWL ontologies using OWL-API while it is partly available through OWL import/export function of Hozo.

Acknowledgement. A part of this research is supported by the Ministry of Health, Labor and Welfare, Japan, the Japan Society for the Promotion of Science (JSPS) through its “Funding Program for World-Leading Innovative R&D on Science and Technology (FIRST Program)”, Grant-in-Aid for Young Scientists (A) 20680009.

References

1. Gruber, T.: A translation approach to portable ontology specifications. In: Proc. of JKAW 1992, pp. 89–108 (1992)
2. Guarino, N.: Some Ontological Principles for Designing Upper Level Lexical Resources. In: Proc. of International Conference on Lexical Resources and Evaluation (1998)
3. Smith, B., et al.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology* 25(11), 1251–1255 (2007)
4. Kozaki, K., Kitamura, Y., Ikeda, M., Mizoguchi, R.: Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of Role and Relationship. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 213–218. Springer, Heidelberg (2002)
5. Mizoguchi, R., et al.: A Model of Roles within an Ontology Development Tool: Hozo. *J. of Applied Ontology* 2(2), 159–179 (2007)
6. Kozaki, K., et al.: Role Representation Model Using OWL and SWRL. In: Proc. of 2nd Workshop on Roles and Relationships in Object Oriented Programming, Multi-agent Systems, and Ontologies, Berlin, July 30-31 (2007)
7. Mizoguchi, R., et al.: An Advanced Clinical Ontology. In: Proc. of ICBO, pp. 119–122 (2009)
8. Hahn, U., et al.: Turning Lead into Gold? In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 182–196. Springer, Heidelberg (2002)
9. Mizoguchi, R., et al.: River Flow Model of Diseases. In: Proc. of ICBO 2011, pp. 63–70 (2011)
10. Kou, H., Ohta, M., Zhou, J., Kozaki, K., Mizoguchi, R., Imai, T., Ohe, K.: Development of Fundamental Technologies for Better Understanding of Clinical Medical Ontologies. In: Proc. of International Conference on Knowledge Engineering and Ontology Development (KEOD 2010), Valencia, Spain, October 25-28, pp. 235–240 (2010)
11. Adams, N., Cannon, E.O., Murray-Rust, P.: ChemAxiom -An Ontological Framework for Chemistry in Science. In: Proc. of ICBO, pp. 15–18 (2009)
12. Suominen, O., Viljanen, K., HyvÄnen, E.: User-Centric Faceted Search for Semantic Portals. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 356–370. Springer, Heidelberg (2007)
13. Holli, M.: Crisp, Fuzzy, and Probabilistic Faceted Semantic Search, PhD Thesis, Aalto University, Finland (2010)
14. Rodríguez-Castro, B., Glaser, H., Carr, L.: How to Reuse a Faceted Classification and Put it on the Semantic Web. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 663–678. Springer, Heidelberg (2010)

Mid-Ontology Learning from Linked Data

Lihua Zhao and Ryutaro Ichise

Principles of Informatics Research Division,
National Institute of Informatics, Tokyo, Japan
{lihua,ichise}@nii.ac.jp

Abstract. The Linking Open Data(LOD) cloud is a collection of linked Resource Description Framework (RDF) data with over 26 billion RDF triples. Consuming linked data is a challenging task because each data set in the LOD cloud has specific ontology schema, and familiarity with ontology schema is required in order to query various linked data sets. However, manually checking each data set is time-consuming, especially when many data sets from various domains are used. This difficulty can be overcome without user interaction by using an automatic method that integrates different ontology schema. In this paper, we propose a Mid-Ontology learning approach that can automatically construct a simple ontology, linking related ontology predicates (class or property) in different data sets. Our Mid-Ontology learning approach consists of three main phases: data collection, predicate grouping, and Mid-Ontology construction. Experimental results show that our Mid-Ontology learning approach successfully integrates diverse ontology schema, and effectively retrieves related information.

Keywords: Mid-Ontology, linked data, semantic web, ontology learning.

1 Introduction

The Linking Open Data (LOD) cloud^[1] is a collection of Resource Description Framework (RDF) data in <subject, predicate, object> triples^[12]. The latest LOD cloud contains 203 data sets mainly categorized into seven domains: cross-domain, geographic, media, life sciences, government, user-generated content, and publications. Things are represented using the Uniform Resource Identifier (URI), and identical or related things are linked with the predicate owl:sameAs, where OWL is the Web Ontology Language designed to share or publish ontologies on the Web^[2].

Although many applications such as linked data browsers, ontology-driven semantic search engines, and some domain specific applications have been developed by consuming linked data sets, integrating ontology schema or data sets from diverse domains remains a challenging problem^[2]. This is because multiple data sets provide different values for the same predicate of an object or provide different terms to represent the same predicate^[1]. SPARQL query is a powerful RDF query language that enables users to access linked data^[12]. However,

¹ <http://lod-cloud.net/>

users have to understand ontology schema of data sets to construct SPARQL queries. Learning all the ontology schema is not feasible and is time-consuming, because each data set has a specially designed ontology and thousands of distinct ontology predicates might exist. Querying with one simple ontology that integrates various ontologies can simplify SPARQL queries and help semantic web application developers to easily understand ontology schema to search on linked data sets.

In order to solve this problem, an automatic method to create a simple ontology that integrates ontology schema from diverse domain data sets needs to be developed. Ontology learning technology can automate the ontology construction process from structured, semi-structured or unstructured data [7]. An ontology learning cycle that includes ontology design, ontology learning, and validation phases is introduced in [18]. However, most of the research on the ontology learning technology focuses on text files. We designed an automatic ontology learning approach to adapt to the LOD data sets, which can create an integrated ontology from diverse data sets.

In this paper, we present an automatic Mid-Ontology learning method, which includes ontology manipulations such as ontology term extraction, ontology matching, and ontology integration. Ontology integration is defined as a process that generates a single ontology from different existing ontologies [3]. An automatically constructed ontology is called a Mid-Ontology, which integrates related ontology predicates from diverse linked data sets.

This paper is organized as follows. In Section 2 we introduce our automatic Mid-Ontology learning approach that involves data collection, predicate grouping, and Mid-Ontology construction. In Section 3, we evaluate our Mid-Ontology learning approach from four different aspects: effectiveness of data reduction, quality of Mid-Ontology, effectiveness of information retrieval with SPARQL query, and characteristics of integrated predicates. In Section 4, we discuss advantages of our Mid-Ontology learning approach and possible applications. In Section 5, we list some previous related research and compare this research with our approach. In Section 6, we present our conclusions and propose future work.

2 Mid-Ontology Learning Approach

In the LOD cloud, data sets are linked with owl:sameAs at an instance level, but few links exist at the class or property level. Although the RDF link types owl:equivalentClass and owl:equivalentProperty are designed to state two classes or properties actually refer to the same concept, there are only few links at a class level or property level [12]. Hence, whenever linked data are queried with SPARQL, the predicates of ontology schema must be manually learnt, which is not conceivable if there are thousands of distinct predicates.

Our aim is to automatically construct a simple ontology that integrates ontology schema from linked data sets. Hence, we focus on instances linked with owl:sameAs to retrieve related information. For example, Table 1 shows the

Table 1. Collected data based on the URI: “http://dbpedia.org/resource/Berlin”

Predicate	Object
<i>http://dbpedia.org/property/name</i>	Berlin
<i>http://dbpedia.org/property/population</i>	3439100
<i>http://dbpedia.org/property/plz</i>	10001-14199
<i>http://dbpedia.org/ontology/postalCode</i>	10001-14199
<i>http://dbpedia.org/ontology/populationTotal</i>	3439100
.....
<i>http://www.geonames.org/ontology#alternateName</i>	Berlin
<i>http://www.geonames.org/ontology#alternateName</i>	Berlyn@a/f
<i>http://www.geonames.org/ontology#population</i>	3426354
.....
<i>http://www.w3.org/2004/02/skos/core#prefLabel</i>	Berlin (Germany)
<i>http://data.nytimes.com/elements/first_use</i>	2004-09-12
<i>http://data.nytimes.com/elements/latest_use</i>	2010-06-13

collected <predicate, object> pairs of instances that indicate “Berlin” from DBpedia, Geonames, and NYTimes. In Table 1, there are three distinct predicates that indicate “Berlin” and belong to different data sets. If we can integrate these related predicates into one predicate, we can query all the data sets with one single predicate that indicates the name of a place.

In this section, we describe the architecture of our Mid-Ontology learning approach as shown in Fig. 1. The architecture of our approach includes three phases: data collection, predicate grouping, and Mid-Ontology construction.

2.1 Data Collection

Although the SameAs (owl:sameAs) link is designed to link identical things, it also links related or similar things in published linked data sets [11]. Hence, we can find identical or related information if we investigate instances linked with owl:sameAs. In this section, we describe our data collection phase in three steps: extract data linked with owl:sameAs, remove noisy instances of the core data set, and collect predicates and objects to construct the final data set for our Mid-Ontology learning.

Extract Data Linked with owl:sameAs

In order to extract data linked with owl:sameAs, we have to select a core data set, which serves as a hub of several linked data sets. A good core data set should have inward or outward links to other data sets from diverse domains. Then we collect all the instances that have the SameAs links with instances in the core data set.

For instance, suppose we select DBpedia as the core data set, we select all the DBpedia instances that have SameAs links to other data sets such as Geonames and NYTimes. Table 1 shows an example of the collected data based

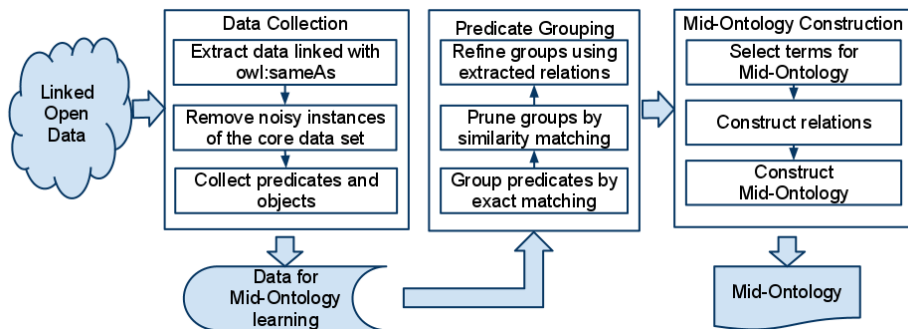


Fig. 1. Architecture of our Mid-Ontology learning approach

on the DBpedia instance “<http://dbpedia.org/resource/Berlin>”. Since both the Geonames instance “<http://sws.geonames.org/2950159/>” and the NYTimes instance “<http://data.nytimes.com/N50987186835223032381>” are linked with this instance, we collect the contents of these two URIs.

Remove Noisy Instances of the Core Data Set

We say an instance is *noisy*, if none of the triples of the instance contains information that can represent the characteristics of the instance. For instance, if all the triples of an instance are SameAs links, broken links, or types, we cannot learn any information that can represent the characteristics of an instance. We remove all the noisy instances of the core data set before collecting predicates and objects.

Collect Predicates and Objects

Data collection is based on each retrieved instance of the core data set, where we collect all the $\langle \text{predicate}, \text{object} \rangle$ pairs of the instance and of instances connected with the instance. Hereafter, we use PO to represent a $\langle \text{predicate}, \text{object} \rangle$ pair. The explanation of variables and functions appear in this paper are illustrated in Table 2. In this PO collection step, we do not collect the triples with the SameAs links, because we have already collected triples of the linked instances. Furthermore, for the instances of the core data set, if there is any link to another instance of the core data set, we also collect triples from the linked instance. The collected data consists of PO pairs based on each instance of the core data set, which is used for the predicate grouping phase.

2.2 Predicate Grouping

Grouping related predicates from different ontology schema is critical for Mid-Ontology construction, because there exist many similar ontology predicates that

Table 2. Explanation of variables and functions

Variables/Functions	Explanation
PO	A <predicate, object> pair.
G	A group of PO pairs.
$T(G)$	Pre-processed terms of predicates in G .
$O(G)$	Objects stored in G .
$P(G)$	Predicates stored in G .
$WNSim(T(G_i), T(G_j))$	WordNet-based similarity between $T(G_i)$ and $T(G_j)$.
$StrSim(O(G_i), O(G_j))$	String-based similarity between $O(G_i)$ and $O(G_j)$.
$Sim(G_i, G_j)$	Similarity between G_i and G_j .

represent the same thing. The predicate grouping phase consists of three main steps: group predicates by exact matching, prune groups by similarity matching, and refine groups using extracted relations.

Group Predicates by Exact Matching

The first step in predicate grouping is creating initial groups of predicates that share the same information or the same object by the exact string matching method. Collected data set is based on each instance of the selected core data, which consists of PO pairs. We perform pairwise comparison of PO_i and PO_j , and create initial groups G_1, G_2, \dots, G_k by checking whether they share identical predicate or object. Here, G is a group of PO pairs as explained in Table 2.

For example, in Table 1, both `geo-onto:alternateName`² and `db-prop:name`³ have the same object “Berlin”, and the predicate `geo-onto:alternateName` has another object “Berlyn@af”. Hence, these two predicates and objects are grouped together to create an initial group. After creating initial groups for all the PO pairs, we create initial groups for each PO pair that has not yet been grouped. For instance, `nyt-prop:first_use`⁴ is in an initial group by itself, because no predicate has the same object “2004-09-12” and no identical predicate exists in the data.

Prune Groups by Similarity Matching

The second step in predicate grouping is pruning initial groups by knowledge-based similarity matching and string-based similarity matching, which are commonly used to match ontologies at the concept level [9]. As we noticed that some of the same values that are written in different languages or some semantically identical words such as U.K. and United Kingdom may be ignored in exact matching, we realized that similarity matching is necessary to group semantically similar predicates.

In our approach, we adopted nine knowledge-based similarity measures [15], namely, LCH, RES, HSO, Jiang and Conrath (JCN), LESK, PATH, Wu and

² `geo-onto` is the abbreviation of <http://www.geonames.org/ontology#>

³ `db-prop` is the abbreviation of <http://dbpedia.org/property/>

⁴ `nyt-prop` is the abbreviation of <http://data.nytimes.com/elements/>

Table 3. Example of two groups of predicates from Table 1

Group	Predicates	Corresponding Object
G_i	http://dbpedia.org/property/population	3439100
	http://dbpedia.org/ontology/populationTotal	
G_j	http://www.geonames.org/ontology#population	3426354

Palmer (WUP), LIN, and VECTOR, which are based on WordNet (a large lexical database of English [10]), and four string-based similarity measures, namely, prefix, suffix, Levenshtein distance, and n-gram, as introduced in [13]. Knowledge-based similarity measures are applied to compare pre-processed terms of predicates, because the terms of predicates are more likely to have semantic meanings. On the other hand, string-based similarity measures are applied to compare objects of predicates, because objects may contain URIs rather than lexical labels. In the following, the term $T(G)$ indicates the pre-processed terms of predicates in G , the term $O(G)$ indicates the objects stored in G , and the term $P(G)$ indicates the predicates stored in G , as shown in Table 2.

In order to extract terms of predicates, we pre-process each predicate of PO pairs by performing natural language processing (NLP), which includes tokenizing terms, removing stop words, and stemming terms using the porter stemming algorithm [16]. NLP is a key method for the data pre-processing phase in which terms are extracted from ontologies; this method helps to improve the performance of ontology building [4].

For initial groups $\{G_1, G_2, \dots, G_k\}$, we apply similarity measures on pairwise initial groups. $Sim(G_i, G_j)$ is the similarity value between G_i and G_j calculated using the formula:

$$Sim(G_i, G_j) = \frac{WNSim(T(G_i), T(G_j)) + StrSim(O(G_i), O(G_j))}{2}$$

where $WNSim(T(G_i), T(G_j))$ is the average of the nine applied WordNet-based similarity values and $StrSim(O(G_i), O(G_j))$ is the average of the four string-based similarity values. For WordNet-based similarity measures, we do not count values of zero, which indicate that no similarity values have been returned from WordNet-based similarity measures.

If $Sim(G_i, G_j)$ is higher than a predefined threshold, we consider that two initial groups share similar predicates, and we merge these two groups. After comparing all the pairwise initial groups, we remove the initial group G_i , if it has not been merged during this pruning process and has only one PO pair.

Here, we show how to calculate the similarity between two initial groups G_i and G_j as listed in Table 3, where these two initial groups are created based on Table 1. Suppose G_i includes db-prop:population and db-onto: populationTotal⁵ with the object “3439100”, and group G_j includes the predicate geo-onto:population with the object “3426354”. $T(G_i)$ includes “population” and “total”, while $T(G_j)$ includes “population”. Here, $O(G_i)$ is “3439100” and $O(G_j)$ is “3426354”.

⁵ db-onto is the abbreviation of <http://dbpedia.org/ontology/>

Table 4. Example of WordNet-based similarity measures on pairwise terms

Pairwise Terms	LCH	RES	HSO	JCN	LESK	PATH	WUP	LIN	VECTOR
population, population	1	1	1	1	1	1	1	1	1
population, total	0.4	0	0	0.06	0.03	0.11	0.33	0	0.06

Table 5. Example of String-based similarity measures on pairwise objects

Pairwise Objects	prefix	suffix	Levenshtein distance	n-gram
“3439100”, “3426354”	0.29	0		0

Table 4 shows WordNet-based similarity values of pairwise terms in G_i and G_j . $WNSim(T(G_i), T(G_j))$ is 0.5825, which is the average of 15 similarity values larger than zero, as listed in Table 4. $StrSim(O(G_i), O(G_j))$ is 0.145, which is the average of four string-based similarity values, as shown in Table 5. Hence, the final similarity $Sim(G_i, G_j)$ is 0.36375, which is the average of 0.5825 and 0.145. If this value is higher than the predefined threshold, we merge G_i and G_j .

Refine Groups Using Extracted Relations

The final group refining step is to split the predicates of each pruned group G_i according to the relations of `rdfs:domain` or `rdfs:range`. Because even though the objects or terms of predicates are similar, the predicates may belong to different domains or ranges. For further refining, we count the frequency of $P(G_i)$ in all of the data, and keep the $P(G_i)$ that appears with a frequency that is higher than the predefined frequency threshold. The final refined groups of predicates are passed to the next phase, which is Mid-Ontology construction.

2.3 Mid-Ontology Construction

According to the groups of predicates, we construct the Mid-Ontology with automatically selected terms and a specially designed predicate.

Select Terms for Mid-Ontology

In order to perform automatic term selection, we pre-process all the terms of the predicates in each group by tokenization, stop words removal, and stemming. We also keep original terms, because sometimes one single word is ambiguous when it is used to represent a group of terms. For example, “area” and “areaCode” have totally different meanings, but may have the same frequency because the former is extracted from the latter. Hence, when two terms have the same frequency, we choose the longer one. The predicate `mo-onto:Term` is designed to represent a term of a class, where the “Term” is automatically selected.

Construct Relations

We designed a predicate `mo-prop:hasMembers` to link groups of predicates with the Mid-Ontology classes. This predicate indicates that a group of integrated predicates are members of a class of the Mid-Ontology.

Construct Mid-Ontology

A Mid-Ontology is automatically created with refined groups of integrated predicates, automatically selected terms, and a designed predicate `prop:hasMembers`, which links groups of predicates and Mid-Ontology classes.

2.4 Implementation

Many semantic web tools are developed to help researchers query through the linked data, publish linked data, or manage enormous data sets. Virtuoso⁶ is a high-performance server that supports storage of a large RDF data, provides SPARQL endpoint, and supports creation of RDF models [8]. A Virtuoso Jena RDF Data Provider is also provided, which enables Java applications to directly query the Virtuoso RDF data through Jena RDF Frameworks.

For knowledge-based similarity matching, we used WordNet::Similarity⁷ [15], which is implemented in perl. Several WordNet-based similarity measuring algorithms are implemented in this tool. If two terms are identical, we return 1; otherwise, we apply WordNet-based similarity measures. The similarity measures JCN, PATH, WUP, LIN, and VECTOR return normalized values between zero and one. However, the similarity values of LCH, RES, HSO, and LESK are not normalized. In order to normalize similarity values of LCH, RES, HSO, and LESK, we divide the returned value by the maximum value that we can obtain from all the pairwise terms in the collected data. The normalized similarity Sim_{alg} is calculated using the formula:

$$Sim_{alg} = \frac{WordNet_{alg}}{Max_{alg}}$$

where $WordNet_{alg}$ indicates the returned value from WordNet::Similarity tool, and the Max_{alg} indicates the maximum value we obtained from the $WordNet_{alg}$. The Max_{alg} of LCH, RES, HSO and LESK are 3.7, 10, 16 and 5.6 respectively.

3 Experimental Evaluation

In this section, we evaluate our Mid-Ontology learning approach from different aspects after introducing the experimental data. First, we evaluate the effectiveness of data reduction during the data collection phase. Second, we evaluate the quality of created Mid-Ontology to show the improvements achieved with and without our approach. Third, we evaluate our approach with a SPARQL example to demonstrate that we can successfully extract information with single integrated predicate without understanding all the related predicates. Fourth, we show that from the characteristics of integrated predicates, we can easily observe how instances from different data sets are linked together.

⁶ <http://virtuoso.openlinksw.com/>

⁷ <http://wn-similarity.sourceforge.net/>

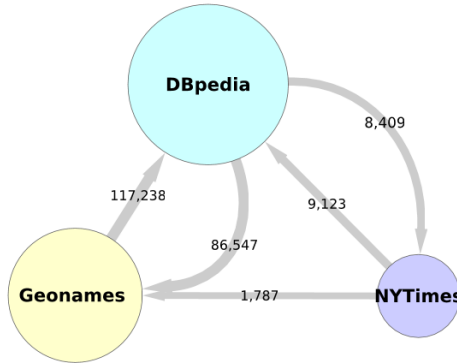


Fig. 2. SameAs links between data sets

3.1 Experimental Data

We used the following three data sets in the LOD cloud to evaluate our approach.

DBpedia is a core cross-domain data set that describes over 3.5 million things including persons, places, music albums, films, video games, organizations, species, and diseases. DBpedia has more than 232 million RDF triples and more than 8.9 million distinct URIs.

Geonames is a data set that is categorized in the geographic domain and contains more than 7 million unique URIs that represent geographical information on places across the world.

NYTimes data is a small data set that consists of 10,467 subject news, where 4,978 are about people, 1,489 are about organizations, 1,910 are about locations, and 498 are about descriptors.

Fig. 2 shows the sameAs links connecting the above three data sets, plotted using Cytoscape [17]. In this figure, the size of a node is determined by the total number of distinct instances in a data set, in a logarithmic scale. The thickness of an arc is determined by the number of sameAs links as labeled on each arc, in a logarithmic scale.

3.2 Evaluation of Data Reduction

We evaluate the effectiveness of data reduction during the data collection phase by comparing the number of instances in the original data sets with the number of instances we extracted after performing owl:sameAs retrieval process and noisy data removal process.

DBpedia has served as a hub within the Web of Data, because of the breadth of topical coverage and the wealth of inward and outward links connecting instances in DBpedia to instances in other data sets [12]. Hence, we select DBpedia as a core data set, and collect all the instances that have the SameAs links with instances in DBpedia.

Table 6. Number of distinct instances during data collection phase

Data set	Before reduction	owl:sameAs retrieval	Noisy data removal
DBpedia	8,955,728	135,749	88,506
Geonames	7,479,714	128,961	82,054
NYTimes	10,467	9,226	8,535

The instances of DBpedia with no more than three distinct predicates are defined as “noisy” instances and are removed from collected linked instances. This is because we observed that these instances contain predicates owl:sameAs, db-prop:wordnet-type⁸ and db-prop:hasPhotoCollection which links to a broken URI. Although these instances have SameAs links, we cannot learn any information that can represent the characteristics of a DBpedia instance.

Table 6 illustrates the number of distinct instances that exist before and after linked data retrieval and noisy data removal are performed during the data collection process. The retrieved linked sub-data set contains 135,749 DBpedia instances, 128,961 Geonames instances, and 9,226 NYTimes instances, while our database contains 8,955,728 DBpedia instances, 7,479,714 Geonames instances, and 10,467 NYtimes instances, respectively.

Then, we pre-process the extracted sub-data set by removing noisy instances of DBpedia. After noisy data removal, we obtained 88,506 DBpedia instances, 82,054 Geonames instances, and 8,535 NYTimes instances, which are 65%, 64% and 92.5% of the number of instances in the extracted linked data, respectively. DBpedia contains 7,964 distinct predicates, while Geonames and NYTimes contain 26 and 14, respectively, in the collected data set. We only retain 1,229 DBpedia predicates from among the 88,506 DBpedia instances by removing predicates that have a frequency of less than 100 and by retaining terms that have a length between 1 and 26.

We dramatically scaled down the data sets by collecting information of linked instances in the data collection phase to keep instances that share related information. Furthermore, we successfully removed noisy instances, which may affect the quality of created ontology.

3.3 Ontology Evaluation

In order to evaluate the quality of the created Mid-Ontology(MO), we calculate the accuracy of the Mid-Ontology using the following formula:

$$ACC(MO) = \frac{\sum_{i=1}^n \frac{|Correct\ Predicates\ in\ C_i|}{|C_i|}}{n}$$

where n is the number of classes in the Mid-Ontology, and $|C_i|$ indicates the number of predicates in class C_i . The ACC(MO) is the average of the accuracy of

⁸ db-prop is the abbreviation of <http://dbpedia.org/property/>

Table 7. Improvement achieved by our approach

MO	Number of Classes	Number of Predicates	Cardinality	Correct Groups	Correctly Labeled	Accuracy
MO _{no_p_r}	11	300	27.27	3(27.27%)	9(81.81%)	68.78%
MO _{no_p}	17	270	15.88	5(29.41%)	13(76.47%)	68.96%
MO _{no_r}	26	230	8.85	19(73.08%)	20(76.92%)	89.13%
MO	29	180	6.21	22(75.86%)	24(82.75%)	90.10%

each class in the Mid-Ontology. If all related or identical predicates are correctly integrated in each class, the accuracy $ACC(MO)$ reaches 1.

Table 7 shows the improvements achieved by our Mid-Ontology approach through a comparison of the Mid-Ontologies created with and without our approach. The main features of our approach are group pruning with similarity measures and group refining by checking the ranges and domains of predicates. The first column lists four Mid-Ontologies created by different approaches, i.e., MO_{no_p_r}, that without the pruning and refining processes, MO_{no_p}, that denotes without the pruning process, MO_{no_r}, that denotes without the refining process, and MO, that denotes with both pruning and refining processes. We manually check each group of predicates to determine whether they share identical or related information, and we examine whether each term can be represented for predicates in that class without disambiguation.

We can compare the results of MO_{no_p_r} and MO_{no_p}, MO_{no_r} and MO to evaluate the performance of the refining process. Although the refining process only slightly improved the accuracy of the Mid-Ontology from 68.78% to 68.96% and from 89.13% to 90.10%, it can divide classes into more specific topics according to ranges and domains. The performance of the pruning process can be evaluated by comparing the results of MO_{no_p_r} and MO_{no_r}, MO_{no_p} and MO. The group pruning process significantly improved the accuracy of the Mid-Ontology, i.e., from 68.78% to 89.13% and from 68.96% to 90.10%.

The last row shows our final Mid-Ontology obtained by applying both the pruning and refining processes. The accuracy of the Mid-Ontology and percentage of correctly grouped classes are significantly improved compared to the values in the first row, which were obtained when the pruning and refining processes were not performed. Although the percentage of correctly labelled terms is higher than in the case of methods that do not involve our approach, the accuracy of labels is not affected by the pruning and refining processes.

As Table 7 shows, when both pruning and refining are conducted, the number of correctly grouped classes and the accuracy of the Mid-Ontology can be improved. Furthermore, the total number of integrated predicates from the three data sets are reduced after applying each process, and the number of classes are increased. The cardinality is calculated using the formula:

$$Cardinality = \frac{|Number\ of\ Predicates|}{|Number\ of\ Classes|}$$

Table 8. Predicates grouped in mo-onto:population

```

<rdf:Description rdf:about="mid-onto:population">
<mo-prop:hasMembers rdf:resource="http://dbpedia.org/property/population"/>
<mo-prop:hasMembers rdf:resource="http://dbpedia.org/property/popLatest"/>
<mo-prop:hasMembers rdf:resource="http://dbpedia.org/property/populationTotal"/>
<mo-prop:hasMembers rdf:resource="http://dbpedia.org/ontology/populationTotal"/>
<mo-prop:hasMembers rdf:resource="http://dbpedia.org/property/einwohner"/>
<mo-prop:hasMembers rdf:resource="http://www.geonames.org/ontology#population"/>
</rdf:Description>

```

Table 9. SPARQL Example 1: Find places with a population of more than 10 million

```

SELECT DISTINCT ?places
WHERE{
  mid-onto:population mo-prop:hasMembers ?prop.
  ?places ?prop ?population. FILTER (xsd:integer(?population) > 10000000). }

```

where the cardinality indicates the average number of predicates in a class of the ontology. A low cardinality indicates that we successfully removed redundant predicates, which may reduce the accuracy of the Mid-Ontology.

The Mid-Ontology integrated 180 predicates from three different data sets and assigned them to 29 classes, with an average cardinality of 6.21. On the other hand, other Mid-Ontologies created without pruning or refining process have higher cardinality values than our Mid-Ontology, but have lower accuracies.

Comparing our Mid-Ontology with the Mid-Ontologies created without our approach, the total number of integrated predicates decreased by 40%, but the number of classes increased by a factor of almost 2.6, with a 30% improvement in accuracy. This means that the average cardinality of classes decreased and that unrelated predicates were successfully removed to improve the accuracy in each class. From the results in Table 7, we can conclude that our method can remove unrelated predicates that cause a low accuracy, and the refining process can filter out predicates with different ranges and domains.

3.4 Evaluation with a SPARQL Example

We evaluate the effectiveness in information retrieval with the Mid-Ontology created with our approach, by presenting a SPARQL query example. Table 8 shows one of the 29 groups in the Mid-Ontology, that integrates predicates that indicate population from DBpedia and Geonames. This group does not contain any NYTimes predicate, because there is no predicate that indicates population in the NYTimes data. We observed that NYTimes instances are linked with other data sets according to the labels of news headings, which are represented by “http://www.w3.org/2004/02/skos/core#prefLabel”.

The advantage of our approach is that we can retrieve related information with an automatically integrated Mid-Ontology. Table 9 shows a SPARQL example

Table 10. Results with each single predicate under the same condition as in Table 9

Single property for population	Number of Results
http://dbpedia.org/property/population	177
http://dbpedia.org/property/popLatest	1
http://dbpedia.org/property/populationTotal	107
http://dbpedia.org/ontology/populationTotal	129
http://dbpedia.org/property/einwohner	1
http://www.geonames.org/ontology#population	244

Table 11. Sample classes in the Mid-Ontology

DBpedia	DBpedia & Geonames	DBpedia & Geonames & NYTimes
mo-onto:birthdate	mo-onto:population	mo-onto:name
mo-onto:deathdate	mo-onto:prominence	mo-onto:long
mo-onto:motto	mo-onto:postal	

in which this `mo-onto:population` is used to find places that have a population of more than 10 million. This SPARQL query automatically queries with all the predicates listed under `mo-onto:population` as shown in Table 8.

We find 517 places with `mo-onto:population`, while with the single predicate listed in Table 10, we can find 177, 1, 107, 129, 1, and 244 places, respectively. Since the predicates grouped in this class all correctly represent population, the returned results are all correct. The results queried with `mo-onto:population` are a combination of the results retrieved with each predicate in that group. Furthermore, it is difficult to manually find all the six predicates that indicate population in different data sets.

As this example shows, our approach simplifies SPARQL queries and returns all the possible results without user interaction; in contrast, it is time-consuming to find each single predicate manually through user interaction.

3.5 Characteristics of Integrated Ontology Predicates

We evaluate whether our approach successfully integrates related predicates by illustrating samples of classes in the Mid-Ontology. Table 11 shows some of the classes in the Mid-Ontology, which integrated predicates from DBpedia, Geonames, and NYTimes instances. The classes listed in the first column include only predicates from DBpedia ontology, which indicate the birth date, death date, and motto of a person. The second column lists classes that integrate predicates from both DBpedia and Geonames, which indicate the population, prominence, and postal code of a place. The last column lists classes that integrate predicates from DBpedia, Geonames, and NYTimes, which indicate the name of a thing and the longitude of a place.

The class `mo-onto:name` indicates the name of a person, an event, or a place, which integrates predicates from all the three data sets. From the characteristics of integrated classes, we can observe that the linked instances between DBpedia

and Geonames are about places, and the instances that link DBpedia, Geonames, and NYTimes are based on a person or an event happened in a place.

4 Discussion

Experimental results demonstrate that our Mid-Ontology learning approach successfully integrates predicates from different data sets. The automatically created Mid-Ontology has a high quality, and can be applied in the information retrieval field. Since the Mid-Ontology integrates the most related predicates, we can search potential related triples or instances from LOD cloud with a simple SPARQL query.

Furthermore, our Mid-Ontology learning approach is implemented with the collected data set that is extracted with owl:sameAs. Hence, our Mid-Ontology can find missing links that should be linked with owl:sameAs. For example, the predicate mo-onto:population has predicates of DBpedia and Geonames that indicate population. Therefore, we can find the same place in DBpedia and Geonames by searching for places with the same population.

For instance, the DBpedia instance “<http://dbpedia.org/resource/Cyclades>” has db-prop:population with a value of “119549”, and the Geonames instance “<http://sws.geonames.org/259819/>” also has a predicate that represents population, i.e., geo-onto:population with a value of “119549”. Both of DBpedia and Geonames URIs indicate the place “Cyclades”, but there is no owl:sameAs link between these two URIs.

Therefore, we can find missing links with our Mid-Ontology if there exist predicates from different domains grouped under the same Mid-Ontology class. In our constructed Mid-Ontology, we can find missing links according to mo:birthdate, mo:population, mo:postalcode, etc.

5 Related Work

Some researchers have proposed similar ideas about constructing an intermediate-layer ontology and reducing data sets. For instance, the authors in [14] automatically generated alignments from linked instances in the Linked Data, especially geospatial, zoology, and genetics data sources. During data preprocessing, they only considered linked instances and removed unimportant properties to reduce search space. Their algorithm discovers equivalent and subsumption relations and models one Linked Data source through ontology alignment. However, their approach is limited to specific domains of data sets, while our approach can be applied to data sets from any domain.

Some researchers have proposed the construction of an intermediate-level ontology to connect general ontologies and an upper ontology. The authors in [5] introduced a method to construct an intermediate-level ontology by mapping an upper ontology, PROTON, to the concepts of DBpedia, Geonames, and Freebase described in the FactForge. They enriched the upper ontology by adding 166

new classes and 73 new properties; the resulting ontology was a large one. The end-users have to understand the large ontology to construct SPARQL queries.

The authors in [6] analyzed the basic properties of SameAs network, Pay-Level-Domain network, and Class-Level Similarity network. They analyzed the Pay-Level-Domain network to examine how data publishers are connected, by comparing the five most frequent types. However, when only frequent types are considered, it is not possible to determine exactly how data are connected.

In contrast to the approaches adopted in the related research described above, our Mid-Ontology learning approach is aimed at constructing a small ontology by integrating predicates and can be directly applied to semantic web applications. With our Mid-Ontology, we can easily determine the kinds of things that are linked together by observing the characteristics of the integrated predicates. Furthermore, user interaction is not needed for the Mid-Ontology construction, and our approach is applicable to linked data sets as long as they are connected.

The drawbacks of our approach are that a hub data set is necessary for extracting linked instances and that related predicates cannot be found if data sets are not directly connected in the LOD cloud. One possible solution is to investigate on the connected components [6] in the LOD cloud, by applying clustering technology, and by analyzing the contents of the connected components.

6 Conclusion and Future Work

In this paper, we proposed a Mid-Ontology learning approach that involves the use of Linked Open Data and can help semantic web application developers to integrate diverse ontology schema without learning the entire ontology schema. The main procedures of our approach are data collection process, the ontology predicate grouping process, and Mid-Ontology construction process. The predicate grouping algorithm applied lexical similarity matching to collect similar predicates and implemented the relation extraction method to refine predicate groups. Our approach can automatically extract the most related predicates between linked data sets, and integrate them in the Mid-Ontology. Experimental results show that the amount of data can be dramatically reduced in the data collection phase and that the accuracy of the Mid-Ontology can be significantly improved by the group pruning and refining processes. Furthermore, with the Mid-Ontology, potential information can be effectively retrieved in a simple SPARQL query.

In future work, we will apply our approach to the Billion Triple Challenge (BTC) data set, which is collected by crawling real-world linked data. Since our current approach only considers data sets directly linked with a hub data set, it cannot extract relations by crawling links at more than one depth. We plan to extend our approach, so that it can crawl at two or three depths of links to collect interesting information from the linked data.

References

1. Auer, S., Lehmann, J.: Creating knowledge out of interlinked data. *Semantic Web* 1(1-2), 97–104 (2010)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. *International Journal on Semantic Web and Information Systems* 5(3), 1–22 (2009)
3. Choi, N., Song, I.-Y., Han, H.: A survey on ontology mapping. *ACM SIGMOD Record* 35, 34–41 (2006)
4. Cimiano, P.: *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc. (2006)
5. Damova, M., Kiryakov, A., Simov, K., Petrov, S.: Mapping the central lod ontologies to proton upper-level ontology. In: *Proceedings of the Fifth International Workshop on Ontology Matching*, pp. 61–72 (2010)
6. Ding, L., Shinavier, J., Shangguan, Z., McGuinness, D.L.: Sameas networks and beyond: Analyzing deployment status and implications of owl: sameas in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 145–160. Springer, Heidelberg (2010)
7. Drumond, L., Girardi, R.: A survey of ontology learning procedures. In: *Proceedings of the Third Workshop on Ontologies and their Applications* (2008)
8. Erling, O., Mikhailov, I.: Virtuoso: Rdf support in a native rdbms. In: *Semantic Web Information Management*, pp. 501–519 (2009)
9. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
10. Fellbaum, C. (ed.): *WordNet: An Electronic Lexical Database*. MIT Press (1998)
11. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs isn't the same: An analysis of identity in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 305–320. Springer, Heidelberg (2010)
12. Heath, T., Bizer, C.: *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool (2011)
13. Ichise, R.: An analysis of multiple similarity measures for ontology mapping problem. *International Journal of Semantic Computing* 4(1), 103–122 (2010)
14. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and building ontologies of linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 598–614. Springer, Heidelberg (2010)
15. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet:similarity: Measuring the relatedness of concepts. In: *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pp. 1024–1025 (2004)
16. Porter, M.F.: An algorithm for suffix stripping. In: *Readings in Information Retrieval*, pp. 313–316 (1997)
17. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res.* 13(11), 2498–2504 (2003)
18. Zhou, L.: Ontology learning: state of the art and open issues. *Information Technology and Management* 8, 241–252 (2007)

An Ontological Formulation and an OPM Profile for Causality in Planning Applications

Irene Celino and Daniele Dell'Aglio

CEFRIEL – Politecnico of Milano, via Fucini 2, 20133 Milano, Italy
{irene.celino,daniele.dellaglio}@cefriel.it

Abstract. In this paper, we propose an ontological formulation of the planning domain and its OWL 2 formalization. The proposed meta-model conceptualizes planning rules and actions and the *causality* between them. We also show that our planning metamodel can be seen as a relevant scenario of the Open Provenance Model (OPM) and we define our planning OPM profile.

This ontological representation is then exploited to define automated means for the verification of correctness and consistency of a planning domain model. We claim that Semantic Web technologies can provide an effective solution to this important – and often underestimated – problem for planning applications.

1 Introduction and Motivations

Planning is a branch of AI dealing with the automated creation of plans, i.e. a sequence of actions that, starting from an initial state of the “world”, leads to a state that meets some desired goals. Finding a suitable plan among all possible sequences of actions is a complex problem that requires a detailed comprehension of the world, the agents acting in this world, the possible actions, the causal relationships between those actions, etc. The specification of such knowledge about the planning problem is usually indicated as *domain theory*.

When formalizing a domain theory in some representation format, it is therefore key to use an enough expressive language and to assure the correct and consistent representation of the world, so that the planning algorithms can operate and compute the optimal plans. Several approaches and languages have been proposed in literature to formalize the planning problem, including the well-known STRIPS [1] and the more recently standardized PDDL language [2]. Those representation formats and languages are also employed to check the consistency of the generated plans (e.g., to recognize an inconsistency in a plan if two incompatible events are applied simultaneously).

Checking the coherence and rationality of the domain theory itself, on the other hand, is a task that is usually discarded or delegated to the modeller that formalize the planning problem. When modelling the actions and their causal relationships, for example, it is indeed important to validate the model, e.g. by checking that all modelled states of the world are “reachable” in a sequence of actions. Especially when the domain theory is large and complex, this kind of validation becomes of utmost importance.

In this paper, we illustrate how the Semantic Web can be successfully employed to represent and reason upon such planning domain models. We propose an ontological formalization of a planning metamodel, i.e. a domain-independent specification of the planning problem, on top of which different domain theories can be developed to represent different planning situations. In doing so, we also show that the planning metamodel can be seen as a relevant application case of the Open Provenance Model (OPM) [3]; we provide an OPM profile by mapping the planning metamodel to one of the ontological formulations of OPM, namely the Open Provenance Model Vocabulary (OPMV) [4]. The ontological formulation can then be employed to define a set of rules to check the consistency of a domain theory that uses the planning metamodel; this is aimed at giving modellers a means to check and support their modelling task.

The remainder of the paper is structured as follows. Section 2 introduces the ontological formulation of the planning problem, together with its representation in OWL 2 [5] and the explanation of our modelling choices; Section 3 explains the planning metamodel as an OPM Profile and its semantics; Section 4 gives some examples of automated checks of the causality in domain theories defined on the planning metamodel. We present a complex scenario of Simulation Learning for Crisis Management in Section 5 in which we apply our metamodel; related work is illustrated in Section 6 while Section 7 concludes the paper and gives some hints on possible extensions of this work.

2 The Ontological Formulation of the Planning Problem

Ontologies are generally used to *assert* statements that are considered true in the modelled world; in planning application, however, dynamics is the predominant dimension, e.g., in a planning domain theory we can include definition of actions that are mutually exclusive and therefore cannot be “asserted” at the same time. Thus it could be considered unusual or unsuitable to use ontologies to represent planning knowledge.

However, our investigation does not deal with the search for the optimal plan, i.e. our formalization does not want to be used within the planning algorithm. Indeed, our purpose is different: through an ontological representation, we aim to predicate on the possible states of the planning world and on the causality of state transitions, i.e. the conditions under which the world state changes. Within the planning metamodel we want to “statically” represent the possible “dynamics” of the world.

In this section, we explain our modelling of the planning problem and of its causality definition. We reuse as much as possible the terminology used in planning literature [6][12]; some part of the vocabulary can specifically refer to the terminology used in timeline-based planning [7][8].

2.1 Planning Metamodel

As introduced above, a *domain theory* defines the planning problem, in terms of the agents, their possible actions and the causality between them. The *planning*

problem consists in identifying a set of relevant aspects whose (temporal) evolutions need to be controlled to obtain a desired behaviour or to attain a specific goal. In the following, we introduce the main primitives of a domain theory.

Components represent logical or physical subsystems whose properties may vary in time, thus they are the relevant variables to be planned for. Components evolve over time; events can happen and decisions can be taken on components: those events and decisions alter the components evolution. Components can either be intelligent *agents*, i.e. the characters involved in the planning – both human and artificial ones –, or other entities, such as buildings, weather, rivers, which are *resources* to be controlled in the planning.

We can classify components on the basis of their *control*. The temporal behaviour of *controllable components* is decided by the planner; those components define the search space for the planning problem, and their evolution ultimately represent the problem solution. Conversely, the evolution of *uncontrollable components* is given to the planner as input; those components are imposed over time thus they can only be observed: they can be seen as additional/external data and constraints for the planning problem.

Actions are temporally tagged events. They are always related to a component and they represent events or decisions that alter the component behaviour, causing the transition of the component between two possible states. Usually the term *decision* is used in relation to an uncontrollable component, while the term *event* is preferred to indicate an action determined by the planner that happens to a controllable component; in the following we will use only the term action for the sake of simplicity. Actions always refer to some component and are characterized by some *parameters*. The conditions under which actions can occur are regulated by the domain theory.

The domain theory defines *actions’ causality*, i.e. the combinations of components behaviours that are acceptable with respect to actions happening on other components. We represent such causality by means of *planning rules*, also called *synchronizations*. A planning rule specifies the “consequences” of actions, i.e. it states the relation between two or more actions. In their generic form, planning rules relate how an action (also called reference action) can activate one or more actions (target actions) if some conditions or constraints on components and/or on action’s parameters are verified.

Rule conditions are the constraints defined within a planning rule. In their generic form, rule conditions impose requirements on the actions involved in a synchronization, thus they can be represented as relations between action parameters. A special case of constraint, very relevant for the timeline-based planning problem, is the *temporal condition*: it represents a temporal constraint on an action (e.g., an action must have a duration of 10 minutes) or on the temporal sequence of two actions (e.g., an action must start only when another action finishes). Allen’s Interval Algebra [9] is used to formalize temporal conditions.

Let’s consider the following example of planning rule: if an ambulance with a patient arrives at a hospital and the latter has remaining capacity, then the patient can be admitted and the number of available beds is decreased by one.

In the example we identify two components, the ambulance and the hospital; a reference action (the ambulance carrying the patient to the hospital), a target action (the hospital reserves a bed to the patient) and a condition (there are available beds in the hospital). It is important to note that a reference action can enable the activation of one or more actions (if the rule conditions are satisfied).

2.2 An OWL 2 Formulation of the Planning Metamodel

We represent the conceptualization defined in the previous section in OWL 2 [5]. The complete definition of this ontology is available at <http://swa.cefriel.it/ontologies/tplanning>¹; in this section, we explain some of the modelling choices.

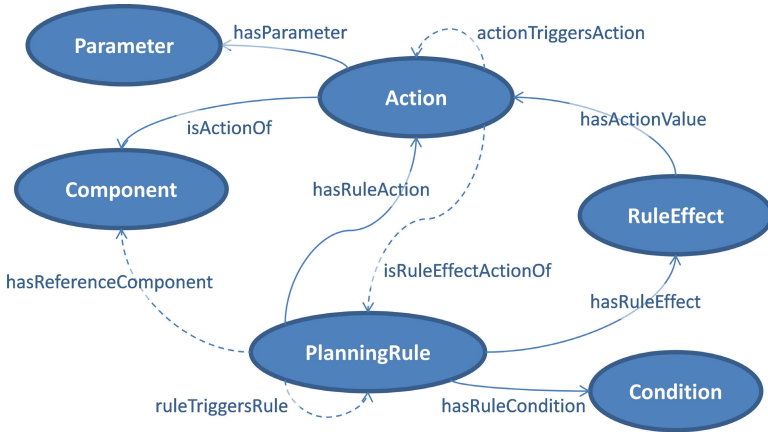


Fig. 1. A graphical representation of the main entities in our planning ontology

Figure 1 illustrates the concepts introduced above with the properties that interrelate them. **Actions** refer to **Components** and are described by **Parameters**; a **PlanningRule** puts in relation some causal action (related via the **hasRuleAction** property) with some **RuleEffect** (which, in turn, can include “target” actions); the causality constraints in a planning rule are defined by **Conditions**.

With the use of property chain axioms [5], we introduce also some derived properties. For example, starting from the basic properties defining a planning rule (indicated by a solid line in Figure 1), we infer other properties that identify the planning causality (indicated by a dashed line), as follows:

$$\begin{aligned}
 \text{tpl:hasReferenceComponent} &\sqsubseteq \text{tpl:hasRuleAction} \circ \text{tpl:isActionOf} \\
 \text{tpl:isRuleEffectActionOf} &\sqsubseteq \text{tpl:hasActionvalue}^- \circ \text{tpl:hasRuleEffect}^- \\
 \text{tpl:actionTriggersAction} &\sqsubseteq \text{tpl:hasRuleAction}^- \circ \text{tpl:isRuleEffectActionOf}^- \\
 \text{tpl:ruleTriggersRule} &\sqsubseteq \text{tpl:isRuleEffectActionOf}^- \circ \text{tpl:hasRuleAction}^-
 \end{aligned}$$

¹ In the following, the **tpl** prefix (*temporal-planning*) will be used to indicate terms from this ontology. We omit the **tpl** prefix in the figures for sake of readability.

The last two properties indicates the *causality between actions* (an action triggers another action if there is a planning rule which is activated by the first action and has the second action as effect of its activation) and the *causality between rules* (a rule triggers another rule if there is an action which is an effect of the first rule and activates the second rule).

2.3 Modelling Conditions in Planning Rules

Most part of the planning causality, however, is usually included in the rule conditions. As introduced above, a condition imposes restrictions on the involved actions, their components or their parameters. Conditions can be assignments (in the example above, the hospital capacity is decreased by one), constraints (e.g., the hospital capacity must be greater or equal to one) or temporal conditions (e.g., the ambulance must arrive at the hospital before the patient can be admitted). Figure 2 illustrates the different types of rule conditions.

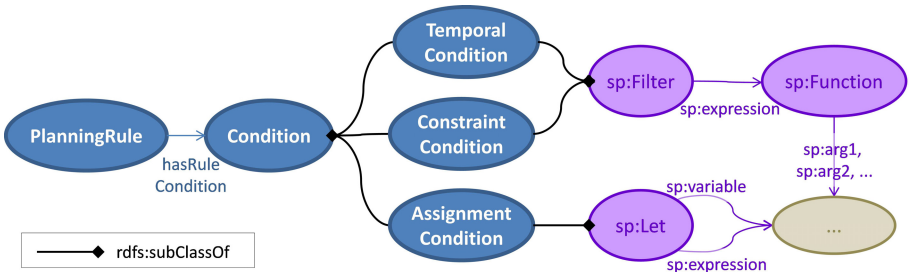


Fig. 2. Modelling of conditions in planning rules and their relation to SPARQL clauses (FILTER and LET) as modelled in SPIN

In the SPARQL query language [10], FILTER clauses are used to express constraints on query variables; moreover, even if not part of the official specification, some SPARQL extensions also define LET clauses to express assignments on query variables². It was therefore natural to us to assimilate planning rule conditions to SPARQL FILTER and LET clauses. For their modelling, we reused the well-known SPARQL Inferencing Notation [11], also known as SPIN (see again Figure 2). SPIN allows to model SPARQL queries in RDF, thus enabling to define query patterns together with the vocabulary or ontology they refer to. In our case, SPIN allows us to model the conditions that affect planning actions and their parameters together with the definition of the actions themselves.

SPIN already defines a number of different constraint types, called *functions*; for example, SPIN models Boolean functions (equal, not equal, greater than, etc.), mathematical functions (addition, subtraction, etc.) and operations on strings (e.g. regular expressions). Since in planning the time dimension has an important role in the causality definition, we extended this SPIN modelling of

² For example, the popular Jena framework implements the LET clause in the ARQ library.

functions to include *temporal relations* as defined by Allen [9]. Figure 3 shows our modelling: we defined a `TemporalFunction` for each Allen relation (before, after, temporal-equal, etc.); each function can be further described by one or more temporal “ranges”, that indicate the intervals characterizing the relation (e.g. action A starts 5 to 10 minutes after action B ends).

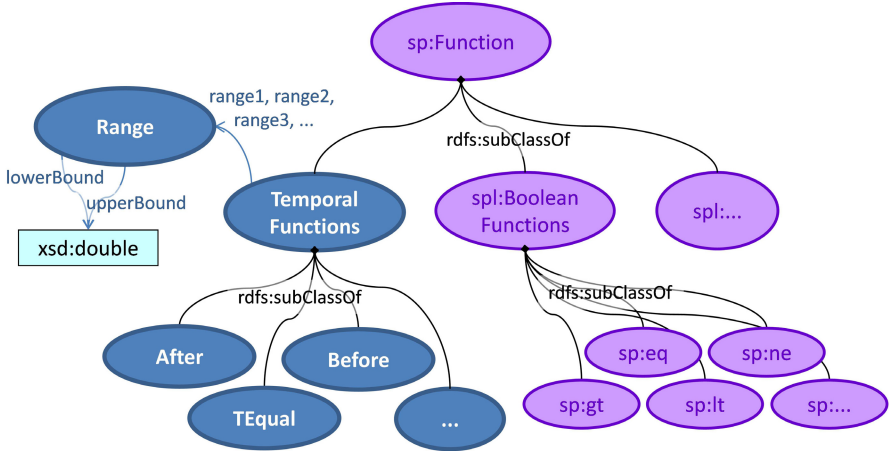


Fig. 3. Extension to the SPIN modelling of functions with Allen’s temporal relations

3 The Planning Causality as an Open Provenance Model

The Open Provenance Model Specification [3] was designed to meet several requirements, among which defining provenance in a technology-agnostic manner, supporting a digital representation of provenance and defining a core set of rules that identify the valid inferences that can be made on provenance representations. The basic nodes and edges in OPM are graphically represented in Figure 4.

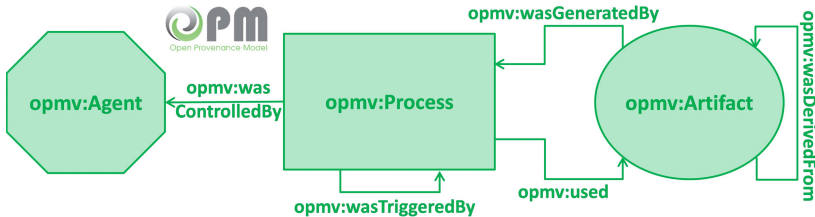


Fig. 4. The basic constituents of a provenance graph according to OPM

According to this specification, an OPM profile is a specialisation of OPM that remains compatible with the semantics of OPM but that defines a best practice or usage guideline for a specific provenance application. The planning meta-model introduced in Section 2 can be seen as an OPM profile, in that the

causality between actions by means of planning rules is a way to represent the actions “derivation”, “use” and “generation” during the planning process. It is worth noting that, while OPM is usually employed to trace provenance in *past* process executions, in the planning case the actions causality represents a *potential future* provenance information (indeed, the actions defined in a planning domain model represent action “templates” rather than “instances” of actions).

In this section, we define our Planning OPM profile and its formalization in a mapping between our planning metamodel and the Open Provenance Model Vocabulary [4], one of the ontological formulations of OPM. We also express the OPM completion rules and inferences, as defined in [3], and we explain how we relaxed some of the OPM constraints to better capture the concept of “provenance” in planning. We are aware that the formalization provided in the following has stronger assertions than those in OPM; still, we confine those restrictions to our OPM profile, in which they are meaningful and valid, and do not intend them as of general value outside our profile.

3.1 Mapping the Planning Metamodel to OPMV

Figure 5 graphically shows our planning OPM profile³ (cf. with Figure 1). The planning rules are our main *processes*, the components are our *agents* and the actions are the *artifacts* of the planning. Thus, with reference to the OPM nodes and edges as defined in the OPM Vocabulary [4], in our planning profile we assert that:

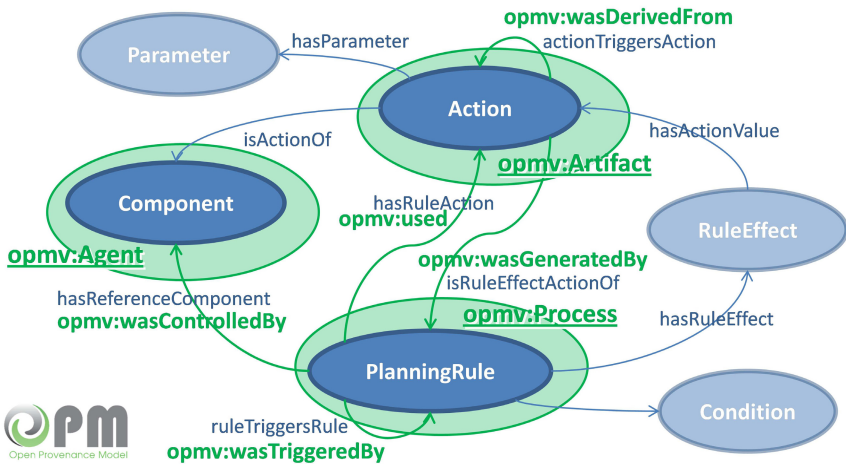


Fig. 5. The planning metamodel as an OPM Profile

³ The complete definition of the planning OPM profile is also available on the Web at <http://swa.cefriel.it/ontologies/causality-provenance>

$$\begin{aligned} \text{tpl:Component} &\sqsubseteq \text{opmv:Agent} \\ \text{tpl:PlanningRule} &\sqsubseteq \text{opmv:Process} \\ \text{tpl:Action} &\sqsubseteq \text{opmv:Artifact} \end{aligned}$$

In fact, in a similar way to what happens in OPM, planning rules are on the one hand related and influenced by the components, and on the other hand they refer to actions in input and they “produce” sets of actions as output.

With respect to the definition of properties, we map the planning predicates to the relations defined in OPM, as follows:

$$\begin{aligned} \text{tpl:hasRuleAction} &\sqsubseteq \text{opmv:used} \\ \text{tpl:isRuleEffectActionOf} &\sqsubseteq \text{opmv:wasGeneratedBy} \\ \text{tpl:hasReferenceComponent} &\sqsubseteq \text{opmv:wasControlledBy} \\ \text{tpl:actionTriggersAction} &\sqsubseteq \text{opmv:wasDerivedFrom} \\ \text{tpl:ruleTriggersRule} &\sqsubseteq \text{opmv:wasTriggeredBy} \end{aligned}$$

3.2 Completion Rules and Inferences

The definition of the planning OPM profile enables a set of completion rules and inferences [3], i.e. a set of rules that allows to derive further provenance relationships between processes and artifacts in a provenance graph. In our case, applying those rules to our planning OPM profile can help in inferring indirect or implicit causal relationships between actions and planning rules defined in a planning domain theory.

The OPM completion rules can be summarized as follows:

$$\begin{aligned} \text{opmv:wasTriggeredBy} &\equiv \text{opmv:used} \circ \text{opmv:wasGeneratedBy} \\ \text{opmv:wasDerivedFrom} &\sqsupseteq \text{opmv:wasGeneratedBy} \circ \text{opmv:used} \end{aligned}$$

The first line above formalizes the so-called *artifact introduction and elimination* completion rule: a process was triggered by another process if and only if an artifact used by the first process was generated by the second one.

The second line expresses the *process introduction* completion rule: if an artifact was derived from another artifact, there must have been a process that generated the first artifact and used the second one. OPM explicitly states that in general the converse rule (process elimination) does not hold, because without any internal knowledge of the process, it is not possible to assert an actual dependency between the two artifacts. However, OPM also offer the possibility to relax this constraint within a specific OPM profile; in our case, the processes are always planning rules which – by definition – express the causality dependency between actions (i.e., artifacts). Thus, in our planning OPM profile, we state that both the *process introduction* and the *process elimination* completion rules hold, replacing the second line of the axioms above with the following one:

$$\text{opmv:wasDerivedFrom} \equiv \text{opmv:wasGeneratedBy} \circ \text{opmv:used}$$

Thus, because of the mapping we defined above, in our planning OPM profile, the following completion rules hold:

$$\begin{aligned} \text{tpl:ruleTriggersRule} &\equiv \text{tpl:hasRuleAction} \circ \text{tpl:isRuleEffectActionOf} \\ \text{tpl:actionTriggersAction} &\equiv \text{tpl:isRuleEffectActionOf} \circ \text{tpl:hasRuleAction} \end{aligned}$$

Those completion rules let us derive the causal relationships between any couple of actions or processes in a planning domain theory.

Additionally, OPM define *multi-step inferences* to account for indirect causes of an artifact or a process as effect of multiple steps. Specifically, OPM defines the multi-step version of `wasDerivedFrom`, `used`, `wasGeneratedBy` and `wasTriggeredBy` edges (which all depend on the transitive closure of the `wasDerivedFrom` relation). We can express those multi-step inferences as follows:

$$\begin{aligned} &\textit{Transitive}(\text{opmv:wasDerivedFrom}) \\ \text{opmv:used} &\sqsubseteq \text{opmv:used} \circ \text{opmv:wasDerivedFrom} \\ \text{opmv:wasGeneratedBy} &\sqsubseteq \text{opmv:wasDerivedFrom} \circ \text{opmv:wasGeneratedBy} \\ \text{opmv:wasTriggeredBy} &\sqsubseteq \text{opmv:used} \circ \text{opmv:wasDerivedFrom} \\ &\quad \circ \text{opmv:wasGeneratedBy} \end{aligned}$$

Thanks to the mapping between our planning metamodel and the OPMV, the following multi-step inference rules hold:

$$\begin{aligned} &\textit{Transitive}(\text{tpl:actionTriggersAction}) \\ \text{tpl:hasRuleAction} &\sqsubseteq \text{tpl:hasRuleAction} \circ \text{tpl:actionTriggersAction} \\ \text{tpl:isRuleEffectActionOf} &\sqsubseteq \text{tpl:actionTriggersAction} \circ \text{tpl:isRuleEffectActionOf} \\ \text{tpl:ruleTriggersRule} &\sqsubseteq \text{tpl:hasRuleAction} \circ \text{tpl:actionTriggersAction} \\ &\quad \circ \text{tpl:isRuleEffectActionOf} \end{aligned}$$

Again, those inferences can be employed to derive indirect causal relationships between actions and processes.

Summing up, the definition of an OPM profile for our planning metamodel allows us to reuse the provenance primitives to analyse and infer new knowledge about the causality between actions and planning rules in a domain theory. In the following section, we will show how the above inferences can help in checking the planning domain modelling.

4 Automated Checking of Causality in Planning Models

Whereas planning software is supposed to conform to a solution search algorithm, a planning domain model is supposed to capture a piece of reality and so it requires its own acceptance criteria and tests. Using our ontological formulation of the planning metamodel, we can devise guidelines and tests to capture different levels of consistency for domain models. We would like to stress that, while the typical concerns of a *planner* are efficiency, correctness and completeness of the planning algorithms, here we concentrate on the *modeller’s* point of view, whose concerns include validation, expressive power and maintenance of a domain model [12].

In this section, we give some examples of controls that are enabled by our ontological formulation of the planning metamodel and by its mapping to OPM.

Those controls can be successfully employed to support the modeller’s task, i.e. to verify the correctness and consistency of the planning domain model. Without claiming to be exhaustive, in the following we formalize some of those controls and we explain how those checks can be easily implemented via SPARQL 1.1 [10] queries.

4.1 Model Completeness and Action Reachability

As outlined in [12], usually modellers start from the identification of the relevant objects (planning components), then continue with the definition of their relations (actions on components), afterwards they analyse the possible world states and their transitions (planning rules), and so on. When modelling a large or complex planning domain, the number of introduced entities can be very high and verifying the consistency and meaningfulness of the whole model can become complex.

We define a planning domain model as *complete* when all modelled components are involved in some action and all modelled actions are involved in some planning rule. It is worth noting that “orphan” components or actions does not make the domain theory inconsistent per se, but they can suggest an unfinished or lacking modelling. Setting a control to check model completeness is aimed to support the modeller to identify potential lacks or shortcomings in the domain definition.

Component and actions making the model incomplete can be defined as:

$$\begin{aligned} \text{tpl:OrphanComponent} &\sqsubseteq \text{tpl:Component} \sqcap \forall \text{tpl:isActionOf} \neg .\perp \\ \text{tpl:OrphanAction} &\sqsubseteq \text{tpl:Action} \sqcap \forall \text{tpl:isRuleEffectActionOf} .\perp \\ &\quad \sqcap \forall \text{tpl:hasRuleAction} \neg .\perp \end{aligned}$$

Checking the completeness of the model therefore means that the above defined classes have no instances in the planning domain model. Conversely, if some “orphans” are found, those are the domain entities the modeller should look at to identify potential pitfalls. Assuming a closed world assumption, we can implement this check simply by querying the planning domain model expressed in our planning metamodel with the following SPARQL 1.1 [10] queries:

```
SELECT ?component
WHERE {
  ?component a tpl:Component .
  FILTER NOT EXISTS { ?action tpl:isActionOf ?component . }
}
```

```
SELECT ?action
WHERE {
  ?action a tpl:Action .
  ?rule a tpl:PlanningRule .
  FILTER NOT EXISTS { ?rule tpl:hasRuleAction ?action . }
  FILTER NOT EXISTS { ?action tpl:isRuleEffectActionOf ?rule . }
}
```

Another property of a domain model a modeller could wish to check is *action reachability*. We define an action reachable if there is at least a planning rule which causes that action, i.e. which has that action as effect. Again, this kind of control is aimed at supporting the modeller to identify potentially incomplete entity definition: if an action is introduced in a domain, it is very likely that the modeller considered it possible to generate that action in some plan. Nevertheless, it is also perfectly reasonable that a defined action appears only as the condition to fire a planning rule and not as its effect. This can happen when the action refers to an uncontrollable component (cf. Section 2.1): in this case, since the action activation depends on an agent external to any generated plan, there is no need for a planning rule to cause that action in the domain model.

An unreachable action can be defined as follows:

$$\text{tpl:UnreachableAction} \sqsubseteq \text{tpl:Action} \sqcap \forall \text{tpl:isRuleEffectActionOf} . \perp \\ \sqcap \forall \text{tpl:isActionOf} . \text{tpl:ControllableComponent}$$

Following the definition of the planning OPM profile (cf. Section 3.1), the class above can be also expressed as follows:

$$\text{tpl:UnreachableAction} \sqsubseteq \text{opmv:Artifact} \sqcap \forall \text{opmv:wasGeneratedBy} . \perp \\ \sqcap \forall \text{opmv:used} . \text{opmv:wasControlledBy} . \\ \text{tpl:ControllableComponent}$$

Thanks to OPM multi-step inferences (cf. Section 3.2), this definition includes all possible provenance paths that connect actions (artifacts) with planning rules (processes).

Again, checking the action reachability in a domain model means verifying that the above defined class has no instances in the domain theory. To identify the unreachable actions, and thus understand the appropriateness of their definition, a modeller can use the following SPARQL 1.1 query (or the respective one that makes use of OPMV properties as per the planning OPM profile):

```
SELECT ?action
WHERE {
  ?action a tpl:Action.
  FILTER NOT EXISTS {
    ?rule tpl:hasRuleEffect/tpl:hasActionValue ?action .
  }
  FILTER NOT EXISTS {
    ?action tpl:isActionOf [ a tpl:UncontrollableComponent ] .
  }
}
```

4.2 Constraint Checking

As illustrated in Section 2, defining a planning rule includes also the introduction of a set of conditions, i.e. constraints on the rule activation. It is often the case that most of the “rational” and complexity of a planning domain theory lies

in its rules' conditions. Therefore, verifying the consistency of a domain model means checking the satisfiability of the constraints defined in the planning rules.

For example, let's say that we want to capture the rules of an educational institution. A planning rule could say that, when a new student arrives and asks to join the school, if he/she is above legal age (condition), the school can enrol him/her. Another planning rule could state that, if a student is hurt and he/she is below legal age (condition), the school should inform his/her parents. The previous two rules are both reasonable and the first one is a sort of pre-condition for the second one (people are considered students only after their enrolment); still, it is apparent that the second rule will never be triggered, since no student of this institution can be under legal age. Thus, a modeller has to check all rules' conditions to understand if they ever apply.

In our planning metamodel, we decided to assimilate rule conditions to SPARQL FILTER and LET clauses (cf. Section 2.3). This modelling choice comes of help also for constraint checking: starting from the condition definition, it is natural to create SPARQL queries with those clauses, in case combining different conditions; executing those queries on the planning domain model helps the modeller to identify inconsistencies and potential modelling mistakes.

5 Applying Our Approach to Simulation Learning

Simulation Learning is a kind of training aimed to improve soft skills [13]. Simulation Learning systems generally re-create near-real environments for training sessions, in which learners are subject to stimuli: they have to learn how to deal with the simulated situation and how to react to it. Such simulations need to be effective and engaging, so that the learners do not simply memorise notions, but they actively and permanently acquire skills, practice and knowledge. In this context, simulation sessions can be generated using planning technology from a learning domain model.

The Pandora project⁴ aims to provide a platform for Crisis Management simulation learning, providing a near-real training environment at affordable cost. The Pandora platform [14] makes use of Timeline-based Planning technologies [7,8] to plan the simulation sessions and it exploits the ontological framework explained in this paper to represent the actions causality in the crisis simulation scenario. To this end, we specialized the planning ontology introduced in Section 2.2 with the relevant entities of Crisis Management training, thus specifying the Pandora ontology⁵ and we set up a Linked Data-empowered Knowledge Base [15] to manage the domain theory definition.

Modelling a Crisis Management scenario – or any Simulation Learning scenario – means creating the crisis events to stimulate trainees (e.g., a street is flooded, a hospital electricity becomes scarce for a black-out) and to plan for different storyboard evolutions in response to trainees' actions (e.g., depending on the Crisis Managers decisions, the crisis situation becomes more or less critical).

⁴ Cf. <http://www.pandoraproject.eu/>

⁵ Cf. <http://swa.cefriel.it/ontologies/pandora>

We applied the approach outlined in Section 4 to support the modeller in verifying the domain theory: while not substituting the manual intervention, this method proved to be a useful means to detect potential problems, before checking the plans generated by the planning algorithms. Indeed, the adoption of our ontological metamodel and its verification via SPARQL queries (both the generic completeness and reachability controls from Section 4.1 and some manually-defined and domain-dependent queries built on top of the Pandora rules conditions as described in Section 2.3) allowed the modeller to identify missing definitions and unreachable actions. Though preliminary and qualitative, this evaluation makes us believe in the usefulness and efficacy of our proposed approach. We are currently investigating an automated way to generate the SPARQL queries needed for constraint checking, starting from the planning rule conditions’ definition.

6 Related Work

Different works in literature dealt with the ontological representation of planning and scheduling knowledge. Early models like [16] and [17] were aimed at representing planning tasks and planning problems; however, they were limited in scope and did not considered all the relevant entities of the planning world. The most comprehensive planning ontology so far is described in [18]: it is the first formalization that includes the temporal dimension and the notion of agents. Still, that model was aimed at proposing an operational specification of the planning problem with a set of “executable” definitions.

In contrast, our planning metamodel is not directly oriented to the search for plans; with our formalization we aim at supporting the modelling of planning domain theories, by identifying potential problems prior to the execution of the planning algorithms. Moreover, our metamodel is more detailed than the ontology described in [18], because we make the constraints on the planning rules “first-class citizens” of our conceptualization. This `Condition` concept not only allows for the declarative definition of a complete domain theory, but it also provides a mechanism to reuse and share constraints between different planning rules. Furthermore, we underscored the temporal aspect of planning by introducing the `TemporalCondition` primitive and by modelling the `TemporalFunctions` hierarchy.

In the planning community, a dedicated workshop [19] was organized to discuss “the role of ontologies in Planning and Scheduling”. The result was that ontological languages like RDFS and OWL are more expressive than the ones in the planning field, for example because of the Open World Assumption; however, they cannot be “directly applied” in planning systems because they are usually employed to represent static knowledge. While we agree with this statement, our investigation is oriented precisely in the possible cooperation between ontologies and planning on their “boundary”: we can say that our metamodel statically captures the dynamics of planning. Thus we are convinced that ontology-based knowledge representation can bring benefits to current planning technologies [20].

Finally, a note about the languages used in planning: PDDL [2] is currently the most popular, even if its iterative standardization did not prevent the spread of a multitude of different dialects. While a comparison of expressivity between PDDL and ontological languages is out of scope of this paper⁶, we would like to stress again that we propose the use of an ontological formalization *outside* the search process to find a solution to the planning problem. Our formalization and causality checks are complementary to the consistency and validation of the plans generated in the solution space.

7 Conclusions

In this paper, we presented our approach to formalize the planning primitives and their relations with an ontology. We believe that this is a good example of interplay between Semantic Web and Planning technologies [20], since knowledge representation and reasoning – even in simple forms as we did in this work – can be of great help during the planning modelling: the automatic checking of the domain theory characteristics (e.g., completeness and reachability as defined in this paper) supports the modellers’ job, because it helps them to identify potential problems in their modelling even before checking the consistency of the generated plans. Moreover, because of the spread and success of knowledge sharing on the Web, including the Linked Data movement, planning modelling can be simplified or reduced by reusing and linking to pre-existing datasets, as we illustrated in a previous work [15].

To this end, we introduced an OWL 2 representation of this planning meta-model and its formulation as an Open Provenance Model Profile, through a mapping between our metamodel and the OPM Vocabulary. The reason why we chose to adopt OPM is two-fold. On the one hand, the *provenance* abstractions are very similar to the ones used in *causality* models, like those we have in our planning metamodel. On the other hand, the completion rules and inferences defined in OPM offer a simple yet powerful means to perform checks; this is why we leveraged those reasoning means to formulate our model checks.

Our future works are oriented in two directions: extending the causality checks introduced in this paper and applying analysis and mining on the actual “executions” of plans generated on the basis of domain theories represented in our metamodel. With regards to the former, we will take into account some more characteristics already modelled in the planning ontology introduced in Section 2 and enhance accordingly the causality controls defined in Section 4.

Regarding the latter, once a domain theory has been modelled and used to generate plans, those plans can be “executed” in planning applications (e.g., in the simulation learning scenario introduced in Section 5, when training sessions take place); the recording of those executions can be seen as “streams” of events, i.e. assertions that are valid in a specific time-frame. We believe that we can exploit those time-stamped assertions to refine the causality modelling: e.g., comparing the actual streams of happened events in different sessions, we can

⁶ A discussion of the possible interplay between PDDL and Datalog is offered in [21].

identify the most frequent patterns of events which have some causality aspect as well as the least frequent plan options; those can be interesting hints for the planning modeller to improve the domain theory. Dealing with plan executions means taking in consideration the *temporal* dimension, thus we need proper approaches; to this end, we aim at employing Stream Reasoning technologies [22], also by following other experiences in ex-post provenance analysis of workflow executions as in [23].

Acknowledgments. This research is partially funded by the EU PANDORA project (FP7-ICT-2007-1-225387). We would like to thank the project partner for their collaboration.

References

1. Fikes, R., Nilsson, N.: STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2, 189–208 (1971)
2. Gerevini, A., Long, D.: Plan Constraints and Preferences in PDDL3. Technical report, R.T. 2005-08-47, Dipartimento di Elettronica per l'Automazione, Università degli Studi di Brescia (2005)
3. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., den Bussche, J.V.: The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems* (2010)
4. Zhao, J.: Open Provenance Model Vocabulary Specification (2010), <http://purl.org/net/opmv/ns>
5. Hitzler, P., Kroetzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer. W3C Recommendation (October 27, 2009), <http://www.w3.org/TR/owl2-primer/>
6. Russel, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Pearson Education Inc. (2003)
7. Cesta, A., Fratini, S.: The timeline representation framework as a planning and scheduling software development environment. In: 27th Workshop of the UK Planning and Scheduling SIG (2008)
8. Cesta, A., Cortellessa, G., Fratini, S., Oddi, A.: Developing an end-to-end planning application from a timeline representation framework. In: 21st Applications of Artificial Intelligence Conference (2009)
9. Allen, J.F.: Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11), 832–843 (1983)
10. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Working Draft (2011), <http://www.w3.org/TR/sparql11-query/>
11. Knublauch, H.: SPIN Modeling Vocabulary (October 20, 2009), <http://spinrdf.org/spin.html>
12. Mccluskey, T.L., Porteous, J.: Engineering and Compiling Planning Domain Models to Promote Validity and Efficiency. *Artificial Intelligence* 95, 1–65 (2000)
13. Aldrich, C.: *Simulations and the Future of Learning: An Innovative (and Perhaps Revolutionary) Approach to e-Learning*. Pfeiffer (2003)

14. Bernardi, G., Cesta, A., Coraci, L., Cortellessa, G., De Benedictis, R., Mohier, F., Polutnik, J., Vuk, M.: Only Hope remains in the PANDORA's.jar – Pervasive use of planning in a training environment. In: 21st International Conference on Automated Planning and Scheduling, System Demonstrations and Exhibits, Best Demo Award (2011)
15. Celino, I., Dell'Aglia, D.: A Linked Knowledge Base for Simulation Learning. In: Proceedings of the 1st International Workshop on eLearning Approaches for the Linked Data Age (Linked Learning 2011), co-located with the 8th Extended Semantic Web Conference, ESWC 2011 (2011)
16. Mizoguchi, R., Vanwelkenhuysen, J., Ikeda, M.: Task Ontology for Reuse of Problem Solving Knowledge. In: Towards Very Large Knowledge Bases, pp. 46–57. IOS Press (1995)
17. Gil, Y., Blythe, J.: Planet: A sharable and reusable ontology for representing plans. In: The AAAI - Workshop on Representational Issues for Real-World Planning Systems, pp. 28–33 (2000)
18. Rajpathak, D., Motta, E.: An ontological formalization of the planning task. In: International Conference on Formal Ontology in Information Systems (FOIS 2004), pp. 305–316 (2004)
19. Olivares, J.F., Onaindia, E. (eds.): Workshop on the Role of Ontologies in Planning and Scheduling, co-located with the 15th International Conference on Automated Planning and Scheduling, ICAPS 2005 (2005)
20. Celino, I., Dell'Aglia, D., De Benedictis, R., Grilli, S., Cesta, A.: Ontologies, rules and linked data to support crisis managers training. IEEE Learning Technology Newsletter, Special Issue Semantic Web Technologies for Technology Enhanced Learning 13(1) (2011)
21. Thiebaut, S., Hoffmann, J., Nebel, B.: In Defense of PDDL Axioms. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 961–968 (2003)
22. Della Valle, E., Ceri, S., van Harmelen, F., Fensel, D.: It's a Streaming World! Reasoning upon Rapidly Changing Information. IEEE Intelligent Systems 24(6), 83–89 (2009)
23. Miles, S., Wong, S.C., Feng, W., Groth, P., Zauner, K.P., Moreau, L.: Provenance-based validation of e-science experiments. Journal of Web Semantics 5(1), 28–38 (2007)

A New Matchmaking Approach Based on Abductive Conjunctive Query Answering

Jianfeng Du^{1,2}, Shuai Wang¹, Guilin Qi³, Jeff Z. Pan⁴, and Yong Hu¹

¹ Guangdong University of Foreign Studies, Guangzhou 510006, China
jfd@mail.gdufs.edu.cn

² State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China

³ Southeast University, Nanjing 211189, China

⁴ The University of Aberdeen, Aberdeen AB243UE, UK

Abstract. To perform matchmaking in Web-based scenarios where data are often incomplete, we propose an extended conjunctive query answering (CQA) problem, called *abductive CQA problem*, in Description Logic ontologies. Given a consistent ontology and a conjunctive query, the abductive CQA problem computes all *abductive answers* to the query in the ontology. An abductive answer is an answer to the query in some consistent ontology enlarged from the given one by adding a bounded number of individual assertions, where the individual assertions that can be added are confined by user-specified concept or role names. We also propose a new approach to matchmaking based on the abductive CQA semantics, in which offer information is expressed as individual assertions, request information is expressed as conjunctive queries, and matches for a request are defined as abductive answers to a conjunctive query that expresses the request. We propose a sound and complete method for computing all abductive answers to a conjunctive query in an ontology expressed in the Description Logic Program fragment of OWL 2 DL with the Unique Name Assumption. The feasibility of this method is demonstrated by a real-life application, rental matchmaking, which handles requests for renting houses.

1 Introduction

Matchmaking is a useful facility for comparing offers with requests. It determines whether an offer matches a request or whether a request matches an offer, and has been widely used in Web service discovery [18,3,10], skill matching [6], marriage matching [2] and product matching in e-marketplaces [14,15,5,16].

Existing approaches to matchmaking can be divided into two categories, namely syntactic ones and semantic ones. Syntactic approaches usually exploit keyword-based search methods to compare offers with requests. These approaches make little use of the background knowledge and the semantics about offers or requests, and will easily miss right matches or yield wrong matches. Semantic approaches usually use ontologies to formalize offers and requests. Since an ontology provides

the background knowledge and formalizes offers and requests with certain semantics, semantic approaches can make the matchmaking results more sound and complete than syntactic approaches.

The World Wide Web Consortium (W3C) has proposed the Web Ontology Language (OWL), for which the newest version is OWL 2 [11], to model ontologies. OWL is based on a family of formal languages, called *Description Logics* (DLs) [1]. In particular, the most expressive and decidable species of OWL 2, OWL 2 DL, corresponds to the DL *SR_QIQ* [13]. The proposal of OWL has motivated the industry to upgrade many applications to DL-based ones. Recent semantic approaches to matchmaking are also based on DLs. These approaches can be roughly divided into two sorts, described below.

The first sort approaches, such as [3,10,18], exploit a DL ontology to compute semantic distances between offers and requests, where offers and requests are expressed as DL concept descriptions (simply DL concepts). These approaches mainly focus on defining a reasonable distance function between two DL concepts. The primary drawback is that they cannot guarantee that the computed distances adhere to the DL semantics. For example, when an offer matches a request, i.e., the offer is subsumed by the request under the DL semantics, the computed distance may not be zero.

The second sort approaches, such as [14,15,5], also express offers and requests as DL concepts, but they exploit DL inference methods to compute different matches. In the approaches proposed in [14,15], a popular DL inference method, namely concept subsumption checking, is used to compute several kinds of matches. Two kinds that are most related to this work are respectively the *potential match* proposed in [15] and the *intersection match* proposed in [14], where a potential match for a request is an offer matching a portion of the request, while an intersection match for a request is an offer consistent with the request. In the approach proposed in [5], two non-standard DL inference methods, namely concept abduction and concept contraction, are used to compute *possible matches*. A possible match for a request is an offer that gets subsumed by the request after adding some information to the offer (i.e. abduction) and removing some information from the request (i.e. contraction).

All the aforementioned semantic approaches are not easy to scale to real-life applications that involve a large number of offers and requests, because composing the DL concepts for offers and requests is time consuming and laborious. A more practical approach should alleviate human efforts to formalize offers or requests. Hence, we use for reference another approach which is based on conjunctive query answering (CQA). In this approach, offer information is expressed as individual assertions in the back-end ontology, request information is expressed as conjunctive queries posed upon the back-end ontology, and matches for a request are defined as answers to a conjunctive query that expresses the request. This approach enables an efficient way to construct a matchmaking system. That is, a large portion of the back-end ontology, which stores data about offers (i.e. offer information), can be automatically extracted from Web sources using ontology population techniques [4,8], while a very small portion of the

back-end ontology, which stores background knowledge, can be manually built using ontology editors. This approach has been used to solve the fuzzy match-marking problem [16], where the back-end ontology is expressed by a Datalog program with *scoring atoms* that calculate the match degrees. It has also been used to support Semantic Web search [9] for DL back-end ontologies.

However, the CQA based approach is unsuitable in Web-based scenarios where data of the back-end ontology come from the World Wide Web and are often incomplete. Since an offer that is not originally an answer to a conjunctive query can be turned into an answer to after missing data are added, the offer can also be considered as a match. To capture this idea, we propose an extended CQA problem, called *abductive CQA problem*. Given a consistent ontology and a conjunctive query, the abductive CQA problem computes all *abductive answers* to the query in the ontology. An abductive answer is an answer to the query in a certain consistent ontology enlarged from the given one by adding a bounded number of individual assertions, where the individual assertions that can be added are confined by two disjoint sets of concept or role names. The names in the first set are called *abducible predicates*. The possibly added individual assertions can be on abducible predicates only. The names in the second set are called *closed predicates*. The possibly added individual assertions cannot make the enlarged ontology entail any individual assertion that is on closed predicates and is not entailed by the given ontology.

Based on the abducible CQA semantics, we propose a new semantic approach to computing all matches for a given request, where these matches are abductive answers to a conjunctive query that expresses the request. This notion of match is similar to the notion of potential match [15] and the notion of intersection match [14] — all of them regard matches for a request as offers satisfying a certain portion of the request. We also propose a method for computing all abductive answers to a conjunctive query. The method encodes the abductive CQA problem into a Prolog program and solves it with Prolog engines. To ensure that the method is sound and complete, we assume that the given ontology is expressed in the Description Logic Program (DLP) [12] fragment of OWL 2 DL and adopts the Unique Name Assumption [1]. The DLP fragment of OWL 2 DL underpins the OWL 2 RL profile of OWL 2 [11] and is often used in applications that require efficient reasoning facilities. The Unique Name Assumption, which explicitly declares that any two different individual names correspond to different elements in the interpretation domain, is often used with DLs.

We conducted experiments in a real-life application, rental matchmaking, which handles requests for renting houses in an ontology with more than one million individual assertions. We carefully designed ten benchmark queries for this application. Experimental results show that the proposed method is rather efficient in computing abductive answers to the benchmark queries.

The remainder of the paper is organized as follows. After providing preliminaries in the next section, in Sect. 3 we give more details on the abductive CQA problem. Then in Sect. 4, we describe the proposed method for computing

all abductive answers to a conjunctive query. Before making a conclusion, we present our experimental evaluation in Sect. 5.

2 Preliminaries

2.1 OWL 2 and DLP

We assume that the reader is familiar with OWL 2 [11] and briefly introduce the most expressive and decidable species of OWL 2, OWL 2 DL, which corresponds to the DL *SRONTQ* [13] with datatypes. An OWL 2 DL ontology consists of an RBox, a TBox and an ABox. The RBox consists of a finite set of role inclusion axioms and role assertions. The TBox consists of a finite set of concept inclusion axioms. The ABox consists of a finite set of individual assertions. OWL 2 DL (i.e. *SRONTQ*) is a syntactic variant of a fragment of First-order Logic, and the semantics of a *SRONTQ* ontology \mathcal{O} can be defined by translating to a formula $\pi(\mathcal{O})$ of First-order Logic with equality, where π is defined in Table 1. We use the traditional rule form to represent $\pi(\mathcal{O})$. For example, consider the ontology (called the *rental ontology*) used in our experiments, the following two axioms in the RBox of the rental ontology: $\text{Tra}(\text{isA})$ and $\text{hasFacility} \circ \text{isA} \sqsubseteq \text{hasFacility}$, and the following two axioms in the TBox: $\text{House} \sqsubseteq \text{Building}$ and $\text{Building} \sqcap \text{TrafficLine} \sqsubseteq \perp$, can be translated to the following First-order rules.

$$\begin{aligned} R_1 &= \forall x, y, z : \text{isA}(x, y) \wedge \text{isA}(y, z) \rightarrow \text{isA}(x, z) \\ R_2 &= \forall x, y, z : \text{hasFacility}(x, y) \wedge \text{isA}(y, z) \rightarrow \text{hasFacility}(x, z) \\ R_3 &= \forall x : \text{House}(x) \rightarrow \text{Building}(x) \\ R_4 &= \forall x : \text{House}(x) \wedge \text{TrafficLine}(x) \rightarrow \perp \end{aligned}$$

Rule R_1 tells that if x is more specific than y and y is more specific than z , then x is more specific than z . Rule R_2 tells that if x has a facility y and y is more specific than z , then x also has a facility z . Rule R_3 tells that if x is a renting house, then it is also a building. Rule R_4 tells if x is a building, then it cannot be a traffic line, where the symbol \perp in rule consequences denotes a contradiction.

A *model* of an OWL 2 DL ontology \mathcal{O} is a model of $\pi(\mathcal{O})$ under the traditional First-order semantics. \mathcal{O} is said to be *consistent* if it admits at least one model. An individual assertion α is said to be *entailed* by \mathcal{O} , denoted by $\mathcal{O} \models \alpha$, if α is satisfied by all models of \mathcal{O} . The *Unique Name Assumption* [1] in \mathcal{O} is an assumption that $\mathcal{O} \models a \not\approx b$ for any two different individual names a and b occurring in \mathcal{O} .

The DLP [12] fragment of OWL 2 DL is the intersection of OWL 2 DL and Horn Logic, another fragment of First-order Logic in which all rules have no existential quantifiers or disjunctions in the consequence part. We simply call an ontology *DLP ontology* if it is expressed in the DLP fragment of OWL 2 DL and adopts the Unique Name Assumption.

2.2 Conjunctive Query Answering

A *conjunctive query* is an expression of the form $\exists \vec{y} : \text{conj}(\vec{x}, \vec{y}, \vec{c})$, where \vec{x} is a vector of *distinguished variables*, \vec{y} a vector of *non-distinguished variables* and

Table 1. The semantics of *SRQIQ* by mapping to First-order Logic

Translating <i>SRQIQ</i> concepts to First-order Logic	
$\pi_x(\top) = \top$	$\pi_x(\perp) = \perp$
$\pi_x(A) = A(x)$	$\pi_x(\neg C) = \neg\pi_x(C)$
$\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$	$\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$
$\pi_x(\exists r.C) = \exists y : \text{ar}(r, x, y) \wedge \pi_y(C)$	$\pi_x(\forall r.C) = \forall y : \text{ar}(r, x, y) \rightarrow \pi_y(C)$
$\pi_x(\exists r.\text{Self}) = \text{ar}(r, x, x)$	$\pi_x(\{a\}) = x \approx a$
$\pi_x(\geq_n r.C) = \exists y_1, \dots, y_n : \bigwedge_{i=1}^n (\text{ar}(r, x, y_i) \wedge \pi_{y_i}(C)) \wedge \bigwedge_{1 \leq i < j \leq n} y_i \not\approx y_j$	
$\pi_x(\leq_n r.C) = \forall y_1, \dots, y_{n+1} : \bigwedge_{i=1}^{n+1} (\text{ar}(r, x, y_i) \wedge \pi_{y_i}(C)) \rightarrow \bigvee_{1 \leq i < j \leq n+1} y_i \approx y_j$	
Translating axioms to First-order Logic	
RBox: $\pi(r_1 \circ \dots \circ r_n \sqsubseteq s) = \forall x_1, \dots, x_{n+1} : \bigwedge_{i=1}^n \text{ar}(r_i, x_i, x_{i+1}) \rightarrow \text{ar}(s, x_1, x_{n+1})$	
$\pi(\text{Tra}(r)) = \forall x, y, z : \text{ar}(r, x, y) \wedge \text{ar}(r, y, z) \rightarrow \text{ar}(r, x, z)$	
$\pi(\text{Sym}(r)) = \forall x, y : \text{ar}(r, x, y) \rightarrow \text{ar}(r, y, x)$	$\pi(\text{Ref}(r)) = \forall x : \text{ar}(r, x, x)$
$\pi(\text{Dis}(r, s)) = \forall x, y : \neg \text{ar}(r, x, y) \vee \neg \text{ar}(s, x, y)$	$\pi(\text{Irr}(r)) = \forall x : \neg \text{ar}(r, x, x)$
TBox: $\pi(C \sqsubseteq D) = \forall x : \pi_x(C) \rightarrow \pi_x(D)$	
ABox: $\pi(C(a)) = \pi_x(C)[x \mapsto a]$	$\pi(r(a, b)) = \text{ar}(r, a, b)$
$\pi(\neg r(a, b)) = \neg \text{ar}(r, a, b)$	$\pi(a \not\approx b) = a \not\approx b$
Translating <i>SRQIQ</i> ontologies to First-order Logic	
$\pi(\mathcal{O}) = \bigwedge_{\alpha \in \mathcal{O}} \pi(\alpha)$	

Note: A denotes a concept name; $\text{ar}(r, x, y)$ denotes $s(y, x)$ if r is an inverse role s^- , or denotes $r(x, y)$ otherwise.

\vec{c} a vector of individuals or constants. $\text{conj}(\vec{x}, \vec{y}, \vec{c})$ denotes a conjunction of *atoms* of the form $A(v)$ or $r(v_1, v_2)$, where A is an *atomic concept* (i.e. a concept name), r is an *atomic role* (i.e. a role name) or a built-in predicate, and v, v_1 and v_2 are variables in \vec{x} and \vec{y} , or individuals or constants in \vec{c} . A *Boolean conjunctive query* is a conjunctive query without distinguished variables.

Given an OWL 2 DL ontology \mathcal{O} and a Boolean conjunctive query $Q = \exists \vec{y} : \text{conj}(\vec{y}, \vec{c})$, a model \mathcal{I} of \mathcal{O} is said to *satisfy* Q if there exists a tuple of (possibly anonymous) individuals or constants whose substitution for the variables in \vec{y} makes every atom in $\text{conj}(\vec{y}, \vec{c})$ satisfied by \mathcal{I} . Q is said to be *entailed* by \mathcal{O} , denoted by $\mathcal{O} \models Q$, if every model of \mathcal{O} satisfies Q . A tuple \vec{t} of individuals is called an *answer* to a conjunctive query $Q(\vec{x}) = \exists \vec{y} : \text{conj}(\vec{x}, \vec{y}, \vec{c})$ in \mathcal{O} if $\mathcal{O} \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, where $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ denotes a Boolean conjunctive query obtained from $Q(\vec{x})$ by replacing every variable in \vec{x} with its corresponding individual in \vec{t} . The *conjunctive query answering (CQA) problem* is to compute all answers to a conjunctive query in an ontology.

3 The Abductive CQA Problem

By expressing offer information as individual assertions and request information as conjunctive queries, the matchmaking problem can be treated as the CQA problem. For example, when we want to find all renting houses whose price is up to 4000 yuan per month, we can pose the following conjunctive query upon the rental ontology: $\exists y : \text{House}(x) \wedge \text{rent}(x, y) \wedge y \leq 4000$. Then the answers to

this query correspond to renting houses to be found. However, the answers to a conjunctive query may not provide all choices to a requester. For example, when the price of a renting house is missing, possibly due to incomplete extraction from Web sources, this renting house will not be an answer of the aforementioned query, though its rental price is 3000 yuan per month in reality. Hence those answers in a certain enlarged ontology may also correspond to offers that match the request in reality. This enlarged ontology can be seen as the result of adding missing data about offers to the original ontology. Since offer information is expressed as individual assertions, the missing data should be restricted to individual assertions. Moreover, the number of added assertions should be bounded by some constant that reflects the incompleteness of the original ontology, while the added assertions should be confined by certain concept or role names according to the actual situation. Hence, we introduce an extended CQA problem, called *abductive CQA problem*, defined below.

Definition 1 (Abductive CQA). Given a consistent ontology \mathcal{O} , a conjunctive query Q , a non-negative integer k , two disjoint sets of concept or role names S_A and S_C , where the concept or role names in S_A (resp. S_C) are called *abducible predicates* (resp. *closed predicates*), a tuple \vec{t} of individuals is called an *abductive answer to Q* in \mathcal{O} w.r.t. k, S_A and S_C , if there exists a set \mathcal{A} of individual assertions on the predicates in S_A such that $|\mathcal{A}| \leq k$, $\mathcal{O} \cup \mathcal{A} \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, $\mathcal{O} \cup \mathcal{A}$ is consistent, and $\mathcal{O} \models \alpha$ for all individual assertions α on the predicates in S_C such that $\mathcal{O} \cup \mathcal{A} \models \alpha$, where $|S|$ denotes the cardinality of a set S , and \mathcal{A} is said to be *attached with \vec{t}* . The *abductive CQA problem* is to compute all abductive answers to Q in \mathcal{O} w.r.t. k, S_A and S_C .

In the above definition, \mathcal{A} can be seen as the missing data about a certain offer. It consists of at most k individual assertions and should be a set such that the union of it and the given ontology \mathcal{O} is consistent. Here, we call \mathcal{A} a *candidate complement set* if $|\mathcal{A}| \leq k$ and $\mathcal{O} \cup \mathcal{A}$ is consistent. The set of abducible predicates, S_A , restricts all concepts or roles appearing in a candidate complement set to concept or role names in S_A . The set of closed predicates, S_C , further restricts that the union of a candidate complement set and the given ontology does not entail any individual assertion which is on a closed predicate but is not entailed by the given ontology. In general, a concept or role name can be set as an abducible predicate if some of its instances are possibly missing; it can be set as a closed predicate if all individual assertions on it are ensured to be entailed by the given ontology. However, it is not necessary that a concept or role name is set to be either abducible or closed.

The abductive CQA problem is similar to the ABox abduction problem proposed in [7], which computes all minimal sets \mathcal{A} of individual assertions on a set S of predicates such that $\mathcal{O} \cup \mathcal{A}$ is consistent, $\mathcal{A} \not\models G$ and $\mathcal{O} \cup \mathcal{A} \models G$, for a consistent ontology \mathcal{O} and a set G of individual assertions. Compared to the ABox abduction problem, the abductive CQA problem also restricts \mathcal{A} to a set that consists of individual assertions on certain predicates and does not introduce inconsistency. But it computes abductive answers to a conjunctive query

by considering all possible \mathcal{A} instead of computing certain \mathcal{A} . Moreover, it introduces the use of the bounded number k and a set of closed predicates. The bounded number k is used to control the extensiveness of abducible answers, while the usefulness of closed predicates is described below.

One use of the closed predicates is to simulate the use of disjoint concept or role axioms, which declare that two concepts or two roles are disjoint. For example, suppose $\text{City}(a)$ is entailed by \mathcal{O} and $\text{House}(a)$ is not. When \mathcal{O} does not contain the axiom declaring that House and City are disjoint, $\text{House}(a)$ can possibly be entailed by $\mathcal{O} \cup \mathcal{A}$ for some candidate complement set \mathcal{A} . But we actually do not expect $\text{House}(a)$ to be entailed by $\mathcal{O} \cup \mathcal{A}$ as $\text{House}(a)$ does not hold in reality. Hence we can define House as a closed predicate, so that all candidate complement sets \mathcal{A} making $\mathcal{O} \cup \mathcal{A}$ entail $\text{House}(a)$ are not considered in computing abductive answers. Since disjoint concept or role axioms can easily be neglected when manually constructing or maintaining ontologies, the closed predicates are often needed to substitute the use of these axioms. Another use of the closed predicates is to enable some optimizations in computing abductive answers; this will be shown in the next section.

It should be mentioned that closed predicates are not predicates in an NBox (Negation-as-failure Box) [17]. Defining concept or role names in the NBox of an ontology impacts the semantics of the ontology. For example, defining House as a concept name in the NBox amounts to adding the axiom $\text{House} \sqsubseteq \{a_1, \dots, a_n\}$ to the ontology, where a_1, \dots, a_n are all (explicit and implicit) instances of House in the ontology. In contrast, defining concept or role names as closed predicates does not impact the semantics of the ontology; it only determines what kind of candidate complement sets can be considered. In fact, when $k = 0$, the set of abductive answers to Q in \mathcal{O} w.r.t. k, S_A and S_C coincides with the set of answers to Q in \mathcal{O} no matter how S_A and S_C are set.

For the abductive CQA problem, we assume that the given ontology \mathcal{O} has been *extensionally reduced* by replacing concept assertions $C(a)$ with $Q_C(a)$ and role assertions $(\neg)r^-(a, b)$ with $(\neg)r(b, a)$ in the ABox, and by adding $Q_C \sqsubseteq C$ to the TBox, where C is a concept appearing in the ABox but is neither an atomic concept nor a negated atomic concept, Q_C is a new globally unique atomic concept corresponding to C , and r is an atomic role appearing in the ABox. Extensionally reducing an ontology does not impact abductive answers to conjunctive queries in the ontology.

4 A Method for the Abductive CQA Problem

The ABox abduction method proposed in [7] encodes the ABox abduction problem into a Prolog program and solves it with Prolog engines. We extend this method to solve the abductive CQA problem in a consistent extensionally reduced DLP ontology \mathcal{O} . It has been empirically shown in [7] that the ABox abduction method is feasible only when the translated First-order rules have no equality in heads. To guarantee this, expressing \mathcal{O} in the DLP fragment of OWL 2 DL is not enough, thus we also assume that \mathcal{O} adopts the Unique Name

Assumption so that any translated rule with equational head atoms can be converted to a semantically equivalent one by rewriting equational head atoms to inequational atoms and moving them to the rule body.

Compared to the ABox abduction method in [7], the proposed method for abductive CQA has significant extensions. First, the proposed method handles new parameters that are not considered in [7] (i.e. the bounded number k and the set S_C of closed predicates). Second, the proposed method introduces new optimizations based on abducible predicates and closed predicates.

Let $F(\mathcal{O})$ denote the set of First-order rules translated from \mathcal{O} where there are no equational head atoms. The proposed method has the following six steps.

In the first step, for the purpose of optimization the unique minimal model of $F(\mathcal{O})$ is computed and all ground atoms in this model are added to $F(\mathcal{O})$ as *ground facts* (i.e. variable-free rules whose consequence part is not \perp), yielding $F'(\mathcal{O})$. We call a concept or role name P an *addable predicate* if there is a sequence of rules r_1, \dots, r_n in $F(\mathcal{O})$ (where $n \geq 1$) such that P occurs in the head of r_1 , some abducible predicates occur in the body of r_n , and the body of r_i and the head of r_{i+1} have common predicates for all $i \in \{1, \dots, n-1\}$. Intuitively, only individual assertions on addable predicates can possibly be added to the unique minimal model of $F(\mathcal{O})$ after individual assertions on abducible predicates are added to \mathcal{O} . Consider an arbitrary conjunctive query $Q(\vec{x}) = \exists \vec{y} : P(\vec{x}, \vec{y}, \vec{c})$ that consists of a single atom $P(\vec{x}, \vec{y}, \vec{c})$ where P is a non-addable or closed predicate. For an arbitrary set \mathcal{A} of individual assertions that will be attached with some abductive answers of Q in \mathcal{O} , there is not any individual assertion α on P that is entailed by $\mathcal{O} \cup \mathcal{A}$ but not entailed by \mathcal{O} , hence the set of abductive answers of Q in \mathcal{O} coincides with the set of answers of Q in \mathcal{O} no matter how the bounded number k and closed predicates are set. Since the answers of Q in \mathcal{O} can be retrieved from the unique minimal model of $F(\mathcal{O})$, the abductive answers of Q in \mathcal{O} can be directly retrieved from the ground facts in $F'(\mathcal{O})$. Hence, the use of non-addable or closed predicates can yield a more efficient encoding of the abductive CQA problem. To achieve this encoding, the following steps consider $F'(\mathcal{O})$ instead of $F(\mathcal{O})$.

In what follows, we assume that the predicate **House** is closed and the predicates **House**, **Building**, **isA**, **hasFacility**, **locatesIn** and **rent** are addable.

In the second step, all ground facts in $F'(\mathcal{O})$ are encoded into Prolog rules. For example, the ground fact $\rightarrow \text{isA}(a, b)$ is encoded into the following two Prolog rules, where the last rule is written once in the encoded Prolog program for all ground facts on **isA**. In Prolog atoms, the prefix “**pf**” or “**p**” is added to concept or role names, because in the Prolog syntax only for variables the first letter is capitalized, while concept or role names are not variables.

pfisA(a, b).

pisA(X, Y, L_i, L_o) :- **pf**isA(X, Y), $L_o = L_i$.

In the third step, all *definite rules* (i.e. First-order rules whose consequence part is not \perp) in $F'(\mathcal{O})$ that have addable but not closed predicates in heads and have variables are encoded into Prolog rules. Note that any definite rule in $F'(\mathcal{O})$ whose head predicate is non-addable or closed is not encoded, because all

instances of the head predicate can be directly retrieved from the ground facts in $F'(\mathcal{O})$, i.e. retrieved through the Prolog rules generated in the previous step. To manage individual assertions that can be added to \mathcal{O} , every non-built-in atom in a definite rule in $F'(\mathcal{O})$ is encoded as a Prolog atom with an extra input argument and an extra output argument. The input (resp. output) argument is a list representing the original (resp. updated) set of added assertions if the atom is on an addable but not closed predicate, or is the empty list otherwise. A list L is of the form $[t_1, \dots, t_n]$, where t_i is of the form $(a, \text{“rdf:type”}, \mathbf{pA})$ or (a, \mathbf{pr}, b) ; it is called *empty* if it is of the form $[]$. A list L can be decoded into a set of individual assertions $\{t'_1, \dots, t'_n\}$, denoted by $\text{decode}(L)$, where t'_i is rewritten from t_i by rewriting $(a, \text{“rdf:type”}, \mathbf{pA})$ to a concept assertion $A(a)$ and (a, \mathbf{pr}, b) to a role assertion $r(a, b)$. The two extra arguments of a Prolog atom, which is encoded from an atom on non-addable or closed predicate, are set as empty lists, because all instances of the predicate can be directly retrieved from the ground facts in $F'(\mathcal{O})$. In the encoded Prolog rule, the extra input argument of the head atom, L_i , is the extra input argument of the first body atom on addable but not closed predicates, whereas the extra output argument of the head atom, L_o , is the extra output argument of the last body atom on addable but not closed predicates; if there are no body atoms on addable but not closed predicates, an Prolog atom $L_o = L_i$ is added to the body to define what L_o is. For example, the rules R_1 and R_3 in Sect. 2 are encoded into the following two Prolog rules.

$$\begin{aligned} \mathbf{p}isA(X, Z, L_i, L_o) &:- \mathbf{p}isA(X, Y, L_i, L_1), \mathbf{p}isA(Y, Z, L_1, L_o). \\ \mathbf{p}Building(X, L_i, L_o) &:- \mathbf{p}House(X, [], []), L_o = L_i. \end{aligned}$$

It should be noted that all *constraints* (i.e. First-order rules whose consequence part is \perp), such as the rule R_4 in Sect. 2, are not encoded into Prolog rules. This is because constraints are only used to determine if the added assertions are consistent with \mathcal{O} . But this consistency checking in a Prolog engine is based on brute-force like search and is generally less efficient than calling an external consistency checker, so all constraints in $F'(\mathcal{O})$ are ignored.

In the fourth step, all Prolog predicates occurring in cycles in the set of Prolog rules generated in the previous step are declared to be *tabled* predicates. Setting a Prolog predicate tabled means that any Prolog atom on this predicate is prevented from calling multiple times. This is a crucial step for guaranteeing termination when calling Prolog atoms in the encoded Prolog program. It is similar to the first step in the ABox abduction method [7], except that Prolog predicates encoded from non-addable or closed predicates are not set as tabled predicates, because they cannot be head predicates of Prolog rules generated in the previous step and will not occur in cycles in the encoded Prolog program.

In the fifth step, the given conjunctive query Q is encoded into a Prolog rule with a nullary head atom `go`, where every non-built-in atom in Q is also encoded as a Prolog atom with an extra input argument and an extra output argument. Likewise, the input (resp. output) argument is a list representing the original (resp. updated) set of added assertions if the atom is on an addable but not closed predicate, or is the empty list otherwise. In order to output all abductive answers to Q , two Prolog atoms `output(X_1, \dots, X_n, L)` and `fail` are added to the

body of the encoded Prolog rule, where X_1, \dots, X_n are all distinguished variables in Q , and L is the extra output argument of the last body atom on addable but not closed predicates; if there are no body atoms on addable but not closed predicates, L is set as the empty list. $\text{output}(X_1, \dots, X_n, L)$ directly returns true if the tuple $\langle X_1, \dots, X_n \rangle$ has been output; otherwise, if L is the empty list, the atom outputs $\langle X_1, \dots, X_n \rangle$ and returns true; otherwise, if $\mathcal{O} \cup \text{decode}(L)$ is consistent and every ground atom on closed predicates in the unique minimal model of $F(\mathcal{O}) \cup \text{decode}(L)$ is also in the unique minimal model of $F(\mathcal{O})$ (which is checked by calling an external consistency checker), the atom outputs $\langle X_1, \dots, X_n \rangle$ and returns true, else the atom returns fail. In other words, an external consistency checker is called when and only when the tuple $\langle X_1, \dots, X_n \rangle$ has not been output and L is not empty. The Prolog atom fail forces the Prolog engine to enumerate all possible instantiations for variables in the encoded Prolog rule when calling go, so as to obtain all abductive answers to Q . Suppose we want to find all renting houses that locate in GZ (Guangzhou) and have a rental price less than 3000 yuan per month, we can express it as the following conjunctive query.

$$Q(x) = \exists y : \text{House}(x) \wedge \text{locatesIn}(x, \text{GZ}) \wedge \text{rent}(x, y) \wedge y < 3000$$

Then the above query can be encoded into the following Prolog rule.

$$\begin{aligned} \text{go} :- & \text{pHouse}(X, [], []), \text{plocatesIn}(X, \text{"GZ"}, [], L_1), \text{prent}(X, Y, L_1, L_2), \\ & Y < 3000, \text{output}(X, L_2), \text{fail}. \end{aligned}$$

In the last step, for every abducible predicate in S_A , two Prolog rules are added. The first rule says that the updated set of added assertions is the original one, if the individual assertion to be added is already in the original set of added assertions. The second rule says that the updated set of added assertions is obtained from the original one by inserting an individual assertion on P , if the number of added assertions in the original set is less than k . For example, suppose the predicate isA is abducible, then the following two Prolog rules are added, where the Prolog atom $\text{dom}(X)$ ensures X to be an individual or a constant and returns true, $\text{in}(t, L)$ sets t as a member of the list L and returns true if t can possibly be grounded to a member of L or returns false otherwise, $\text{less}(L, k)$ returns true iff the the number of members in L is less than k , and $\text{insert}(t, L, L')$ sets L' as the resulting list obtained by inserting t to L and turns true.

$$\text{pisA}(X, Y, L_i, L_o) :- \text{in}((X, \text{pisA}, Y), L_i), L_o = L_i.$$

$$\text{pisA}(X, Y, L_i, L_o) :- \text{less}(L_i, k), \text{dom}(X), \text{dom}(Y), \text{insert}((X, \text{pisA}, Y), L_i, L_o).$$

The following theorem shows the correctness of the above encoding method.

Theorem 1. *The encoded Prolog program outputs exactly all abductive answers to $Q(\vec{x}) = \exists \vec{y} : \text{conj}(\vec{x}, \vec{y}, \vec{c})$ in \mathcal{O} w.r.t. k, S_A and S_C when calling go.*

Proof. (1) Let \vec{t} be a tuple of individuals or constants output by the encoded Prolog program, and \mathcal{A} be the set of individual assertions attached with \vec{t} when \vec{t} is output. Since backward inference in Prolog is sound, it is clear that $F'(\mathcal{O}) \cup \mathcal{A} \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$. Furthermore, when \vec{t} is output (during calling a Prolog atom on output), it is confirmed that $\mathcal{O} \cup \mathcal{A}$ is consistent, and every individual assertion on closed predicates is not entailed by $\mathcal{O} \cup \mathcal{A}$ unless it is

entailed by \mathcal{O} . According to the last step of the encoding method, \mathcal{A} should only consist of individual assertions on abducible predicates and $|\mathcal{A}| \leq k$. Hence, by Definition II, \vec{t} is an abductive answer to $Q(\vec{x})$ in \mathcal{O} w.r.t. k, S_A and S_C .

(2) Let the tuple \vec{t} be an abductive answer to Q in \mathcal{O} w.r.t. k, S_A and S_C , then there exists a set \mathcal{A} of individual assertions on abducible predicates such that $|\mathcal{A}| \leq k, \mathcal{O} \cup \mathcal{A} \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, $\mathcal{O} \cup \mathcal{A}$ is consistent and $\mathcal{O} \models \alpha$ for all individual assertions α on closed predicates such that $\mathcal{O} \cup \mathcal{A} \models \alpha$. The unique minimal model of $F(\mathcal{O}) \cup \mathcal{A}$ is equal to the set of ground atoms occurring in the least fixpoint of $\Pi^{(n)}$, where $\Pi^{(0)} = \emptyset$ and for $n \geq 1$, $\Pi^{(n)} = \{R\sigma \mid R \in F(\mathcal{O}) \cup \mathcal{A}, \sigma \text{ is a mapping from variables in } R \text{ to constants in } F(\mathcal{O}) \cup \mathcal{A} \text{ such that all body atoms of } R\sigma \text{ occur in } \Pi^{(n-1)}\}$. Let Δ_n ($n \geq 1$) denote the set of ground atoms occurring in $\Pi^{(n)}$.

Consider an arbitrary ground atom α in the unique minimal model of $F(\mathcal{O}) \cup \mathcal{A}$. Let $p(a_1, \dots, a_m, L_i, L_o)$ be encoded from α . In case α is on non-addable or closed predicates, it is clear that calling $p(a_1, \dots, a_m, [], [])$ will return true according to the Prolog rules generated in the second step of the encoding method. In other cases, we show by induction on Δ_n that (*) calling the Prolog atom $p(a_1, \dots, a_m, L_i, L_o)$ will return true with $\text{decode}(L_o)$ set as a subset of \mathcal{A} when L_o is given as a variable and L_i is given as a specific list such that $\text{decode}(L_i) \subseteq \mathcal{A}$. In what follows, we assume that L_o is a variable and $\text{decode}(L_i) \subseteq \mathcal{A}$. Consider the case where $\alpha \in \Delta_1$. If $\alpha \in \mathcal{A}$, then according to the Prolog rules generated in the last step of the encoding method, calling $p(a_1, \dots, a_m, L_i, L_o)$ will return true with $\text{decode}(L_o) \subseteq \text{decode}(L_i) \cup \{\alpha\} \subseteq \mathcal{A}$; otherwise, according to the Prolog rules generated in the second step, calling $p(a_1, \dots, a_m, L_i, L_o)$ will return true with $\text{decode}(L_o) = \text{decode}(L_i) \subseteq \mathcal{A}$. Suppose the result (*) holds for all ground atoms in Δ_k ($k \geq 1$). Consider the case where $\alpha \in \Delta_{k+1} \setminus \Delta_k$. There exists a rule $R \in F(\mathcal{O})$ and a mapping σ from variables in R to constants in $F(\mathcal{O}) \cup \mathcal{A}$ such that all body atoms of $R\sigma$ belong to Δ_k . According to the Prolog rules generated in the third step, there exists an encoded Prolog rule R_e whose head atom is $p(a_1, \dots, a_m, L_i, L_o)$ and whose body atoms except $L_o = L_i$ are encoded from body atoms of $R\sigma$. By inductive hypothesis, all body atoms of R_e except $L_o = L_i$, when being called, will return true with their extra output parameters set as some lists L such that $\text{decode}(L) \subseteq \mathcal{A}$. Thus, calling $p(a_1, \dots, a_m, L_i, L_o)$ will return true with $\text{decode}(L_o) \subseteq \mathcal{A}$ by triggering R_e .

Since \mathcal{O} is a DLP ontology, all arguments of ground atoms in the unique minimal model of $F(\mathcal{O}) \cup \mathcal{A}$ are constants in $F(\mathcal{O}) \cup \mathcal{A}$. Since $\mathcal{O} \cup \mathcal{A} \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, there must be a tuple \vec{s} of constants in $F(\mathcal{O}) \cup \mathcal{A}$ such that $F(\mathcal{O}) \cup \mathcal{A} \models \alpha$ for every ground atom α in $Q(\vec{x})[\vec{x} \mapsto \vec{t}, \vec{y} \mapsto \vec{s}]$. Let $\alpha_1, \dots, \alpha_n$ be all ground atoms in $Q(\vec{x})[\vec{x} \mapsto \vec{t}, \vec{y} \mapsto \vec{s}]$, and β_i be a Prolog atom encoded from α_i as follows: If α_i is a built-in atom, then β_i is encoded as the corresponding built-in atom in Prolog; else if α_i is on non-addable or closed predicates, β_i is encoded to an atom of the form $p(a_1, \dots, a_m, [], [])$; else β_i is encoded to an atom of the form $p(a_1, \dots, a_m, L, L')$, where L is the last parameter of the previous β_j on addable but not closed predicates (if this β_j does not exist, L is the empty list). Consider the Prolog rule for encoding Q which is generated in the fifth

step. β_1, \dots, β_n and $\text{output}(\vec{T}, L)$ will be called one by one when calling go . By the results proved in the previous paragraph, calling β_i will return true for all $i \in \{1, \dots, n\}$, and L must be set as a list such that $\text{decode}(L) \subseteq \mathcal{A}$ before calling $\text{output}(\vec{T}, L)$. When $\text{output}(\vec{T}, L)$ is called, if \vec{T} has not been output, then since $\text{decode}(L) \subseteq \mathcal{A}$, L should be the empty list or satisfy that $\mathcal{O} \cup \text{decode}(L)$ is consistent and every ground atom on closed predicates in the unique minimal model of $F(\mathcal{O}) \cup \text{decode}(L)$ is also in the unique minimal model of $F(\mathcal{O})$, hence \vec{T} should be output. To conclude, \vec{T} must be output when calling go . \square

5 Experimental Evaluation

We conducted experiments in a real-life application, rental matchmaking, which is based on a rental ontology and handles requests for renting houses. The goal of this application is to provide a suitable rental matchmaking system for residents in China, under the current circumstances that many families in China cannot afford a house.

We manually constructed the RBox and the TBox of the rental ontology, which have 129 logical axioms, 36 concept names and 35 role names, using Protégé (version 4.1)¹, a well-known ontology editor. In addition, we built automatic tools to extract information from existing Websites, including a rental Website², a traffic Website³ and an administrative region Website⁴. We manually wrote annotation rules in these tools to convert the extracted data to individual assertions. We also manually added some individual assertions on the role `isA` to define that some facilities are more specific than some other facilities. When adding individual assertions annotated from different Websites to the rental ontology, inconsistency occurs because some homonymous entities that belong to disjoint concepts are treated as the same individual. We resolved the inconsistency by using the method proposed in [8] to remove a cardinality-minimal set of axioms. Finally, we obtained a consistent ontology⁵ with 32,954 individuals and 1,152,336 logical axioms, where the number of renting houses is 9,248. The ontology is an existentially reduced DLP ontology.

We implemented the proposed method in JAVA and used XSB⁶ as the back-end Prolog engine since XSB supports tabled predicates. Considering that the matchmaking results returned by the proposed method have a formal semantics, we did not focus on the quality of the matchmaking results but on the efficiency in computing them, which is a crucial criterion for verifying the feasibility of the method. All experiments were conducted on a PC with Pentium Dual Core 2.80GHz CPU and 16GB RAM, running Win 7 (64 bit).

¹ <http://protege.stanford.edu/>

² <http://www.soufun.com/>

³ <http://www.8684.cn/>

⁴ <http://www.chinaqihua.cn/>

⁵ <http://jfd�.limewebs.com/papers/rental.zip>

⁶ <http://xsb.sourceforge.net/>

$Q_1(x) = \text{House}(x)$
Meaning: <i>Find all renting houses.</i>
$Q_2(x) = \exists y : \text{House}(x) \wedge \text{locatesIn}(x, \text{Liwan}) \wedge \text{rent}(x, y) \wedge y \geq 1000 \wedge y \leq 2000$
Meaning: <i>Find all renting houses in the Liwan district (a district in Guangzhou) whose rental price is from 1000 to 2000 yuan per month.</i>
$Q_3(x) = \text{House}(x) \wedge \text{locatesIn}(x, \text{Baiyun}) \wedge \text{numOfBedrooms}(x, 2) \wedge \text{numOfLivingRooms}(x, 1) \wedge \text{locatesNear}(x, \text{GDUFS})$
Meaning: <i>Find all renting houses in the Baiyun district (a district in Guangzhou) that have two bedrooms and one living-room and locate near GDUFS (Guangdong University of Foreign Studies).</i>
$Q_4(x) = \exists y : \text{House}(x) \wedge \text{locatesIn}(x, \text{Liwan}) \wedge \text{numOfBedrooms}(x, y) \wedge y \geq 1 \wedge y \leq 3 \wedge \text{numOfLivingRooms}(x, 1) \wedge \text{numOfKitchens}(x, 1) \wedge \text{numOfBathrooms}(x, 1)$
Meaning: <i>Find all renting houses in the Liwan district that have one to three bedrooms, one living-room, one kitchen and one bathroom.</i>
$Q_5(x) = \text{House}(x) \wedge \text{locatesIn}(x, \text{ZhuJiangNewTown}) \wedge \text{towards}(x, \text{South})$
Meaning: <i>Find all southward renting houses in the Zhu Jiang New Town.</i>
$Q_6(x) = \exists y, z : \text{House}(x) \wedge \text{locatesIn}(x, \text{Liwan}) \wedge \text{locatesNear}(x, y) \wedge \text{isOriginalOf}(y, z) \wedge \text{Bus}(z)$
Meaning: <i>Find all renting houses in the Liwan district that locate near the origin stop of some bus line.</i>
$Q_7(x) = \exists y_1, y_2 : \text{House}(x) \wedge \text{locatesNear}(x, y_1) \wedge \text{locatesNear}(x, y_2) \wedge y_1 \neq y_2 \wedge \text{isLineOf}(y_1, \text{HEMC_GDUFS}) \wedge \text{isLineOf}(y_2, \text{HEMC_GDUFS})$
Meaning: <i>Find all renting houses locating near two different traffic lines both of which have a stop called HEMC_GDUFS (the section of Guangdong University of Foreign Studies which is in Higher Education Mega Center).</i>
$Q_8(x) = \text{House}(x) \wedge \text{locatesIn}(x, \text{Liwan}) \wedge \text{hasFacility}(x, \text{Club}) \wedge \text{hasFacility}(x, \text{SportsArea})$
Meaning: <i>Find all renting houses in the Liwan district that have clubs and sports areas.</i>
$Q_9(x) = \exists y : \text{House}(x) \wedge \text{numOfBedrooms}(x, 3) \wedge \text{numOfLivingRooms}(x, 2) \wedge \text{rent}(x, y) \wedge y \leq 3000 \wedge \text{hasFacility}(x, \text{Club}) \wedge \text{hasFacility}(x, \text{SportsArea})$
Meaning: <i>Find all renting houses that have three bedrooms and two living-rooms, have a rental price no more than 3000 yuan per month, and have clubs and sports areas.</i>
$Q_{10}(x) = \exists y : \text{House}(x) \wedge \text{locatesIn}(x, \text{GZ}) \wedge \text{floorNo}(x, y) \wedge \text{numOfFloors}(x, z) \wedge y \neq z$
Meaning: <i>Find all renting houses in GZ (Guangzhou) that are not at the top floor of a building.</i>

Fig. 1. The benchmark queries for testing the proposed method

We carefully designed ten benchmark queries, shown in Fig. 1, to test the efficiency of the proposed method, where all individuals in these queries are automatically generated URIs that correspond to names in Chinese, but they are shown by meaningful names here for readability. The first query is the basic one which can be directly answered by the original rental Website. The next four

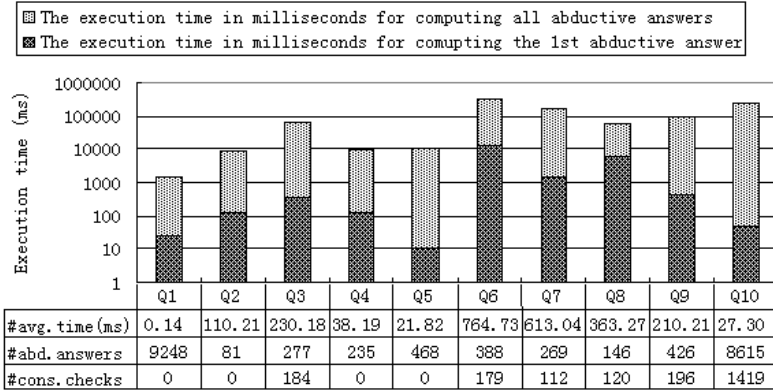


Fig. 2. The statistics for each benchmark query (Note: in the bottom table, row 1 shows the average execution time in milliseconds for computing one abductive answer, row 2 shows the number of abductive answers, and row 3 shows the number of calls to an external consistency checker)

queries are complex queries about multiple aspects of renting houses. The 6th query and the 7th query are complex queries about renting houses and traffic lines, and they are not supported by the original rental Website. The last three queries involve more complex reasoning. For example, the 8th query and the 9th query involve reasoning on the rules R_1 and R_2 in Sect. 2.

We tested the proposed method on computing all abductive answers to every benchmark query in the rental ontology w.r.t. k, S_A and S_C , where $k = 1$, $S_A = \{\text{locatesNear, hasFacility, rent, towards, floorNo, numOfFloors}\}$ (which consists of all predicates on which the information may be incomplete) and $S_C = \{\text{House}\}$ (which consists of one predicate on which the information is surely complete).

Our implemented system works in two phases. In the first phase, all information except specific conjunctive queries is encoded into a Prolog program, which is then loaded into XSB. This phase is independent of any given query and is performed offline. In our experiments, this phase was done in 875 seconds. In the second phase, every benchmark query is encoded into a Prolog rule. Then this rule is added to the Prolog program obtained in the first phase and is evaluated by XSB. The statistics in this phase are shown in Fig. 2. For 7/1/2 benchmark queries, the first abductive answer was computed in 1/2/13 seconds. For 4/3/3 benchmark queries, all abductive answers were computed in 10/100/300 seconds. For all benchmark queries, each abductive answer was computed in one second on average. Note that all benchmark queries have abductive answers in the rental ontology and the evaluation of six benchmark queries needs to call an external consistency checker. This shows that the system is able to efficiently handle nontrivial requests for renting houses.

6 Conclusion and Future Work

We have proposed a new semantic approach to matchmaking based on the abductive CQA semantics. By considering that data are often incomplete in Web-based scenarios, this approach defines matches for a request as abductive answers to a conjunctive query that expresses the request. Furthermore, we proposed a sound and complete method for computing all abductive answers in a consistent extensionally reduced DLP ontology. Experimental results on rental matchmaking demonstrated the feasibility of the proposed method.

For future work, we plan to define reasonable measures for ranking abductive answers. These measures can be computed according to the minimal sets of added assertions attached with abductive answers. We also plan to define another extended CQA semantics to perform matchmaking in an inconsistent and incomplete ontology without rendering the ontology consistent beforehand.

Acknowledgement. Jianfeng Du and Shuai Wang are partly supported by the NSFC under grant 61005043 and the Undergraduate Innovative Experiment Project in Guangdong University of Foreign Studies. Guilin Qi is partly supported by Excellent Youth Scholars Program of Southeast University under grant 4009001011, the NSFC under grant 61003157, Jiangsu Science Foundation under grant BK2010412, and the Key Laboratory of Computer Network and Information Integration (Southeast University). Jeff Z. Pan is partly supported by the EU K-Drive project and the RCUK dot.rural project. Yong Hu is partly supported by the NSFC under grant 70801020.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press (2003)
2. Batabyal, A.A., DeAngelo, G.J.: To match or not to match: Aspects of marital matchmaking under uncertainty. *Operations Research Letters* 36(1), 94–98 (2008)
3. Bianchini, D., Antonellis, V.D., Melchiori, M.: Flexible semantic-based service matchmaking and discovery. *World Wide Web* 11(2), 227–251 (2008)
4. Cimiano, P., Völker, J.: Text2onto - A Framework for Ontology Learning and Data-Driven Change Discovery. In: Montoyo, A., Muñoz, R., Métais, E. (eds.) *NLDB 2005*. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
5. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. *Electronic Commerce Research and Applications* 4(4), 345–361 (2005)
6. Colucci, S., Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M., Mottola, M.: A formal approach to ontology-based semantic match of skills descriptions. *Journal of Universal Computer Science* 9(12), 1437–1454 (2003)
7. Du, J., Qi, G., Shen, Y., Pan, J.Z.: Towards practical abox abduction in large OWL DL ontologies. In: *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1160–1165 (2011)

8. Du, J., Shen, Y.: Computing minimum cost diagnoses to repair populated DL-based ontologies. In: Proc. of the 17th International World Wide Web Conference (WWW), pp. 265–274 (2008)
9. Fazzinga, B., Gianforme, G., Gottlob, G., Lukasiewicz, T.: Semantic Web Search Based on Ontological Conjunctive Queries. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 153–172. Springer, Heidelberg (2010)
10. Fenza, G., Loia, V., Senatore, S.: A hybrid approach to semantic web services matchmaking. *International Journal of Approximate Reasoning* 48(3), 808–828 (2008)
11. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. *Journal of Web Semantics* 6(4), 309–322 (2008)
12. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. of the 12th International World Wide Web Conference (WWW), pp. 48–57 (2003)
13. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SRQIQ*. In: Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 57–67 (2006)
14. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proc. of the 12th International World Wide Web Conference (WWW), pp. 331–339 (2003)
15. Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: A system for principled matchmaking in an electronic marketplace. In: Proc. of the 12th International World Wide Web Conference (WWW), pp. 321–330 (2003)
16. Ragone, A., Straccia, U., Noia, T.D., Sciascio, E.D., Donini, F.M.: Fuzzy matchmaking in e-marketplaces of peer entities using datalog. *Fuzzy Sets and Systems* 160(2), 251–268 (2009)
17. Ren, Y., Pan, J.Z., Zhao, Y.: Closed world reasoning for OWL2 with NBox. *Journal of Tsinghua Science and Technology* 15(6), 692–701 (2010)
18. Shu, G., Rana, O.F., Avis, N.J., Chen, D.: Ontology-based semantic matchmaking approach. *Advances in Engineering Software* 38(1), 59–67 (2007)

GeniUS: Generic User Modeling Library for the Social Semantic Web

Qi Gao, Fabian Abel, and Geert-Jan Houben

Web Information Systems, Delft University of Technology
{q.gao,f.abel,g.j.p.m.houben}@tudelft.nl

Abstract. In this paper, we present GeniUS, a generic topic and user modeling library for the Social Semantic Web that enriches the semantics of social data and status messages particularly. Given a stream of messages, it allows for generating topic and user profiles that summarize the stream according to domain- and application-specific needs which can be specified by the requesting party. Therefore, GeniUS can be applied in various application settings. In this paper, we analyze and evaluate GeniUS in six different application domains. Given users' status messages from Twitter, we investigate the quality of profiles that are generated by different GeniUS user modeling strategies for supporting various recommendation tasks ranging from product recommendations to more specific recommendations as required in book or software product stores. Our evaluation shows that GeniUS succeeds in inferring the semantic meaning of Twitter status messages. We prove that it can successfully adapt to a given domain and application context allowing for tremendous improvements of the recommendation quality when domain-specific semantic filtering is applied to remove noise from the profiles.

Keywords: user modeling, social web, semantic web, twitter, semantic enrichment, filtering.

1 Introduction

On Social Web platforms such as Facebook, Google+ or Twitter people post status messages in which they report about their daily life or share their opinions on events and topics they are interested in. Exploiting those message streams promises to be of benefit for applications that need to understand the current demands and concerns of the people [1,2,3]. However, status messages are usually unstructured and short which makes it difficult for applications to automatically understand their semantic meaning. Consequently, it is difficult to exploit these message streams to understand user interests and demands.

Applications that aim for personalization require profile information about their users such as the preferences or interests. Different applications may need specific types of user profiles. For example, an online book store that features a book recommendation functionality requires information about a user's interests in books, a music recommendation platform needs to gather information about

a user’s musical taste and a movie recommendation system has to infer a user’s preferences in movies. Today, systems typically collect data on their own, i.e. they base their decisions on the user data people generate within the system and ignore user data available in other systems (on the Social Web). This may pose sparsity problems when new users register to the system or in situations where users infrequently interact with the system. For example, if a new user signs up to the online book store then the book recommendation functionality might get problems because it lacks information about the user’s preferences in books. In such situations where applications suffer from sparse information about a user, it can be helpful to consider profile information from other sources on the Social Web [4].

People may reveal interests in books, music or movies in their status messages on platforms such as Twitter or Facebook. However, filtering out and inferring user profile information that is appropriate for a given application is difficult because the semantics of Twitter and Facebook status messages are not explicitly defined. For example, the online book store, music platform and movie recommendation system would need to infer the semantic meaning of a Twitter message such as “Boom! Haddon rules: <http://bit.ly/4YG6t8s>” and decide whether it reveals relevant information about the publisher’s preferences in books, music or movies respectively[1]. Hence, inferring domain- and application-specific user profile information from (external) Social Web activities is a difficult task.

In this paper, we propose a generic approach to constructing user profiles based on textual user-generated content on the Social Web. The main contributions of our work can be summarized as follows.

- We introduce GeniUS[2] – a generic library that can generate customized user profiles based on text messages people post on the Social Web.
- GeniUS enriches user data with semantic information and allows for the generation of meaningful RDF-based profiles that are well-connected to the Linked Open Data cloud and therefore better support interoperability between applications that aim for personalization.
- We apply the GeniUS library on a big Twitter dataset and demonstrate how the user profile construction can be customized to serve a given application domain.
- We evaluate the quality of the generated user profiles in context of different recommender systems and show that domain-specific profiles constructed by GeniUS allow for better personalization than generic Social Web profiles.

In the subsequent sections, we will summarize related work (Section [2]) and introduce the GeniUS framework for topic and user modeling on the Social Semantic Web (Section [3]). Analysis and evaluation are presented in Section [4] and Section [5] respectively before we conclude in Section [6].

¹ The tweet actually refers to the science fiction novel “Boom!” by Mark Haddon.

² We make the GeniUS library publicly available via

<http://wis.ewi.tudelft.nl/tweetum/>

2 Related Work

In the last decade, some research efforts have been done on generic user modeling systems [5] and methods for sharing user models across application boundaries [6]. Mehta et al. introduced strategies for cross-system user modeling and personalization [7] while Abel et al. investigated cross-system user modeling strategies on the Social Web [4]. With the advent of Semantic Web technologies, the interoperability of user profiles is enhanced. The General User Modeling Ontology (GUMO) provides a uniform representation of distributed user profiles [8]. Semantic Web vocabularies such as FOAF [9], SIOC [10], and the Weighted Interest vocabulary [11] further support the re-use of user profiles in different applications. In this paper, we will make use of these existing standards to facilitate usage and integration of GeniUS by client applications.

User modeling and personalization on the Social Web have been studied in the past years as well. Firan et al. defined tag-based profiles [12] to describe users' behavior in the context of music recommendations. Sen et al. refer to tag-based recommender systems as tagommenders and compare different strategies that infer interests into items via the users' interests into tags [13]. Cai et al. exploit interests into tags to personalize search in tagging systems [14]. However, a shortcoming of those approaches is that the semantics of user profiles are not explicitly defined as tags suffer from ambiguity and synonymy. Our GeniUS library disambiguates the meaning of terms that people post in their status messages and generates semantically well-defined user profiles that can be re-used in various application contexts.

Understanding the semantics of unstructured content on the Social Web is a non-trivial task, especially for the short status messages such as Twitter messages or Facebook status messages. Laniada and Mika defined metrics to characterize hashtags – words that start with “#” – to help assessing hashtags as strong representative of Twitter messages (tweets) [15]. Rowe et al. exploit contextual information to enrich the semantics of tweets by relating tweets to conference events [16]. In previous work, we proposed methods for understanding the semantics of tweets by discovering correlations between tweets and news articles [17] and further evaluated the benefits of this enrichment strategy for personalized news recommendations [3]. In this paper, we go one step further and introduce a library for semantic enrichment of social data to facilitate topic and user modeling, which is neither limited to a specific data source nor limited to a specific application context. Instead, the proposed GeniUS library allows for customized user modeling in different application domains.

3 GeniUS: A Generic Library for Topic and User Modeling

GeniUS is a generic library for topic and user modeling on the Social Semantic Web. Given textual user-generated content, GeniUS constructs semantic profiles that summarize the content of unstructured user data. With GeniUS, we

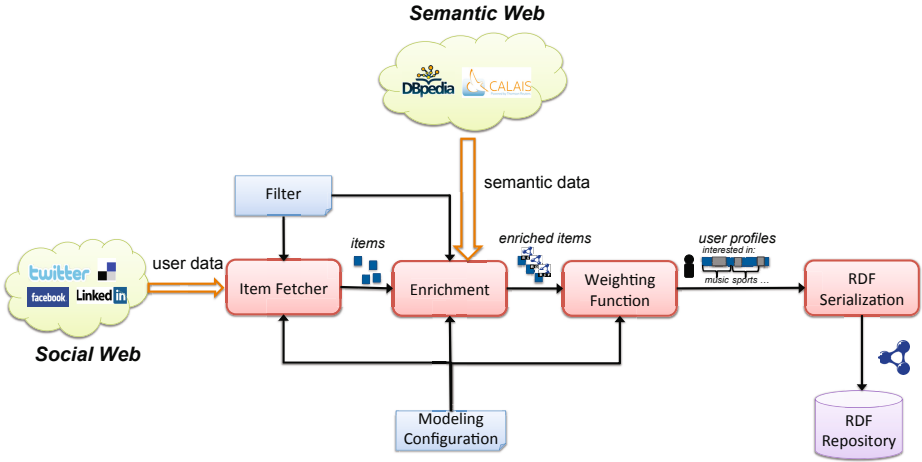


Fig. 1. The architecture of GeniUS Library

aim to (i) provide a flexible and extensible library that is able to serve different applications; (ii) produce semantically meaningful profiles to enhance the interoperability of profiles between applications; and (iii) customize the construction of user profiles according to the information needs of different applications. In this section, we first describe the architecture of GeniUS, including its four main modules, and then demonstrate how GeniUS can be applied to generate domain-specific user profiles.

3.1 Architecture of GeniUS

The architecture of GeniUS is presented in Figure 1. It is composed of four sequential modules, which process the given Social Web content to construct the profile. It consists of the *Item Fetcher*, *Enrichment*, *Weighting Function*, and *RDF Serialization*. Moreover, there are means for customizing the profile construction based on the demands of the application that is using GeniUS to obtain profiles: *Modeling Configuration* and *Filter*.

Item Fetcher. Many Social Web services provide APIs that allow for collecting data to serve external applications. For example, Twitter exposes its data through the Twitter Streaming and Search API³. The main function of *Item Fetcher* is to collect raw content, either directly from the Social Web or from a local repository. Topic profiles can be generated if the item fetcher is configured – using keyword or SPARQL queries – to collect data that refers to a given topic while user profiles are generated if the fetched posts were published by the same user. We transform and represent the raw content

³ <https://dev.twitter.com/>

based on a structured data model using the Semantically Interlinked Online Communication (SIOC) ontology [10]. Moreover, client applications can feed GeniUS with any types of SIOC items ranging from social bookmarking posts to (micro-)blog posts. The SIOC ontology provides a broad range of vocabulary concepts to describe information from the Social Web so that the *Item Fetcher* is highly flexible regarding the type of content it can handle. For example, to conduct our analysis and experiments on leveraging Twitter messages with GeniUS (see Section 4 and Section 5), we adopt *sioc:Post* to represent the tweets and *sioc:UserAccount* to describe the user account via which the messages were published.

Enrichment. To better understand the semantics of the content collected via the *Item Fetcher*, we further extract relevant concepts from the textual content. This step is accomplished by using existing services. In particular, GeniUS provides adaptors for Zemanta⁴ and SpotLight⁵ [18]. In this paper, we use SpotLight to extract DBpedia concepts from tweets. Each extracted concept is identified with a unique, resolvable URI so that the meaning of a concept is well-defined and the semantically enriched Twitter posts as well as the generated profiles are well-connected to the Linked Open Data cloud⁶.

Weighting Function. One of the main features of GeniUS is user profile construction for representing users' preferences and interests. Those types of profiles are essential for applications that aim for personalization. GeniUS mainly adopts the Vector Space Model to represent users' interests, i.e. a user profile is thus a set of weighted concepts. Therefore, we utilize the Weighted Interests Vocabulary and Weighting Ontology [11] as data model to represent user interest profiles.

The GeniUS library allows for different weighting functions to measure the importance and popularity of concepts in a profile ranging from straightforward strategies such as concept frequency (count the number of messages that refer to a concept) to more sophisticated strategies that compute a weight as a function of time (e.g. recently mentioned concepts are weighted stronger). Furthermore, client applications can specify their own weighting functions to customize the profile generation.

RDF Serialization. The constructed user profiles can be outputted as RDF using FOAF [9] in combination with the Weighted Interest Vocabulary. Profiles can also be stored in an RDF repository. Client applications or end-users can thus retrieve RDF-based profiles from the repository and perform sophisticated RDF queries over the profiles. At the moment, GeniUS provides adapters for the Sesame RDF repository⁷ to store the RDF profiles and allow for SPARQL queries.

In addition to the four modules mentioned above, which process the raw content and construct topic and user profiles, GeniUS also provides two configuration

⁴ <http://developer.zemanta.com/>

⁵ <http://dbpedia.org/spotlight>

⁶ <http://www4.wiwiw.fu-berlin.de/lodcloud/state/>

⁷ <http://www.openrdf.org/>

modules to enable a flexible modeling process that is required in order to generate domain- and application-specific profiles.

Modeling Configuration. The aforementioned four GeniUS modules are exposed as interfaces (in Java) so that developers can easily extend GeniUS and implement new functions based on their needs. For example, we implemented time-sensitive weighting functions [19] for applications that require more recent and dynamic characteristics of user profiles. The *Modeling Configuration* is used to configure which implementation for each module should be used in a GeniUS modeling process. With different combinations of module implementations, GeniUS has a variety of modeling alternatives to adapt to different applications.

Filter. With the *Filter* feature, GeniUS is able to filter out irrelevant profile information and can construct user profiles that represent certain characteristics of a user. We currently have implemented three types of filters: (i) filtering based on temporal constraints, (ii) keyword-based filtering, and (iii) semantic filtering. The first strategy can filter out content collected via the *Item Fetcher* or can prevent the *Item Fetcher* from collecting items that do not fulfill the given temporal constraints. For example, one can restrict the Twitter message collection process to tweets that were published within a certain period of time. The second filtering module can filter out items that do not contain a given set of keywords or specifically collect items that match the given keywords. With the enrichment of the user-generated items with meaningful concepts, GeniUS can also perform semantic filtering to generate customized profiles that characterize a user in context of a specific domain. In the subsequent section, we will reveal how we use SPARQL queries as semantic filters on the semantically enriched items to filter out irrelevant noise before constructing the actual (weighted interest) profile.

3.2 Domain-Specific User Profile Construction Using GENIUS

By utilizing filtering functionality, the library is able to build flexible user profiles for different application domains on demand. The constructed user profiles are represented in RDF with well-defined semantics.

In this paper, we use Twitter as an example to illustrate and evaluate the domain-specific profile construction. Given a short Twitter post like:

Awesome, love the new Garageband for iPad <http://is.gd/SJqVav> #apple

we collect the content of this message and additional information such as the user identifier of the creator and the creation time via the Twitter Streaming API. Semantic Web vocabularies such as SIOC, Dublin Core and FOAF are applied to represent the Twitter message. We have also built a parser to process the content of the message and extract hashtags and URLs that are mentioned in a tweet. The extracted hashtags are identified using TagDef [8] as depicted in the following code snippet.

⁸ <http://www.tagdef.com>


```

@prefix sioc: <http://rdfs.org/sioc/spec/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix tagdef: <http://tagdef.com/> .

<http://twitter.com/bob/status/73748435752333312>
  a <sioc:Post> ;
  dcterms:created "2011-05-26T15:52:51+00:00" ;
  sioc:has_creator <http://twitter.com/bob> ;
  sioc:content "Awesome, love the new Garageband for iPad http://is.gd/SJqVav #apple" ;
  sioc:links_to <http://is.gd/SJqVav> ;
  sioc:has_topic tagdef:apple .

```

The message is thus represented as *sioc:Post* and specifies metadata (e.g. *dcterms:created*, *sioc:has_creator*) as well as basic information about the content (e.g. *sioc:content*, *sioc:links_to*). *sioc:has_topic* is used to describe the semantic meaning of the content of the tweet. However, using TagDef does not allow for disambiguating the semantic meaning of the tweet and hashtag specifically. For example, *apple* may refer to the fruit or to the technology company. Hence, further semantic enrichment is required to specify the meaning of a Twitter message more accurately. Therefore, we perform named entity recognition and identify DBpedia concepts in tweets (using disambiguation functionality provided by DBpedia spotlight [18]). This allows us to further describe the topic of the tweet with further RDF statements using again the *sioc:has_topic* property:

```

@prefix sioc: <http://rdfs.org/sioc/spec/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix tagdef: <http://tagdef.com/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .

<http://twitter.com/bob/status/73748435752333312>
  a <sioc:Post> ;
  dcterms:created "2011-05-26T15:52:51+00:00" ;
  sioc:has_creator <http://twitter.com/bob> ;
  sioc:content "Awesome, love the new Garageband for iPad http://is.gd/SJqVav #apple" ;
  sioc:links_to <http://is.gd/SJqVav> ;
  sioc:has_topic tagdef:apple ;
  sioc:has_topic dbpedia:Apple_Inc. ;
  sioc:has_topic dbpedia:GarageBand ;
  sioc:has_topic dbpedia:iPad .

```

Given the semantic enrichment and the inferred additional RDF statements, we can disambiguate the meaning of the Twitter message: it refers to a software product (*dbpedia:GarageBand*) developed by the Apple company (*dbpedia:Apple_Inc.*) that is now available for the *iPad* device (*dbpedia:iPad*). By following the DBpedia URIs, applications can obtain further background information such as type information (e.g. *dbpedia:GarageBand* is of type *dbo:Software* and *yago:AudioEditors*⁹) or a list of persons that are involved in Apple (e.g. *dbo:keyPerson*).

With the enriched concepts, GeniUS constructs profiles using a given weighting scheme. FOAF and the Weighted Interests Vocabulary are applied to describe the user and her preferences and interests into topics (based on the concepts that are referenced from the tweets). Since people publish Twitter messages on various different subjects, a generic approach, which considers all kinds of concepts

⁹ Here, *dbo* and *yago* refer to the DBpedia ontology (<http://dbpedia.org/ontology/>) and Yago ontology (<http://dbpedia.org/class/yago/>) respectively.

that are referenced from the tweets, produces user profiles that contain a variety of topics. In the following example, the extract of the complete profile thus specifies topics of interests from different domains such as the music, software or movie domain:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix wi: <http://purl.org/ontology/wi/core#> .
@prefix wo: <http://purl.org/ontology/wo/core#> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix genius: <http://persweb.org/genius#> .
<http://twitter.com/bob>
  a foaf:Person;
  wi:preference [
    a wi:WeightedInterest ;
    wi:topic dbpedia:Jazz ;
    wo:weight [
      a wo:Weight ;
      wo:weight_value 0.5889 ;
      wo:scale genius:Scale ]
    ] ;
  wi:preference [
    a wi:WeightedInterest ;
    wi:topic dbpedia:Second_Life ;
    wo:weight [
      a wo:Weight ;
      wo:weight_value 0.3114 ;
      wo:scale genius:Scale ]
    ] ;
  wi:preference [
    a wi:WeightedInterest ;
    wi:topic dbpedia:Short_film ;
    wo:weight [
      a wo:Weight ;
      wo:weight_value 0.3333 ;
      wo:scale genius:Scale ]
    ] ;
  wi:preference [
    a wi:WeightedInterest ;
    wi:topic dbpedia:GarageBand ;
    wo:weight [
      a wo:Weight ;
      wo:weight_value 0.1638 ;
      wo:scale genius:Scale ]
    ] ; ...
```

The above profile depicts that the user is interested in jazz music (*dbpedia:Jazz*), short movies (*dbpedia:Short_film*) and software products (e.g. *dbpedia:Second_Life*). The higher the weight the higher the inferred interest in a concept. When applying the constructed profiles for a specific application domain, a drawback of such a complete profile is that it lists also concepts that are possibly not relevant in the application context. For example, if a system aims to recommend software products to the above user then concepts such as *dbpedia:Jazz* or *dbpedia:Short_Film* might not add value to the profile while statements about the user's preference into software are more important. Utilizing the semantic filtering feature of GeniUS, application developers can specify a SPARQL query that describes what kind of topic-based profile a client application is seeking for:

```
SELECT DISTINCT ?t WHERE {
  ? <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Software> }
```

Given such a SPARQL query, GeniUS will generate a customized profile where the concepts that do not belong to the software domain are filtered out (see below). The weight of the remaining concepts can be re-adjusted as well, for example, by normalizing the filtered profile.

```
<http://twitter.com/bob>
  a foaf:Person;
  wi:preference [
    a wi:WeightedInterest ;
    wi:topic dbpedia:Second_Life ;
    wo:weight [
      a wo:Weight ;
      wo:weight_value 0.4101 ;
      wo:scale genius:Scale
    ]
  ] ;
  wi:preference [
    a wi:WeightedInterest ;
    wi:topic dbpedia:GarageBand ;
    wo:weight [
      a wo:Weight ;
      wo:weight_value 0.2158 ;
      wo:scale genius:Scale
    ]
  ] ; ...
```

Using semantic filtering, applications can therefore utilize GeniUS to generate profiles that are well-adapted to their domain of interest. In the next two sections, we will conduct further analysis and experiments to show the quality of such customized profiles in the context of recommender systems in different application domains.

4 Analysis of Domain-Specific User Profile Construction

To understand the characteristics of user profiles constructed with GeniUS, we conducted an analysis on a large Twitter dataset. In our analysis, we investigate the characteristics of (i) complete Twitter-based profiles and (ii) six domain-specific types of profiles that were filtered using the semantic filtering method of GeniUS.

4.1 Data Collection

For our analysis, we monitored 73 Twitter users of the Social Handle Archive¹⁰ (SoHarc) over a period of more than six months (from January 1st 2011 to July 7th). SoHarc lists profiles of researchers who are active in computer science and e-learning research in particular. Therefore, we ensured that there were no spam users in our dataset. Using the Twitter Streaming API via the *Item Fetcher* of GeniUS, we collected all public Twitter messages that these users published during the observation period. Overall, we thereby obtained 40,822 tweets. The seven most active users posted more than 1000 tweets while two users were almost inactive and published less than 10 Twitter messages (see Figure 2). We

¹⁰ <http://soharc.upb.de/>

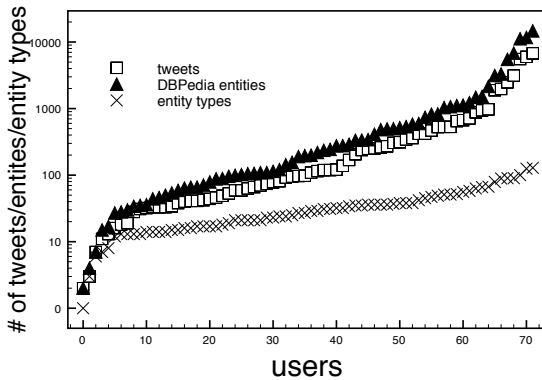


Fig. 2. Number of tweets, DBpedia entities and entity types per user

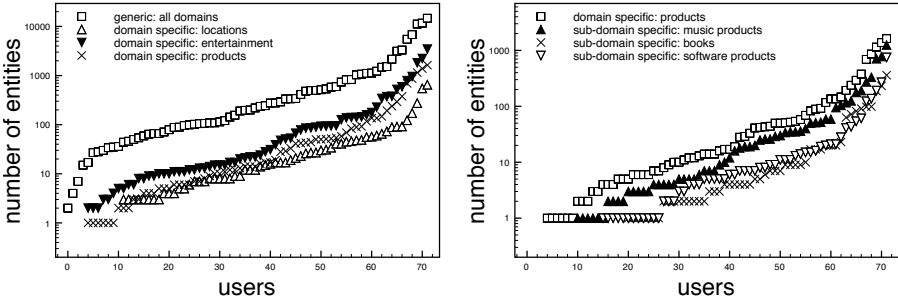
processed all Twitter messages with the semantic enrichment component of GeniUS which we configured so that DBpedia Spotlight was used as named entity recognition service as it allows for higher precision and recall than Alchemy or Zemanta (Mendes et al. report, for example, on precision of around 80% for disambiguating entities) [18]. Furthermore, we utilized the concept frequency as weighting scheme to compute weights for the entities of interest in the corresponding user profiles. Although all users are from the same computer science community, the topics about which they publish tweets show great variety as our analysis will reveal.

4.2 Results

In Figure 2, we plot the number of tweets and enriched DBpedia entities for each user. On average, each user published 567.0 Twitter messages and referred, according to the entity extraction module, to 1097.1 DBpedia entities. 59 of the users (82%) published more than 50 tweets during the observation period and also referred to more than 50 entities. And for each tweet, we extracted on average 1.9 entities. Each entity is identified by a unique URI. Therefore, we further retrieved – by resolving the URI – the types of the entities as specified in the corresponding DBpedia entry. The number of distinct types of entities to which a user refers to in her tweets is listed in Figure 2 as well. The average number of distinct types per user profile is 35.0 which indicates that there is a potential to generate different domain-specific profiles for a given user when categorizing entities of interest according to their types.

To construct application domain based user profiles, we group the types of entities based on the DBpedia ontology¹¹ into several domains. In particular, we select three main domains for the analysis and further experiments: *location*, *entertainment* and *product*. Based on the hierarchies defined in the DBpedia ontology, we further derive three sub-domains from the product domain: *music products*, *books* and *software products*. In our evaluation, we classify

¹¹ <http://wiki.dbpedia.org/Ontology>



(a) Comparison of generic and domain-specific user profile construction

(b) Comparison of domain and sub-domain specific user profile construction

Fig. 3. Comparison of different strategies for user profile construction

items into these domains and test what kind of user profiles do best serve the domain-specific recommender system (recommender in the entertainment domain, book recommender etc.).

Using the semantic filter described in Section 3.2, we construct the (sub-) domain-specific user profiles. Figure 3 characterizes the corresponding profiles and shows the number of entities per user profile for the different types of profile construction strategies. In Figure 3(a), we compare the generic strategy, which utilizes all kinds of entities (no filtering), and the domain specific strategies which filter the profiles so that they contain entities related to the location, entertainment and product domain respectively (semantic filtering). On average, there are 358.0 entities per user profile that belong to one of the three domains. Among them, 12.1% of the entities are categorized as locations (e.g. cities, countries and other places), 54.8% as being related to entertainment (e.g. sport, cultural events) and 33.1% as products (e.g. music albums, books, magazines, software). Some of these profiles are rather sparse. For example, for approximately 30 users, the location-based and product-related profiles contain less than 10 entities. The continuous difference between the size of the generic profiles and the domain-specific profiles indicates that all users reveal interests in different types of domains in their Twitter activities. Similarly, we observe in Figure 3(b) that also the domain-specific profiles related to products feature variety. When further filtering these profiles to obtain profiles that specify interests in music products, books and software products, one still obtains reasonably sized user profiles. Here, interests into music products can be inferred best. On average, 58% of the products mentioned by a user can be classified as music products (e.g. albums, songs) while 18.3% and 23.7% of the products are related to books (including newspapers and magazines) and software products respectively.

Our analysis thus shows that Twitter-based profiles reveal different types of interests of a user. Hence, there is potential to adapt Twitter-based profiles to different application domains. Figure 3 shows that we succeed in generating domain-specific profiles for the great majority of the users. The more specific the

domain the smaller the profiles. Our hypothesis is that the stronger we adapt profiles to the given application domain – i.e. the more restrictive the filtering of the profiles – the better the performance of the corresponding application that consumes the profiles. In the subsequent section, we will investigate whether this hypothesis holds.

5 Evaluation of Domain-Specific User Profile Construction for Recommendation Systems

To test our hypothesis and to evaluate the quality of the Twitter-based user profiles that are created by different user modeling strategies of GeniUS, we apply the generated user profiles in different domain-specific recommendation systems and answer the following research questions:

1. Are the domain-specific user modeling strategies provided by GeniUS beneficial for supporting recommendation systems in different domains? For example, are the sparse – but more focused – user profiles more appropriate than the complete, unfiltered profiles?
2. How does the performance vary between the different domains? For example, for what domains does the Twitter-based user modeling by GeniUS work best?

5.1 Experimental Setup

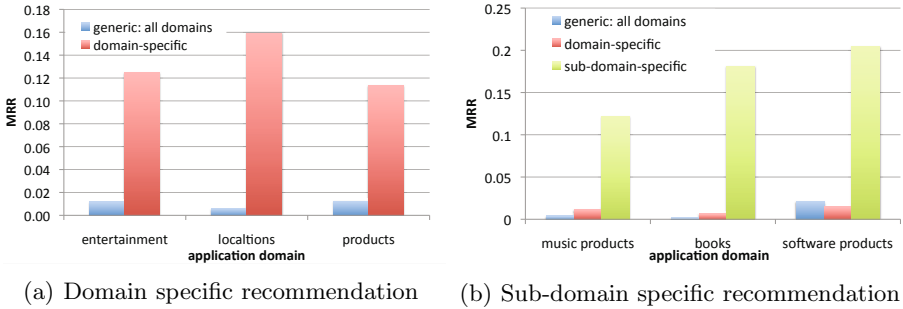
In our evaluation, we test the user profiles created by the different GeniUS strategies in context of different tweet recommendation systems. The main goal of a domain-specific recommender is to recommend tweets to a user that are relevant to the given (sub-)domain and relevant to the user. As we are mainly interested in comparing the quality of user profiles constructed via the different user modeling strategies, we implemented a lightweight content-based recommendation algorithm that we applied in the different application contexts. The algorithm recommends items based on their cosine similarity with a given user, i.e. the more similar an item to a user the higher it will appear in the recommendation ranking.

Given a profile of a user u and a set of candidate items C , we thus transform the list of weighted interests included in the user profile into a vector space model representation $\mathbf{p}(u)$ and represent the candidate items in the same way: $C = \{\mathbf{p}(c_1), \dots, \mathbf{p}(c_n)\}$. The weight of an entity in $\mathbf{p}(c_i)$ corresponds to the occurrence frequency of an entity within a tweet (usually this is either 1 or 0). Given the normalized user and item vector representations, the algorithm computes cosine similarity $\text{sim}_{\text{cosine}}(\mathbf{p}(u), \mathbf{p}(c_i))$ and orders the items according to their similarity score.

For our experiments, we consider the last month of our observation period (see Section 4.1) as the time frame for computing recommendations. For each of the six domains that we analyzed in Section 4, we deployed a recommendation system

Table 1. The average number of relevant items and candidate items for different recommendations

application domain	broad domains			product sub-domains		
	entertainment	locations	products	music	books	software
average number of candidate items	1587.0	151.0	1207.0	756.0	237.0	254.0
average number of relevant items	70.0	13.4	49.6	40.9	16.4	20.3

**Fig. 4.** Results of recommendation experiment

that used the algorithm described above as basis. Hence, the recommendation quality is solely influenced by the user modeling strategy for constructing $p(u)$. The ground truth of tweets which we consider as *relevant* to a specific user u in a particular application domain is given by those messages that were actually posted by u during the recommendation period and also contain at least one concept that belongs to the specified application domain. Hence, we remove all user information from the candidate tweets and try to assign the tweets to the right users by utilizing the user profiles that are constructed based on the tweets a user posted before the start of the recommendation period. The quality of recommendations is measured by means of *MRR* (Mean Reciprocal Rank) which indicates at what rank the first item relevant to the user occurs on average.

The set of candidate items are those tweets that were published during the recommendation period and refer to at least one concept of the application domain of the recommendation system. The average number of relevant items per user and the number of candidate items in each application domain are listed in Table 1. For example, for the broader domains one can infer that the product domain is most challenging: the probability of randomly selecting a relevant item is 0.041 ($= 49.6 / 1207.0$) in contrast to 0.045 and 0.089 for the domains of entertainment and location respectively.

5.2 Results

The results of the recommendation experiments in the six different domains are summarized in Figure 4 and answer the research question raised at the beginning

of this section. Figure 4(a) shows the quality of the recommendations in terms of MRR for the three broader domains: locations, entertainment and products. We compare the recommendation performances that were achieved based on the generic user modeling strategy, which does not make use of the semantic filtering functionality of GeniUS, and the domain-specific user modeling strategy, which filters out concepts from the profiles that are not related to the actual domain.

The domain-specific strategy consistently performs much better than the generic strategy. While the domain-specific strategy achieves, on average, an MRR of 0.13 across the three different domains, the generic strategy performs poorly with 0.01. The domain-specific strategy produces with 0.16 regarding MRR the best results in context of the location-related recommendation system which is according to the proportion of relevant items per user the least challenging domain (see Table II). In contrast, the generic strategy achieves only an MRR of 0.006. Within the context of product recommendations, the domain-specific profile construction method results in a ten times higher MRR and therefore outperforms the generic strategy clearly.

Similar results can be observed in the more narrow domains of music, book and software product recommendation systems. For the music and book domain, one can see that the recommendation quality increases the more specific the profiles are. In the music recommendation setting, the generic strategy fails with 0.004 regarding MRR, the product-specific profile improves the recommendation slightly (MRR: 0.01) and the profiles, which are specifically filtered for the music domain (*sub-domain-specific*), perform best and allow for an MRR of 0.12. The low performance of the generic user modeling strategy can be explained by noise that is introduced by entities that co-occur in tweets. For example, given that a user u tweets “Heading to Italy”, GeniUS will infer that u has some interest into the concept *dbpedia:Italy*. The domain-specific strategy will filter out this interest when recommendations are computed in the music domain while the generic strategy will keep the information. Hence, given a candidate tweet such as “Justin Bieber concert in Italy #music”, the domain-specific user modeling strategy will not cause this item to be recommended to u unless u showed interest into music of Justin Bieber in some of her other tweets that she published before the recommendation period. The generic user modeling strategy however will indicate to the recommender system that the given tweet might be of interest to the user because it mentions the concept *Italy* in which the user proved to be concerned with in the past. For the software domain this effect caused by the noise in the generic user profiles seems to be lower. However, again we observe that the sub-domain-specific user modeling strategy outperforms the other strategies clearly and allows for an MRR of 0.22.

Hence, regarding the research questions raised above, we can conclude that (1) domain-specific user modeling strategies provided by GeniUS allow for a tremendous improvement of the recommendation quality. Semantic filtering of the user profiles seems to remove noise and therefore allows us to adapt and optimize the user profile construction to the target domain. Moreover, we see that (2) the performance improvements are consistent throughout the different

domains. The user modeling quality varies only slightly between the different domains. Furthermore, the performance does not seem to be influenced strongly by the size of the user profiles as a comparison of Figure 3 and Figure 4 reveals. With GeniUS, we thus succeed in generating domain-specific user profiles that allow for optimizing recommendation performance across different domains.

6 Conclusion

In this paper, we introduced GeniUS, a generic topic and user modeling library for the Social Semantic Web. Given (status) messages from services such as Facebook or Twitter, GeniUS creates RDF-based representations that describe the semantic meaning of these messages. Based on the semantically enriched user data, GeniUS provides different strategies for the creation of user interest profiles and provides means to semantically filter those profiles so that they adapt to a given application domain.

Our analysis based on Twitter status messages published by users during a period of several months showed that we succeed in generating profiles for different domains. To test the quality of the Twitter-based user profiles, we conducted recommendation experiments in six different domains and revealed that the domain-specific user modeling strategies, which filter user profiles and limit the concepts in the profiles to concepts related to the given application domain, allow clearly for the best performance.

In future work, we would like to apply GeniUS also on other datasets different from Twitter. This will allow us to investigate what type of Social Web services are most valuable for creating profiles for different types of applications.

Acknowledgements. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. ICT 257831 (ImREAL project) and the Marie Curie action IRSES under grant agreement No. 24761 (Net2 project).

References

1. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web, pp. 851–860. ACM, Raleigh (2010)
2. Chen, J., Nairn, R., Chi, E.H.: Speak Little and Well: Recommending Conversations in Online Social Streams. In: Proceedings of the 29th International Conference on Human Factors in Computing Systems, CHI 2011, pp. 217–226. ACM, Vancouver (2011)
3. Abel, F., Gao, Q., Houben, G.J., Tao, K.: Analyzing User Modeling on Twitter for Personalized News Recommendations. In: Konstan, J.A., Conejo, R., Marzo, J.L., Oliver, N. (eds.) UMAP 2011. LNCS, vol. 6787, pp. 1–12. Springer, Heidelberg (2011)

4. Abel, F., Herder, E., Houben, G.J., Henze, N., Krause, D.: Cross-system User Modeling and Personalization on the Social Web. User Modeling and User-Adapted Interaction (UMUAI), Special Issue on Personalization in Social Web Systems (to appear), <http://wis.ewi.tudelft.nl/papers/2011-umuai-cross-system-um.pdf>
5. Kobsa, A.: Generic user modeling systems. User Modeling and User-Adapted Interaction 11(1-2), 49–63 (2001)
6. Berkovsky, S., Kuflik, T., Ricci, F.: Cross-Domain Mediation in Collaborative Filtering. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 355–359. Springer, Heidelberg (2007)
7. Mehta, B., Niederee, C., Stewart, A.: Towards cross-system personalization. In: International Conference on Universal Access in Human-Computer Interaction, Las Vegas, Nevada, USA (UAHCI 2005). Lawrence Erlbaum Associates (2005)
8. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: Gumo - The General user Model Ontology. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 428–432. Springer, Heidelberg (2005)
9. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.91. Namespace document, FOAF Project (November 2007), <http://xmlns.com/foaf/0.1/>
10. Bojars, U., Breslin, J.G.: SIOC Core Ontology Specification. Namespace document, DERI, NUI Galway (January 2009), <http://rdfs.org/sioc/spec/>
11. Brickley, D., Miller, L., Inkster, T., Zeng, Y., Wang, Y., Damjanovic, D., Huang, Z., Kinsella, S., Breslin, J., Ferris, B.: The Weighted Interests Vocabulary 0.5. Namespace document, Sourceforge (September 2010)
12. Firan, C.S., Nejdil, W., Paiu, R.: The Benefit of Using Tag-based Profiles. In: Proceedings of the 2007 Latin American Web Conference (LA-WEB 2007), pp. 32–41. IEEE Computer Society, Washington, DC, USA (2007)
13. Sen, S., Vig, J., Riedl, J.: Tagommenders: connecting users to items through tags. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, pp. 671–680. ACM, Madrid (2009)
14. Cai, Y., Li, Q.: Personalized search by tag-based user profile and resource profile in collaborative tagging systems. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010, pp. 969–978. ACM, Toronto (2010)
15. Laniado, D., Mika, P.: Making Sense of Twitter. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 470–485. Springer, Heidelberg (2010)
16. Rowe, M., Stankovic, M., Laublet, P.: Mapping Tweets to Conference Talks: A Goldmine for Semantics. In: Social Data on the Web Workshop at the 9th International Semantic Web Conference (ISWC), Shanghai, China, vol. 664 (2010), CEUR-WS.org
17. Abel, F., Gao, Q., Houben, G.J., Tao, K.: Semantic Enrichment of Twitter Posts for user Profile Construction on the Social Web. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011. LNCS, vol. 6644, pp. 375–389. Springer, Heidelberg (2011)
18. Mendes, P.N., Jakob, M., Garcia-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems (I-Semantics), Graz, Austria, pp. 1–8 (September 2011)
19. Gao, Q., Abel, F., Houben, G.J., Tao, K.: Interweaving trend and user modeling for personalized news recommendations. In: Proceeding of the 2011 International Conference on Web Intelligence Web (WI 2011), pp. 100–103. IEEE Press, Lyon (2011)

Enhancing Source Selection for Live Queries over Linked Data via Query Log Mining^{*}

Yuan Tian¹, Jürgen Umbrich², and Yong Yu¹

¹ Shanghai Jiao Tong University, 800 Dongchuan Rd., Shanghai, China
{tian,yyu}@apex.sjtu.edu.cn

² Digital Enterprise Research Institute, National University of Ireland, Galway
juergen.umbrich@deri.org

Abstract. Traditionally, Linked Data query engines execute SPARQL queries over a materialised repository which on the one hand, guarantees fast query answering but on the other hand requires time and resource consuming preprocessing steps. In addition, the materialised repositories have to deal with the ongoing challenge of maintaining the index which is – given the size of the Web – practically unfeasible. Thus, the results for a given SPARQL query are potentially out-dated. Recent approaches address the result freshness problem by answering a given query directly over dereferenced query relevant Web documents. Our work investigate the problem of an efficient selection of query relevant sources under this context. As a part of query optimization, source selection tries to estimate the minimum number of sources accessed in order to answer a query. We propose to summarize and index sources based on frequently appearing query graph patterns mined from query logs. We verify the applicability of our approach and empirically show that our approach significantly reduces the number of relevant sources estimated while keeping the overhead low.

1 Introduction

Data published according to the Linked Data principles can be seen as one distributed and decentralised Web database. The four Linked Data principles [1] guarantee that 1) all data items are represent with unique identifiers in form of URIs, 2) the URIs are dereferenceable HTTP URIs and 3) an HTTP Get operation returns more information represented in RDF [13]. The fourth principle, reuse of and links to identifiers, connects the decentralised published information into a global and open information network. However, Linked Data poses a series of unprecedented challenges for processing queries over the Web content. Typically, the results for a query are aggregated from hundreds of sources published by autonomous data providers. Further, a majority of the sources offer no abilities of executing complex queries and the serve just as a container for the data (e.g., a document). Typical SPARQL query engines operating over these

^{*} The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

decentralized cross-domains can be categorized into the following three possible implementations [4,5].

- *Centralized repository* or data warehousing approaches collect data directly from the Web by means of crawling or downloading RDF dumps. The collected data can be preprocessed and is eventually indexed with optimised data structures for fast and efficient query answering.
- *SPARQL endpoint mediation* decomposes the query into subqueries that can be dispatched to respective SPARQL endpoints of the data sources. After the results are returned, they are assembled and processed by the federator.
- *Online query processing* or dynamic query processing approaches evaluate the queries directly over the dereferenced Web content by exploiting the Linked Data principles or in addition with source selection index structures. These index structures typically contain a summary about the source content as opposed to centralized repositories which store the detailed content.

The drawback of centralize approaches is that the full materialized data has to be crawled and saved beforehand which is a time and resource consuming complex task. These materialized approaches aim to serve a broad spectrum of users and scenarios. However, for dedicated applications or a limited number of users, much unnecessary data is collected. Another disadvantage is that a large part of the data such as sensor readings or microblog posts are constantly at change [18] and index maintenance becomes a very challenging task. Hence, querying against local snapshots defies the possibility of retrieving latest results. The configuration of SPARQL endpoint mediation requires data sources offering a SPARQL endpoints. However, at current stage, most data sources do not have a full SPARQL processing capability. Moreover, one goal of Linked Data is to alleviate the burden of building sophisticated query engines for publishing semantic data. The last group of approaches is a compromise of the first two. It simultaneously supports dynamic query processing and requires no complicated query capability from the data source.

The scope of this paper is to investigate an adaptive source selection approach for the last scenario. We propose to store information about a source based on frequent query patterns mined from query logs. This dynamic approach adaptively tailors the indexed information to the previous user queries as opposed to state-of-the-art source selection approaches which neglect the query history and index information based on fixed heuristics or on characteristics of the data set. In detail, we contribute to this line of research in the following aspects:

1. We consider the problem of source selection from a different angle. Starting from query logs, we mine frequent BGPs that potentially increase the overall performance gain.
2. Compared to previous work [5,6], we extend the index capability from triple patterns and one-join patterns to tree-shape BGPs, which results in a further reduced number of relevant sources.
3. We include a detailed empirical study on the efficacy of our approach. We show summaries of mined BGPs considerably reduced the relevant sources estimated. Furthermore, it demonstrated better effectiveness over complex queries.

This paper is structured as follows. First, we give an overview of our approach and explain the concepts in Section 2. Following that, in Section 3 we propose a framework for summarizing source information. And in Section 4 we discuss how frequent patterns can be mined. Section 5 introduces our source selection algorithm. Empirical studies and analysis are given in Section 6. Finally, Section 7 reviews related work and Section 8 concludes our work.

2 Overview of Source Selection

This section introduces our approach for an efficient execution of SPARQL queries over (online) Web data with an adaptive source selection index based on frequently appearing graph pattern. In our approach, we exclusively focus on the evaluation of SPARQL queries consisting of only basic graph pattern. Considering that SPARQL query language is based on graph pattern matching, more complex SPARQL graph patterns can be decomposed into 5 types of smaller graph patterns: Basic Graph Patterns, Group Graph Patterns, Optional Graph Patterns, Alternative Graph Pattern and Patterns on Named Graphs [16]. As the fundamental building block of these pattern types is the Basic Graph Pattern (BGP) and extending our approach to full SPARQL queries is left as future work. As in Definition 1, a BGP is basically a set of connected triple patterns to which a solution binds a statement.

Definition 1 (Triple Pattern & Basic Graph Pattern). *A triple pattern tp is an RDF triple (s,p,o) with each component being a variable, a resource or a literal. A basic graph pattern bgp is a set of triple patterns $(bgp = \{tp_1, tp_2, tp_3, \dots, tp_n\})$.*

Query answering over a set of BGP is straight forward by combining the solution bindings for the BGPs to give the final results. In most of the SPARQL query engines, graph pattern matching problems are solved by joining the triple bindings. The same strategy applies to live SPARQL query engines in a distributed scenario except that bindings are evaluated over the dereferenced content of selected source. The bottle neck of these query engines lies in network bandwidth and latency. This signifies the importance of source selection as it reduces network round-trips and hence the cost of query execution.

A prevalent way to discard irrelevant sources is to generate data summaries [5]. When data is crawled from its sources, source information is calculated and saved. In this paper, we propose a more sophisticated approach which maintains source summaries for a set of mined frequent subgraph patterns from query logs that later could be taken to enhance source selection.

2.1 Query Relevant Sources

The evaluation of SPARQL queries over the Web requires to evaluate the queries directly over the retrieved content of Web sources. Ideally, one would like to

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX identica:<http://identi.ca/>
PREFIX sioc:<http://rdfs.org/sioc/ns/>

SELECT ?x ?y
WHERE {
  ?y foaf:accountProfilePage identica:robmyers .
  ?y foaf:accountName "robmyers" .
  ?x sioc:follows ?y .
}
    
```

Query 1.1. People following robmyers

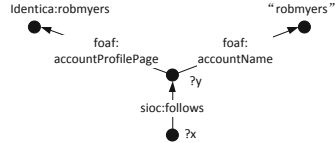


Fig. 1. Visualization of Query 1.1

retrieve only sources which directly contribute query solutions. Thus, query relevant sources for a BGP is defined as the set of sources which can answer the given query. Next, we formally define the set of query relevant sources with solution bindings as defined in [16].

Definition 2 (Relevant Sources). Denote the binding of triple pattern tp_i under solution $\mu : g \mapsto \mathcal{G}$ as $\mu(tp_i)$ where \mathcal{G} is the set of all statements in data graph. $S_g(\mu) = \bigcup_{tp_i \in g} s(\mu(tp_i))$ is called the relevant sources of solution μ with $s(\cdot)$ denoting the sources where bindings can be retrieved. Furthermore, $S(g) = \bigcup_{\mu_i \models g} S_g(\mu_i)$ where $\mu_i \models g$ means μ_i is a solution to g .

2.2 Source Selection

The source selection task is to select for a given query the query relevant sources. More formally, the source selection for a BGP g is the process of estimating $S(g)$. There exist various approaches and methods to estimate the set of query relevant sources. Basically, the source selection task can be done before the query execution and/or during the query execution (e.g., source selection for triple patterns). The former method is the preferred ones since the query relevant sources can be fetched efficiently in a parallel manner and sources containing redundant information can be eliminated. A straight forward solution is to use an approach that materializes and stores RDF statements and their corresponding sources. However, as we have already pointed out, these approaches required a significant amount of disk space given the size of the Web (number of sources) and the infinite set of BGPs. In addition, given the dynamic nature of Web documents, the materialized source selection approach has to face against the problem of index maintenance. Further, the source selection can falsely select query relevant sources if the index is not up-to-date. Solutions trying to overcome the resource requirements and the maintenance problems store only a proportion which require much less storage resources and are easier to maintain. However, in this way source selection potentially increases the number of false positives but we seek to reduce them.

Example 1. Let us consider Query 1.1 – a BGP query generated from our dataset – and its query graph as shown in Fig. 1. The query asks for the people who follow a person with an account name of "robmyers" and an identica profile page. Let us

Table 1. Solution bindings for example query Query [1.1](#)

	$?x$	$?y$	$S(\mu_i)$
μ_1	user:47287	user:1108	http://identi.ca/robmyers/foaf http://identi.ca/jargon/foaf
μ_2	user:9577	user:1108	http://identi.ca/robmyers/foaf http://identi.ca/mattl/foaf

assume that the execution of the query results in two answers $\{\mu_1, \mu_2\}$ as shown in Table [1](#). In addition, Table [1](#) contains the URIs of the query relevant documents for the two answers (cf. column $S(\mu_i)$). We have three documents which are query relevant. The first solution comes from <http://identi.ca/jargon/foaf> and <http://identi.ca/robmyers/foaf> while <http://identi.ca/mattl/foaf> and <http://identi.ca/robmyers/foaf> form the second solution. An ideal source selection would only estimate the three as source relevant.

3 Summarizing Source Information

Our source selection approach operates over an approximated index of graph patterns and their list of relevant sources. In addition, our index structure adaptively optimizes the accuracy and required space of the source selection by considering frequent query patterns and disregarding rare patterns. Initially, we start to summarise and index sources by simple triple patterns. The purpose of this approach is to overcome the cold start problem and we do not require any knowledge about complex index graph patterns. However, performing source selection only with triple patterns introduces a significant amount of false positives. With our adaptive mining approach we decrease the false positive selection ratio and further reduce the number of estimated query relevant sources significantly by summarizing sources based on patterns consisting of more than a single triple pattern. We get these complex query patterns from a frequent subgraph mining process over a set of known and/or executed queries. Our approach can be classified as an adaptive self-tuning approach. The source selection index self tunes its index patterns based on the query load of the past and thus adapts to the most frequent executed query types which is in most scenarios cannot be estimated.

We do not focus on the best possible data structure to index and search the query patterns. In contrary, we focus entirely on the mining parts of pattern and optimisation of the source selection. Thus, the only requirement for the underlying index structure is to support the following atomic operations:

- *lookup* : Given a BGP g , return its relevant sources $\mathcal{S}(g)$;
- *put* : Add an entry pair $(g, \mathcal{S}(g))$ to the index. If key g exists, it will be overwritten with the new entry;
- *remove* : Remove an entry in the index.

These operation are the basic operation for a key/value store. The *lookup* operation is a basic requirement enabling to retrieve pre-calculated summaries.

Leveraging this operation, our approach also supports the approximation of other BGPs whose entries do not exist as information of related patterns is needed. The detailed algorithm will be introduced shortly afterwards in Section 5. The *put* operation adds new entries to the index or overrides the values for existing keys. The *remove* operation is necessary to delete stale information from the index.

We use string values as the key values for the index structure. Each key is a mapping of a basic graph pattern p to a unique minimum depth-first-search (DFS) code by applying a mapping function $h(p)$. These mapping function $h(p)$ regards two isomorphic query graphs the same due to their identical semantics. Our concrete implementation of the function is based on the graph encoding technique proposed in gSpan [19] called DFS lexicographic order.

Generate DFS Subscripting. Following the path each node is visited in the depth-first search algorithm, a DFS tree can be constructed. We denote the node assigned with subscription i denoting the order it is for the first time discovered as v_i . v_i is discovered before v_j if $i < j$. The process slightly differs from [19] that we make the search following the edge directions due to the fact that SPARQL BGPs are directed graphs.

Identify Forward and Backward Edges. The triple pattern that connects nodes v_i and v_j with $i < j$ is denoted as (i, \cdot, j) . If $i < j$, it is called a forward edge. Otherwise, it is a backward edge.

Assign an Linear Order \prec to Edges. For two triple patterns $tp_1 = (i_1, \cdot, j_1)$ and $tp_2 = (i_2, \cdot, j_2)$, the order satisfies the following conditions: (a) if $i_1 = i_2$ and $j_1 < j_2$, $e_1 \prec e_2$; (b) if $i_1 < j_1$ and $j_1 = i_2$, $e_1 \prec e_2$; (c) if $e_1 \prec e_2$ and $e_2 \prec e_3$, $e_1 \prec e_3$.

DFS Code. We rank all triple patterns in the given BGP by order \prec and build the sequence (tp_i) , i.e. all triple patterns in (tp_i) subject to $tp_i \prec tp_{i+1}$. Each DFS tree t of a BGP g has a different such sequence so we denote it as $DFSC(g, t)$. Each element (i, \cdot, j) in $DFSC(g, t)$ can be written as a 5-tuple $(i, j, l_i, p_{(i,j)}, l_j)$, where l_i and l_j are the resource, variable or literal of nodes v_i and v_j and $p_{(i,j)}$ is the predicate of the corresponding triple pattern. All resources, literals and variables are represented as their string format in [13].

DFS Lexicographic Order. The DFS Lexicographic Order builds an linear order between two DFS codes $DFSC(g_x, t_x) = (x_1, x_2, \dots, x_n)$ and $DFSC(g_y, t_y) = (y_1, y_2, \dots, y_m)$. Assume the forward edge set and backward edge set for t_x and t_y are $E_{x,f}, E_{x,b}, E_{y,f}$ and $E_{y,b}$. In this order, $DFSC(g_x, t_x) \prec_{DFSL} DFSC(g_y, t_y)$ iff either of the followings holds true.

1. there exists t , $0 \leq t \leq \min\{m, n\}$, we have $x_k = y_k$ for $k < t$, and $x_t <_c y_t$;
2. $x_k = y_k$ for $0 \leq k \leq m$ and $n \geq m$.

The order $<_c$ above is a relationship between two elements in DFS code sequence $x = (i_x, j_x, l_{i_x}, l_{(i_x, j_x)}, l_{j_x})$ and $y = (i_y, j_y, l_{i_y}, l_{(i_y, j_y)}, l_{j_y})$. One of the following conditions holds if $x <_c y$.

1. $x \in E_{x,b}$, and $y \in E_{y,f}$.
2. $x \in E_{x,b}$, $y \in E_{y,b}$, and $j_x < j_y$.
3. $x \in E_{x,b}$, $y \in E_{y,b}$, $j_x = j_y$, and $l_{(i_x,j_x)} < l_{(i_y,j_y)}$.
4. $x \in E_{x,f}$, $y \in E_{y,f}$, and $i_y < i_x$.
5. $x \in E_{x,f}$, $y \in E_{y,f}$, $i_x = j_y$, and $l_{i_x} < l_{i_y}$.
6. $x \in E_{x,f}$, $y \in E_{y,f}$, $i_x = j_y$, $l_{i_x} = l_{i_y}$, and $l_{(i_x,j_x)} < l_{(i_y,j_y)}$.
7. $x \in E_{x,f}$, $y \in E_{y,f}$, $i_x = j_y$, $l_{i_x} = l_{i_y}$, $l_{(i_x,j_x)} < l_{(i_y,j_y)}$ and $l_{j_x} < l_{j_y}$.

The test of graph isomorphism is proven to be an NP problem [11]. If we can further restrict the patterns mined as tree-shape BGPs, the complexity can be reduced. The encoding of labeled tree into string is thoroughly investigated by many works [17][14][3] and it can be done in $O(n)$ time.

With the function mentioned above, we can employ any existing dictionary structures as our index. It can be shown that they satisfy the requirements above. We only consider an inverted-index in our implementation. The inverted index can be viewed as a function I mapping from a string to a set of sources. The result of $I(h(p)) = \mathcal{S}(g)$ represents all relevant sources for BGP p .

Ideally, the index should contain all summaries for all BGPs allowing a precise estimation of the minimum set of query relevant sources. Unfortunately, on a Web scale and given the openness it is infeasible to pre-calculate and store for all BGPs. Thus, we apply a frequent subgraph mining algorithm to detect the most frequent appearing query graph pattern. Thus, the required disk space can be kept to a minimum and the system is adaptively optimised based on the posed and expected user queries.

4 Mining Frequent BGPs from Query Logs

We employ existing work in frequent pattern mining (FPM) to find promising patterns to summarize the information of sources. Our assumption is that the summary of sources based on BGPs—that are frequently contained as subgraphs in queries—allows an efficient selection of a set of query relevant sources. We adaptively mine the frequent appearing BGPs directly from the query logs and thus tailor the source summary for the prominent user queries. We model the query log as a set $Q = \{g_1, g_2, \dots, g_n\}$ of BGPs. Further, we assume that complex queries can be decomposed into several BGPs in Q . Frequency is represented in terms of $support(g)$, the number of graphs in a set that contain g as a subgraph.

We extend the gSpan algorithm by using a different breadth-first search (BFS) strategy. The skeleton of gSpan is shown in [19]. It tries to output all frequent patterns with a minimum support min_freq from query set Q . Algorithm 1 shows the details of our extended algorithm. The underlying assumption is that a sub-pattern g' of frequent pattern g is also frequent and thus, search algorithms can be used to derive frequent patterns from already known ones. Starting from triple patterns, the algorithm iteratively extends the set of frequent patterns E by either adding a new triple pattern or substituting a variable with a resource that binds to it. The for-loop on Line 9 performs the former operation while

Line 12 performs the latter. Though our search strategy differs from gSpan, we use the same DFS lexicographic order detailed in Section 3 to perform graph isomorphism tests.

Algorithm 1. Mining Frequent Query Pattern

input : min_freq : minimum support; Q : query log in the form of a BGP set
output: The set of frequent BGPs

```

1  $E \leftarrow \emptyset$ 
2  $H \leftarrow$  empty heap
3 forall the  $q \in Q$  do
4     forall the  $tp \in q$  do
5         if  $support(\{tp\}) \geq min\_freq$  and  $g' = \{tp\} \notin E$  then
6              $E \leftarrow E \cup \{g'\}$ ; Insert  $g'$  into  $H$ 
7 while  $H$  is not empty do
8      $g \leftarrow$  pop  $H$ 
9     if  $g$  has a smaller number of nodes than  $maxsize$  then
10         forall the  $g' = g \cup \{tp_i\}$  not in  $E$  with  $support(g') \geq min\_freq$  do
11              $E \leftarrow E \cup \{g'\}$ ; Insert  $g'$  into  $H$ 
12     forall the variable  $x$  that appears in  $g$  do
13          $g' \leftarrow$  substitute a variable  $x$  in  $g$  with a possible binding
14         forall the  $support(g') \geq min\_freq$  do
15              $E \leftarrow E \cup \{g'\}$ ; Insert  $g'$  into  $H$ 
16 return  $E$ 
    
```

If we further assume the distribution of frequent patterns changes over time, this process should be done online. The entries of stale BGPs should be removed and fresh entries should be added. Performing FPM on data streams has also been investigated in [15,12]. The problem of FPM over data streams is subject to further studies but we will not cover this in this paper.

5 Selecting Sources from Summaries

In dynamic query engines, the atomic access to a data source is a reference to a URI, which yields a set of statements. The statements which bind to a triple in the queried BGP are joined to form the final result set. Let us denote $\mathcal{S}_g(g') = \bigcup_{\mu_i \models g} \mathcal{S}_{g'}(\mu_i)$. It suffices to calculate all relevant sources of all triple patterns under g 's solution, i.e. $\mathcal{S}_g(tp_i)$ for all $tp_i \in g$.

Given that summaries of relevant sources are generated and indexed, they can be used in source selection. Here we define sub-BGP as follows.

Definition 3. g_1 is called a sub-BGP of g_2 if any of the following conditions satisfies:

1. g_1 is isomorphic to a graph g' and g' is a subgraph of g_2 ;
2. g_1 can be transformed to g_2 by replacing some resources or literals with variables;
3. There exists a g' such that g_1 is a sub-BGP of g' and g' is a sub-BGP of g_2 .

We denote this sub-BGP relation as $g_1 \sqsubset g_2$.

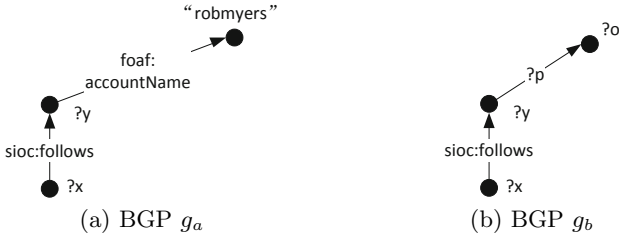


Fig. 2. An example of sub-BGP

An example of sub-BGPs is given in Fig. 2. Both BGPs g_a and g_b are sub-BGPs of our example query shown in Fig. 1, denoted as g_c . We can check g_a is isomorphic to a subgraph in g_c , which demonstrates how condition 1 in Definition 3 is applied. g_a can be transformed to g_b by replacing `foaf:accountName` with `?p` and “robmyers” with `?o` so according to condition 3, g_a is a sub-BGP of g_b . Finally, according to condition 3, $g_a \sqsubset g_b$ and $g_c \sqsubset g_a$ gives $g_c \sqsubset g_b$.

Based upon the following theorem, relevant sources of the triples in a BGP can be reduced using summaries of its sub-BGPs.

Theorem 1. *If $g' \sqsubset g$, it holds that $\mathcal{S}_g(g') \subseteq \mathcal{S}(g')$.*

Proof. We first prove that if $g' \sqsubset g$ it holds $\mu_i \models g'$ for any $\mu_i \models g$. In plain text, this means g 's solution will also be a solution to g' . For a solution μ of g , all $tp_i \in g'$ are in μ 's domain and mapped to a statement. μ is consequently also a solution of g' . So that we have

$$\begin{aligned}
 \mathcal{S}_g(g') &= \bigcup_{\mu_i \models g} \mathcal{S}_{g'}(\mu_i) = \bigcup_{\mu_i \models g} \bigcup_{tp_i \in g'} s(\mu(tp_i)) \\
 &\subseteq \bigcup_{\mu_i \models g'} \bigcup_{tp_i \in g'} s(\mu(tp_i)) = \bigcup_{\mu_i \models g'} \mathcal{S}_{g'}(\mu_i) = \mathcal{S}(g')
 \end{aligned}$$

According to Theorem 1, the solution bindings of a triple pattern in a BGP only exist in the relevant sources of its super-BGPs. When calculating relevant sources for p , all summaries of its sub BGPs can be leveraged to reduce the target set size. Following this idea, our algorithm of generating candidate sources is shown in Algorithm 2.

Algorithm 2. Approximate Relevant Sources

```

input :  $g$  :target BGP
output: The estimation of  $\mathcal{S}(g)$ 

1  $Q \leftarrow$  all triple patterns in  $g$ 
2  $E \leftarrow \{\}$ 
3 while  $Q$  is not empty do
4    $p \leftarrow$  pop head of  $Q$ 
5   forall the  $p'$  that extends  $p$  with an additional triple  $(?s, ?p, ?o)$  and  $p' \sqsubset g$ 
     do
6     if  $p' \notin E$  and  $p'$  is indexed then
7        $\lfloor$  push  $p'$  to  $Q$ 's tail;  $E \leftarrow E \cup \{p'\}$ 
8   forall the  $p'$  that substitute a variable in  $p$  with a resource, a literal do
9     if  $p' \notin E$  and  $p'$  is indexed then
10     $\lfloor$  push  $p'$  to  $Q$ 's tail;  $E \leftarrow E \cup \{p'\}$ 

11 forall the  $tp_i \in g$  do
12    $C[tp_i] \leftarrow \mathcal{S}(\{tp_i\})$ 
13   forall the  $g' \in Q$  do
14     forall the  $tp_i \in g'$  do
15        $\lfloor$   $C[tp_i] \leftarrow C[tp_i] \cap \mathcal{S}(g')$ 

16 return  $\bigcup_{tp_i} C[tp_i]$ 

```

The algorithm first finds all sub-BGPs of the specified BGP g . This process can be done by a Breadth-First-Search (BFS) starting from triple patterns of g that iteratively extends existing BGPs to its super-BGPs. All sub-BGPs that are indexed can be used in estimating g 's sources. Theorem 1 shows that statements that binds to a triple pattern in g 's solution must also exist in g 's sub-patterns. The $C[tp_i]$ in Algorithm 2 is an upper bound of sources that contribute bindings to tp_i in g 's solution. If a source in $C[tp_i]$ is found absent in $\mathcal{S}(p)$ where p is a sub-BGP of g , we can safely remove it from $C[tp_i]$. We will show this process with an example.

Take the BGP in Fig. 3 for example, which represents a source summary of Fig. 1. Our query consists of three triples, $tp_1 = ?x$ `sioc:follows` $?y$, $tp_2 = ?y$ `foaf:accountProfilePage` `identica:robmyers` and $tp_3 = ?y$ `foaf:accountName` "robmyers". If we only consider summaries for triple patterns we calculate the set of query relevant sources as follows: $\mathcal{S}(g) = \bigcup_{tp_i} \mathcal{S}(tp_i)$. The summary of $\mathcal{S}(tp_1)$ contains all sources that states the "follow" relationship. Sources that are not a part of the final solution such as `http://identi.ca/joshuagay/foaf` and `http://identi.ca/berkes/foaf` will be also considered as relevant. A typical phenome observed for source selection over triple pattern is that certain predicates (e.g. `sioc:follows`) appear in millions of data sources from which many are actually query irrelevant. But if the summary of the BGP g_1 in Fig 1 is available, only sources with statements conforming to $(?x, \text{sioc:follows}, \text{user:1108})$ are

returned as variable $?y$ binds to `user:1108` in all solutions. Hence $\mathcal{S}(g1) = \{\text{http://identi.ca/robmyers/foaf}, \text{http://identi.ca/jargon/foaf}, \text{http://identi.ca/matt1/foaf}\}$ and the final calculation will be $\mathcal{S}(g) = (\bigcup_{tp_i} \mathcal{S}(tp_i)) \cap \mathcal{S}(g)$ which is the same as $\mathcal{S}(g)$. Irrelevant sources like `http://identi.ca/joshuagay/foaf` will be abandoned, which in turn reduces the number of sources accessed.

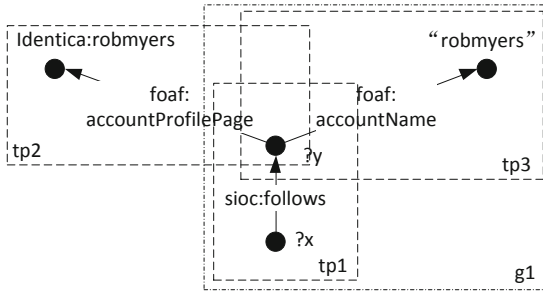


Fig. 3. An example of source summary

6 Evaluation

We design benchmarks to evaluate our approach based on a real world data set crawled from the Web and 5k automatically generated queries. We measure how different query pattern shapes and summaries improve the effectiveness of the source selection and query processing.

6.1 Experimental Setup

Our experiment is conducted on a 2.4GHz dual core personal computer running 64-bit Sun Java 6 and Ubuntu Server Linux with 4GB of memory allocated.

The dataset applied in our experiment was crawled from the Web using the LDSpider [9] framework. It consists of 10,147,442 triples with 1,547,628 resources from 11,250 RDF/XML sources. Its original size in N-Quads [2] format is 2,126 MB. This dataset represents a typical portion of data interconnected using Linked Data principles. All experiments are performed on this dump and queries are also generated based on it.

We perform our experiments on a set of 7054 synthetic BGP queries. These queries are generated using a breadth-first-search (BFS) query generator. The generated queries are further separated into four query sets $Q(3,2,1)$, $Q(4,3,1)$ and $Q(4,3,2)$ respectively with 1193, 1546 and 4315 queries, where $Q(x,y,z)$ represents a set of queries with x nodes, z of which are resources and $x - z$ variables, and y edges. Query sets $Q(4,3,1)$ and $Q(4,3,2)$ contain both path and star queries.

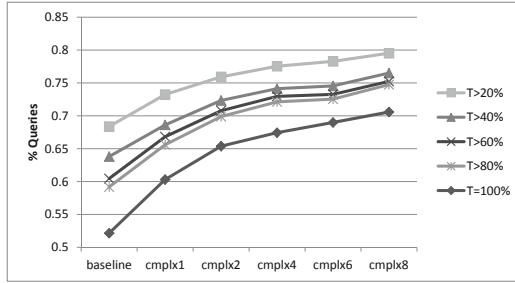


Fig. 4. Distribution of T vs. Query sample size

Initially, we build an summary index containing only a mapping of simple triple patterns and the relevant sources. As a baseline measure, we execute all our queries using the initial triple pattern index. Concretely, we measure the precision of the source selection. In the next rounds, we mine frequent appearing query patterns from 20%, 40%, 60%, and 80% of all queries and build the respective summary indices *cmplx2*, *cmplx4*, *cmplx6* and *cmplx8*. We run again all queries over these indices and measure the fraction of correct estimated query relevant sources. We show in the following subsection that these additional entries significantly improve source selection in terms of relevant source number.

6.2 Results and Analysis

We mined the most frequent sub queries with a fixed minimum frequency threshold of 0.005 for the FPM algorithms in the first experiment and used different thresholds for the second experiment. The threshold here represents the lowest ratio of $\text{support}(g)$ over $|Q|$ where Q is the sampled query set for g to be considered frequent. Thus, the first experiment shows the improvement of the source selection with increase size of the mined query patterns and the second experiment shows how different thresholds of the mining algorithm influence the correctness of the source selection. We measure in both experiments for each benchmark round the fraction of correctly selected sources as query relevant (true positives). Hence, we define $T = \frac{E}{R}$ as the true positive ration, where E is the number of relevant sources estimated using Algorithm 2 and R is cardinality of $\mathcal{S}(g)$, which is the actual number of related sources. A value of “1” denotes the desirable behaviour in which all selected sources contribute solution bindings to the given query.

The results of the first experiment are presented in detail for each query class in Fig. 5 and the aggregated values over all queries in Fig. 4. In detail, Fig. 4 depicts on the x-axis the different summary indexes and on the y-axis the percentage of queries with a T value greater than a certain threshold. The curve with label $T > 20\%$ shows the percentage of queries whose T value is greater than 20 and the one with $T > 80\%$ the percentage of queries with $T > 80\%$, etc. We show the benchmarks for each query class and over all queries, respectively.

In addition, Fig. 5 shows the average T value on the y-axis over each query sets on various query log sizes.

From Fig. 4 we see that slightly more than half of queries have the desired accurate source estimation in the baseline index and improves to up to 70% if we use 80% of all queries for our mining task. In addition, we can see by comparing the curves in Fig. 5 of different query set that queries with higher expressivity benefit more from our approach than queries with lower complexity.

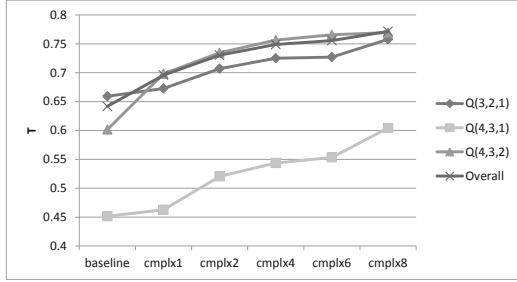


Fig. 5. \bar{T} vs. Query sample size

The figure shows T draws close to 1 as the percentage of queries sampled increases. From Fig. 5, we see only slightly more than half queries have accurate source estimation in the baseline index. This number climbs to more than 70% after 80% of the queries are used in FPM. The average T increases 33% for query set Q(4,3,1), 25% for query set Q(4,3,2) and 18% for query set Q(3,2,1). It proved our hypothesis that queries with higher expressivity benefit more from our approach.

Table 2 shows, in addition, the on disk size of our different summary indices. We can see that the additional space requirement is small compared to the improvement in the source selection of the query processing.

Table 2. Index Size for different size of query logs

	baseline	cmplx 1	cmplx 2	cmplx 4	cmplx 6	cmplx 8
Index Size (MB)	801	814	906.5	989	1040	1098

The second experiment evaluates how the threshold for the frequent subgraph mining algorithm influences the accuracy of the source selection and the space requirements of the summary index. We fix the sample rate to 50% of all generated queries and build different indices based on the patterns mined with the following thresholds: $f50 = 0.005$, $f20 = 0.002$, $f10 = 0.001$, $f5 = 0.0005$, $f2 = 0.0002$ and finally $f1 = 0.0001$. A lower thresholds relates to a lower frequency of patterns and thus, selects more candidates for the frequent pattern mining which potentially leads to more mined patterns. Whereas, a higher value normally leads to the mining of less patterns.

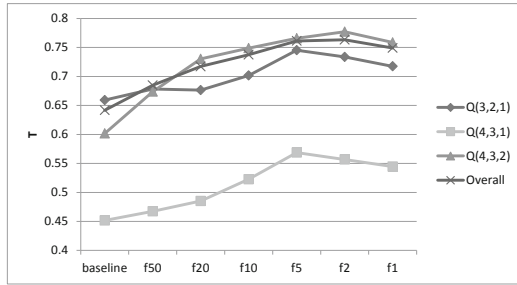


Fig. 6. T vs. FPM threshold

Table 3. Index Size for different mining thresholds

	baseline	$f50$	$f20$	$f10$	$f5$	$f2$	$f1$
Index Size (MB)	801	813	910	949	975	1027	1047

Fig. 6 shows the aggregated result of our benchmark. The x-axis again shows the different index versions and the y-axis the average success selection ratio T for the different query classes. We can observe some interesting results. A threshold of 0.0005 (cf. $f5$ in Fig. 6) shows for practically all query classes the best source selection accuracy (beside the query class $Q(4,3,2)$). We measure a decrease in the accuracy of the source selection for values either smaller or higher than this particular threshold. Higher threshold is biased towards complex patterns which contributes to a less number of queries though being more accurate for a single query.

Table 3 shows the required space consumption for the summary index for different mining thresholds. Clearly, the size grows as the threshold value decreases and more patterns are indexed. Compared with the T values in Fig. 6, we show that larger index size and more indexed patterns does not necessarily imply better results for source selection, which emphasizes the importance of a sophisticated strategy to mine those patterns that can better help the source estimation.

7 Related Work

Traditional query execution over Linked Data considers the data sources in the Web of Data fixed and known beforehand. O. Hartig et al [7,6] proposed a new query execution paradigm which assumes data sources are dynamic and unknown. In this paradigm, queries are locally processed in contrast to the traditional paradigm where queries are decomposed and dispatched.

O. Hartig [8] also discussed the application of hash tables to index relevant information in main memory. It is served to support link traversals in their new

query execution paradigm. Our work use a similar data structure as the index. But we extend the capability of the index to support complex BGPs which can further reduce the number of relevant sources.

Following this line of thought, A. Harth et al. [5] proposed to use Q-tree as the index structure to store and index approximate descriptions about data sources. The problem of source selection are reduced to spatial queries over the Q-tree. Their approach aims at source selection for triple patterns and joined patterns. There exist two disadvantages. Firstly, the performance of this approach is sensitive to sparsity of data points which is highly correlated to the design of the hash function. However, the authors give no details about the hash function used. Secondly, selectivity of joined patterns with two or more steps are not considered to reduce the number of sources. We will come to this point later.

G. Ladwig [10] compared the two approaches above and categorized the former as *Bottom-Up Query Evaluation* and the latter as *Top-Down Query Evaluation*. It is reported in empirical results that the top-down evaluation demonstrated better performance over the bottom-up approach. The performance gain nevertheless needs prior knowledge of relevant sources. Our approach focuses on the top-down approach.

8 Conclusion and Future Work

We presented an approach of selecting relevant sources for query evaluation over Linked Data leveraging query log to enhance query performance. Frequent BGPs are mined from query log to guide building source summaries. We showed by empirical result our approach improves source selection by reducing the number of access to data sources close to the minimum value. We also analyzed the effect of FPM threshold over the size of estimated sources. Our experiment results also reveal that complex queries with higher expressivity benefit more significantly than simple queries.

There also lefts large space of extending our approach and analyzing related aspects. Ranking of relevant sources can be considered to reduce the time retrieving top-k results. Management of updates to index entries should be analyzed to embrace changes of data sources. Mining query streams dynamically further assumes possible changes in query patterns. These related topics are valuable future work.

References

1. Berners-Lee, T.: Linked Data - Design Issues, <http://www.w3.org/DesignIssues/LinkedData.html>
2. Cyganiak, R., Harth, A., Hogan, A.: N-Quads: Extending N-Triples with Context (2009), <http://sw.deri.org/2008/07/n-quads/>
3. Deo, N., Micikevicius, P.: A new encoding for labeled trees employing a stack and a queue. Bulletin of the Institute of Combinatorics and its (2002)

4. Haase, P., Mathaß, T., Ziller, M.: An evaluation of approaches to federated query processing over linked data. In: Proceedings of the 6th International Conference on Semantic Systems, pp. 1–9. ACM (2010)
5. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K., Umbrich, J.: Data summaries for on-demand queries over linked data. In: Proceedings of the 19th International Conference on World Wide Web, pp. 411–420. ACM, New York (2010)
6. Hartig, O.: Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 154–169. Springer, Heidelberg (2011)
7. Hartig, O., Bizer, C., Freytag, J.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
8. Hartig, O., Huber, F.: A main memory index structure to query linked data. In: Proc. of the 4th Int. Linked Data on the Web (2011)
9. Isele, R., Umbrich, J., Bizer, C.: Ldspider: An open-source crawling framework for the web of linked data. In: International Semantic Web Conference 2010, pp. 6–9 (2010)
10. Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
11. Lubiw, A.: Some NP-complete problems similar to graph isomorphism. *SIAM Journal on Computing* (1981)
12. Manku, G., Motwani, R.: Approximate frequency counts over data streams. In: Conference on Very Large Data Bases (2002)
13. Manola, F., Miller, E.: RDF Primer, <http://www.w3.org/TR/rdf-syntax/>
14. Neville, E.: The codifying of tree-structure. *Proceedings of Cambridge Philosophical*, 381–385 (November 1953)
15. Ng, W., Dash, M.: Discovery of Frequent Patterns in Transactional Data Streams. *Transaction on Large-Scale Data-and Knowledge-Centered Systems*, 1–30 (2010)
16. Prud’hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>
17. Prüfer, H.: Neuer beweis eines satzes über permutationen. *Archiv für Mathematik und Physik* (1918)
18. Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., Decker, S.: Towards dataset dynamics: Change frequency of linked open data sources. In: 3rd International Workshop on Linked Data on the Web (LDOW 2010), in Conjunction with 19th International World Wide Web Conference, CEUR (2010)
19. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. *Order a Journal on the Theory of Ordered Sets and its Applications* (2002)

Semantic Caching for Semantic Web Applications^{*}

Mengdong Yang¹ and Gang Wu²

¹ Southeast University
mdyang@seu.edu.cn

² Northeastern University
wugang@ise.neu.edu.cn

Abstract. Data caching is one of the directions for improving the performance of data management systems with reasonable space overhead based on the spatial/temporal locality principle. The intrinsic rich semantics in the Semantic Web applications makes it possible to design meaningful semantic level caching schemes beyond low level individual triples caching and specific application objects caching. In this paper, we propose a caching scheme that captures the semantics originated from both SPARQL query and RDF data. The scheme applies two caching approaches: the SPARQL algebra expression tree based caching and the entity caching. It can efficiently improve the performance of query evaluations that have massive join operations and identical sub-queries between successive queries. Evaluation results on three mainstream RDF benchmarks and one comparison system are provided to illustrate the effectiveness and efficiency of our approach.

Keywords: RDF, AET based Caching, Entity Caching.

1 Introduction

The Semantic Web has become a *Data Web* by specifying explicit semantics to information so as to make the data understandable by both human and software agents. Resource Description Framework (RDF) is the model that defines the syntax for data interchange in the Semantic Web. It has a more flexible expressiveness than the relational data model while increasing the complexity of data processing, especially in large scale. There already has been billions (B) of RDF triples on the Semantic Web, and many of them are frequently accessed (e.g. UniProt¹ has approximately 1.5B triples). Therefore, efficiently processing large scale RDF data becomes an urgent task in the field of Semantic Web data management.

^{*} Gang Wu is partially supported by the National Natural Science Foundation of China under grants 60903010, 61025007 and 60933001, the National Basic Research Program of China under grant 2011CB302206, the Natural Science Foundation of Jiangsu Province under grant BK2009268, and the Key Laboratory of Advanced Information Science and Network Technology of Beijing under grant XDXX1011.

¹ <http://www.uniprot.org/>

RDF data can be seen as entity-relation (E-R) graphs with directed labeled edges corresponding to $\langle \text{subject}, \text{predicate}, \text{object} \rangle$ triples (a.k.a. $\langle \text{subject}, \text{property}, \text{value} \rangle$ triples). For example, the fact that *product1* with label “*label1*” is produced by *producer1* could be expressed as two triples $\langle \text{product1}, \text{label}, \text{“label1”} \rangle$ and $\langle \text{product}, \text{producer}, \text{producer1} \rangle$, where *product1* and *producer1* are URIs that identify unique entities, and “*label1*” is the literal representing the label. If someone wants to query products and their corresponding labels and producers, she/he can write a query by specifying a graph pattern like following with SPARQL query language:

Query 1. Select the products, their labels and producers.

```
SELECT ?product ?p ?l
WHERE{
    ?product      producer      ?p .
    ?product      label         ?l }
```

Here, a graph pattern consists of one or more triple patterns, which are delimited by period delimiter “.”. Identifiers starting with question mark “?” are variables, the values of which are to be determined during query evaluation. A join on $?product=?product$ is required in order to find out triples with variable “*?product*” appearing as subject in both triple patterns $\langle ?product, \text{producer}, ?producer \rangle$ and $\langle ?product, \text{label}, ?label \rangle$. In practice, a SPARQL query may contain many more triple patterns, hence a considerable number of joins needs to be issued during the evaluation, which makes it a hard work. In order to speed up query processing, typical DB techniques like index structures, query optimizations, and parallelization are well studied and utilized in mainstream RDF data management systems (e.g. Sesame [1], Jena [2, 3], Virtuoso [4], RDF-3X [5], YARS2 [6], Oracle’s RDF_MATCH [7]).

As we know, data caching scheme is another direction for improving the performance of data management systems with reasonable space expense based on the spatial/temporal locality principle. It has been widely developed in the field of DB and DB-based Web 2.0 applications. There are three kinds of caching granularity [15], i.e. page level, tuple level, and semantic caching. Since data are usually organized to pages on the disk, designing a page level caching is intuitive but may be insensitive to the physical distribution of the data. Although the finest granularity level, tuples caching, can avoid bad data clustering problem, it will produce more join operations. Semantic caching is believed to have better performance than the others [15, 16], which embeds semantic information about the queries and data.

Related Work

Few of the above RDF data management systems concentrate on RDF data caching design. As for Semantic Web applications, there are only a few researches on the very similar topic. In [10], an approach is proposed to materialize path expressions like $\langle \text{book}, \text{author}, \text{someone} \rangle, \langle \text{someone}, \text{wasBorn}, \text{‘1860’} \rangle$ to a form like *book.author:wasBorn=‘1860’*. Oracle’s RDF_MATCH system[7] implements generic

materialized join views to avoid self-join of triple table. RDF_MATCH also applies subject-property matrix materialized join views to minimize the query processing overheads that are inherent in the canonical triple based RDF representation. Roger Castillo implemented RDFMatView [13] that materialized frequent join patterns based on analysis of SPARQL queries. However, materialized join view isn't a caching scheme after all because it doesn't have a replacement principle.

To the best of our knowledge, the most relevant work in the Semantic Web is [9] where Michael Martin et al. introduced a proxy cache layer between web application and RDF repository. Both triple level query result caching and application object caching are employed. The caching scheme gets a good result on Berlin SPARQL Benchmark (BSBM). Considering the intrinsic rich semantics in the Semantic Web applications, it is possible to design more meaningful semantic level caching schemes beyond low level individual triples caching and specific application objects caching.

Contributions

In this work, we propose a caching scheme that effectively improves the query evaluation performance of complex SPARQL queries over very large RDF graphs. In order to better serve SPARQL query evaluations, the caching scheme is designed to have a semantic level granularity which can capture the semantics originated from both SPARQL query and RDF data. The contributions of this paper include:

- i. We developed a robust and efficient approach that caches intermediate query results based on the SPARQL Algebra Expression Tree. Query identification and intermediate result serialization methods are described to explain how such kind of query caching speeds up the evaluation of SPARQL queries with similar sub-queries.
- ii. We also propose an entity caching approach, which aggregates triples of the same entity according to the ontology. This approach reduces star-shaped joins during query evaluation phase and brings significant performance improvement.
- iii. We implement these caching schemes on Sesame RDF framework. Benchmark evaluation results are provided to illustrate the effectiveness and efficiency. Comparisons with prior related work are made as well.

The rest of this paper is organized as follows. In section 2, we present necessary concept definitions. Section 3 delivers the mechanism of our caching schemes (both SPARQL Algebra Expression Tree based caching and entity caching). In section 4, details of the system implementation are given. In section 5, we present the evaluation result of query performance on LUBM, SP²Bench and BSBM. Analysis is given as well to illustrate the feasibility and effectiveness of our system. Finally, we conclude this paper and give some possibilities of the future work.

2 Preliminaries

2.1 Notions and Definitions

RDF is a data model proposed for the Semantic Web where data are graph-structured. An RDF graph is a directed graph with labeled nodes and edges, in which the basic unit is RDF triple. Definition 1 gives the formal definition.

Definition 1. Assume U is the set of URIs, B the set of blank nodes and L the set of literals. Triple $t = (s, p, o) \in (U \cup B) \times (U) \times (U \cup B \cup L)$ is called an **RDF triple**.

Here, s (subject) represents an entity, while p (predicate) specifies a property of the entity and o (object) the property value. All subjects and objects constitute the vertices of an RDF graph and predicates the edges.

SPARQL is a pattern matching-based RDF query language recommended by W3C. Definitions of triple pattern and basic graph pattern are provided as follows.

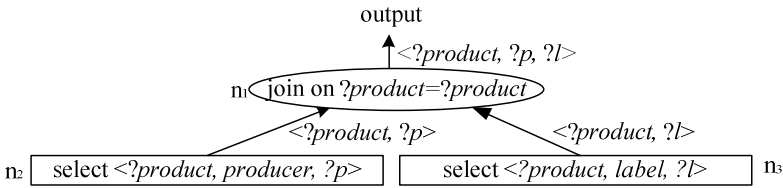


Fig. 1. AET of Query 1

Definition 2. Assume V is an infinite set of variables disjoint from $U, B,$ and L . Then a triple $tp = (s, p, o) \in (U \cup B \cup V) \times (U \cup V) \times (U \cup B \cup L \cup V)$ is a **triple pattern**.

Definition 3. Assume p_1, p_2, \dots, p_n ($n \geq 1$) are all triple patterns, then set $G = \{p_1, p_2, \dots, p_n\}$ is a **Basic Graph Pattern**.

In a basic graph pattern (BGP) SPARQL query, one or more triple patterns are given to specify the graph pattern. During the evaluation phase, a query engine searches the RDF graph for sub graphs that match the graph pattern and returns them as query results. Besides triple selection, **BGP** also issues Join and Projection.

Definition 4. Assume p_a and p_b are two triple patterns in one graph pattern, then a **Join** $j(p_a, p_b, v) = p_a \bowtie_{v=v} p_b$ occurs if and only if a variable v appears in both p_a and p_b (as any of s, p and o).

Definition 5. Given a set of binding names N , then the **Projection** of a binding B with binding names N on binding names N_p is $p(B, N_p) = \{v | v \in N \cap N_p\}$.

Definition 6. Given a SPARQL query Q , then an expression $E = (P, O)$ is the **Algebra Expression** (AE) for evaluating Q , where P is the set of triple patterns and O the set of operations to be executed during query evaluation. In a basic graph pattern SPARQL query Q_b , the corresponding AE is $E_b = (P, J \cup Pr)$, where J is the set of join operations and Pr the set of projection operations.

For instance, Query 1 has the corresponding AE:

$$p(j(<? corp, locatedIn, mtview >, <? corp, hasName, ? name >), \{? corp, ? name\})$$

Obviously, AE can be organized in a tree shaped structure. Hence, we also call such a tree corresponding to the AE an *Algebra Expression Tree* (AET). The AET of Query 1 is shown in Fig. 1.

Leaf nodes (identified by rectangle) in an AET are always triple selection operation nodes, where a triple selection is performed with the specified triple pattern. Non-leaf nodes (identified by ellipse) are other operations including join. As projection is always performed automatically after a node evaluation finish, projections are omitted in AET. An AET can represent the evaluation process of a SPARQL query. In this study, we consider only AETs of **basic graph pattern SPARQL queries**. So AETs in this paper contain only joins, projections, and triple selections.

2.2 SPARQL Query Evaluation

The evaluation of a SPARQL query is similar to that of a SQL selection-projection-join query. To evaluate Query 1, an AET shown in Fig. 1 may be generated at query evaluation phase by query engines e.g. Sesame.

The join order may vary due to the optimization strategy selected and statistics information provided. For complex basic graph pattern queries, there could be much more triple patterns specified, which brings more join operations and makes optimal join ordering a challenge. The tree shown in Fig. 1 is only an example that helps us describe our caching scheme. Those complex query optimization techniques beyond the scope of basic graph pattern will not be discussed in this paper.

Once a SPARQL query is translated into an AET, the query engine will perform a post-order traverse in the tree to ensure that all the sub-AET trees of a node have been evaluated before evaluating the node itself. Apparently, the output from the root node is the evaluation result of the query. Such an evaluation process is usually implemented as a recursion expansion which will not stop until the engine reaches leaf nodes. The recursion expansion is a top-down process, while recursion reduction is a bottom-up one.

Since SPARQL queries can be very complex, the recursive evaluation of AET may be very deep and involving a lot of join nodes in its AET. In real-world Semantic Web applications, such queries are quite common. As join operations usually have heavy computation loads and consume a large amount of memory, they are the temporal/spatial bottleneck of the performance of RDF data management systems, especially for those dealing with large-scale RDF data.

3 Principles and Mechanism

The advantage of efficiency and flexibility of semantic caching schemes relies on the semantic description of data and queries maintained [15]. The intrinsic expressiveness of RDF data and SPARQL query language makes it possible to express rich semantics in Semantic Web applications, and hence to develop reasonable semantic caching

schemes. In this work, our scheme maintains the semantics from two aspects: caching with the SPARQL algebra expression tree; caching with ontology information.

3.1 AET Based Caching

For Semantic Web data management systems that expose SPARQL endpoints, they are more likely to face the scenario described as in Berlin SPARQL Benchmark (BSBM), where a sequence of queries mixed to form a typical query use case. Therefore, in one query mix, different queries share the partial/whole semantics. By maintaining the related result of previous queries, a semantic cache has the ability to speed up the query evaluation of semantically related queries. In this section, we present an AET query plan based caching scheme that caches intermediate join results for possible reuse in future SPARQL queries.

To illustrate our approach, we provide a scenario that consists of two successively issued SPARQL queries that both derived from Query 1.

Query 2.

```
SELECT ?product ?l ?p ?c
WHERE { ?product label ?l.
        ?product producer ?p.
        ?product comment ?c }
```

Query 3.

```
SELECT ?product ?l ?p ?prop
WHERE { ?product label ?l.
        ?product producer ?p.
        ?product productProperty ?prop }
```

Obviously, there are redundant evaluations of triple queries $\langle ?product, producer, ?p \rangle$ and $\langle ?product, label, ?l \rangle$, and the join $\langle ?product, producer, ?p \rangle \bowtie_{?product=?product} \langle ?product, label, ?l \rangle$. It can be easily discovered if we translate

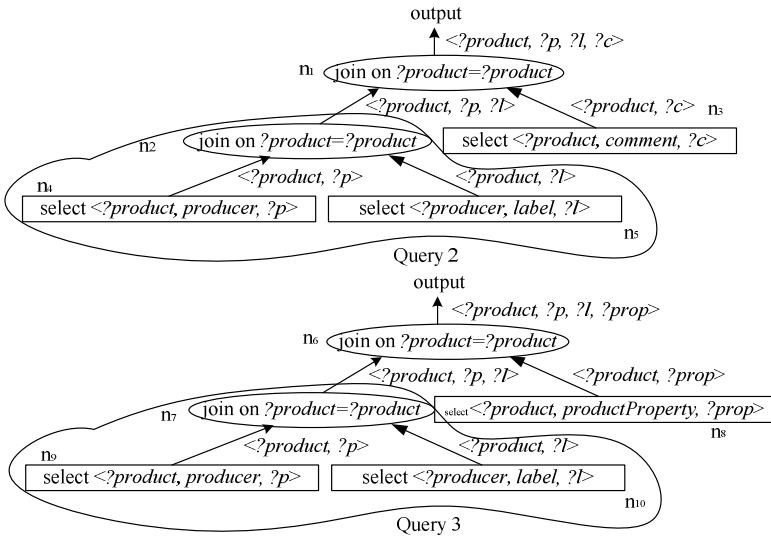


Fig. 2. AETs of Query 2 and Query 3

the queries into AETs. Assume that the two queries have the corresponding AETs shown in Fig. 2. Apparently, they have a common sub tree as annotated by the line. The evaluation result of the sub AET is an intermediate query result. If the sub AET can be detected and its result cached, it will avoid the redundancy.

As can be deduced from the example, more complex queries will lead to more complex AETs and more sub AET evaluation results to be cached.

The cached sub AET results can be reused if there are identical sub AETs to be evaluated again. Once the cache is hit, the result of matched sub AET can be directly accessed, and hence repeated issuing of join can be avoided to saves time. In fact, queries having common sub AET in their corresponding AETs are very common in Semantic Web applications. Intuitively, caching the result of common sub AETs can speed up the evaluation process, thus reduce the temporal and spatial expense of the application.

Identifying AETs

As we know, there may be several logically equivalent algebraic expressions for a given query. On one hand, it makes providing query optimization feasible in a query engine; on the other hand, it makes our caching scheme confront the challenge of identifying an AET so that the query engine can recognize the identical AET and reuse the cached result corresponding to it. To solve this problem, we need first define the equality of two AETs:

Definition 7. Assume R_A and R_B are the root nodes of two AETs T_A and T_B . Then we have:

$$\begin{aligned}
 T_A = T_B \Leftrightarrow & \\
 & (T_A.subj = T_B.subj) \wedge (T_A.pred = T_B.pred) \wedge (T_A.obj = T_B.obj) \\
 & \quad \mathbf{when} (T_A \text{ is a leaf node}) \wedge (T_B \text{ is a leaf node}) \\
 & (T_A.leftChild = T_B.leftChild) \wedge (T_A.rightChild = T_B.rightChild) \\
 & \quad \mathbf{when} (T_A \text{ is a join node}) \wedge (T_B \text{ is a join node}) \\
 & \mathit{false} \quad \mathbf{otherwise}
 \end{aligned}$$

As the equality relation has a recursive definition, we thus obtain a recursive rule that helps to generate an identifier for each unique AET. The rule can be defined as shown below:

Definition 8. Given the root node R of an AET T , then we have

$$\begin{aligned}
 ID(T) = & \quad \text{"<" } T.subj \text{ "," } T.pred \text{ "," } T.obj \text{ ">"} \\
 & \quad \mathbf{when} T \text{ is a leaf node} \\
 & \quad \text{"J(" } ID(T.leftChild) \text{ "," } ID(R.rightChild) \text{ ")"} \\
 & \quad \mathbf{when} T \text{ is a join node}
 \end{aligned}$$

There is still one case left to be considered. Suppose that we use $?c$ instead of $?corp$ for corporations in Query 2. The new query is lexically different from Query 2,

though the two queries have the same logic, and hence identical AETs. To solve the problem, a variable naming normalization is performed on the AET. The generated *ID* from the normalized AET can be used to identify AETs with the same query semantics. The normalization is implemented as a variable labeling method during the pre-order traverse of the AET. Each time the ID generator meets a triple pattern p , it assigns an ID id to each variable v in p . It first looks up the name n of v in a global map m which contains $\langle name, ID \rangle$ pairs. If a pair $\langle n, id \rangle$ exists in m , the generator assigns the found id to v . Otherwise the generator generates a new ID id , and add pair $\langle n, id \rangle$ to m after assigning id to v .

The output of the normalization phase is a new AET with expression $\langle ?v1, locatedIn, mtview \rangle \bowtie_{?v1=?v1} \langle ?v1, hasName, ?v2 \rangle$. Therefore the ID of such a normalized AET is $J(\langle ?v1, locatedIn, mtview \rangle, \langle ?v1, hasName, ?v2 \rangle)$ according to rule defined in Definition 8. The generated ID can then be used to identify an AET with unique query semantics.

Replacement

AET-based caching employs a conventional version of the LRU replacement algorithm. When a cache item addition to the cache repository is required when it is full, the least frequently used cache item is deleted from cache repository to yield storage space for the new item.

Cache Update

AET-based caching caches output result of sub-AETs, which contains the execution result of the join operations in the sub-AETs, so it is complex to compute an updated version of a cache item when it is affected by triple addition/removal operations to the triple repository. Besides, AET-based caching takes effect in a relatively small query context (a query is cached, and the next coming query's AET has an identical sub-AET to the previous one). So considering cache update in AET-based caching is costly and not worthwhile. In our AET-based caching scheme, all cache items immediately become invalid on triple repository update (triple addition/removal to triple repository).

SPARQL Query Evaluation with AET Caching

In order to utilize the caching scheme to improve the performance of SPARQL query evaluation, it needs some modifications to the traditional query evaluation algorithm. A cache lookup operation is inserted before evaluating a sub AET; a cache writing operation is performed before returning the result of a sub AET to its parent node in case of cache miss. With our caching scheme, a SPARQL query engine works as shown in Algorithm 1.

Algorithm 1. AET Evaluation Algorithm with Caching

Input: A normalized AET T_n **Output:** Evaluation result R of input**Global Variable:** $current_node$ (the node whose result is being evaluated by the query engine)**Initialization:** $current_node \leftarrow$ root node N_r of T_n **Function** $evaluate(T)$ **Returns** R 1 $R_T \leftarrow$ the root node of T_n 2 **If** R_T is leaf3 Issue triple selection R_T , result as R_t 4 **Return** R_t 5 **Else**6 Compute ID of T as id 7 **If** cache item $\langle id, content \rangle$ exists in cache repository8 Access cache item, $content$ as R_c 9 **Return** R_c 10 **Else**11 $S_c \leftarrow \{N_1, N_2, \dots, N_n\}$, where N_1, N_2, \dots, N_n are direct child nodes of T 12 $S_{cr} \leftarrow \{evaluate(N_1), evaluate(N_2), \dots, evaluate(N_n)\} = \{R_1, R_2, \dots, R_n\}$ 13 Execute operation on $current_node$ with S_{cr} , result as R_c 14 **If** cache repository is full

15 delete an item from cache repository with LRU

16 Add cache item $\langle id, R_c \rangle$ to cache repository17 **Return** R_c **End Function****Algorithm:** $R = evaluate(T_n)$

The algorithm is recursive. The recursion expands at line 12 where the evaluation of a node depends on the results of all its children. Before the recursion, we add the cache lookup (line 7-9) operation: the query engine first computes the ID of the current sub AET that takes $current_node$ as its root node; then it performs a lookup in the cache repository with the ID. The evaluation result will be directly accessed and returned to the parent node if there is a matched cache item, thus the evaluation from $current_node$ down is avoided. Otherwise (line 10-17) the engine evaluates the result of the current sub AET, and returns the result to the parent node after storing it in the cache repository.

3.2 Entity Caching

AET based caching improves the performance of evaluating SPARQL queries with identical sub AET structure. However, for queries with same semantics but different join ordering, AET based query caching won't work well since the corresponding

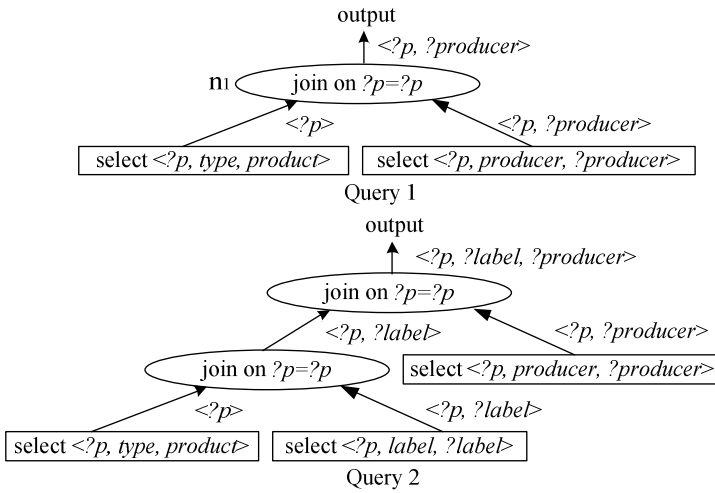


Fig. 3. Queries with Different Join Orders

AETs have different structures. Take AETs shown in Fig. 3 as an example, caching the output of n1 will be not helpful for the evaluation of Query 2 because Query 1 and Query 2 have different join orders due to the query optimization strategies chosen.

Moreover, selecting n properties of an entity means requiring $n-1$ joins in the corresponding AET. In relational database, a join operation only occurs when there is an inter-relational condition restriction. While no join operation is required when selecting properties within a relation. This inspired us to materialize those triples related to the same entity as a view to avoid join operations when retrieving the information of the entity. Fortunately, RDF data model simplify the materialization. We can extract and aggregate triples having the same subject into a tuple containing all properties of the specific entity. A table storing such tuples is shown in Table 1.

Table 1. Sparse Table Storing Aggregated RDF Triples

ID	type	feature	label	producer	description
product1	{product}	{feature1, feature2}	{"p1"}	{producer1}	-
feature1	{productFeature}	-	{"pf1"}	-	{"description1"}
feature2	{productFeature}	-	{"pf2"}	-	{"description2"}
Producer1	{producer}	-	{"pp1"}	-	-

We use a horizontal table [17] to store tuples based on the following two reasons.

1) Flexible ontology representations are allowed with this kind of table schema. First, for an entity, the value of any possible property can be null. Second, an entity can have multiple types and hierarchical structured types.

2) An intra-entity selection can be completed via one single access to the corresponding row in the table avoiding star-shaped joins.

Cache Construction

This subsection explains how cache items are constructed and filled into the cache repository on the cache miss. The cache repository is initially to be empty. On evaluating a query, triple patterns specified in the basic graph pattern are grouped according to subjects. For example, Query 2 in BSBM specifies a set of basic graph patterns and groups as follows (see Table 2).

Table 2. Grouped Triple Patterns and Graph Representation

ID	Triple Patterns
product1	product1 rdfs:label ?label . product1 rdfs:comment ?comment . product1 bsbm:producer ?p . product1 bsbm:productFeature ?f . product1 bsbm:productPropertyTextual1 ?propertyTextual1 . product1 bsbm:productPropertyTextual2 ?propertyTextual2 . product1 bsbm:productPropertyTextual3 ?propertyTextual3 . product1 bsbm:productPropertyNumeric1 ?propertyNumeric1 . product1 bsbm:productPropertyNumeric2 ?propertyNumeric2 .
?p	?p rdfs:label ?producer .
?f	?f rdfs:label ?productFeature .

Each group contains the triple patterns selecting multiple properties of an entity. Relations are also specified between corresponding groups. After grouping, joins are categorized into two types: intra-entity (group) and inter-entity (group). Consequently, a query evaluation is divided into two phases. 1) Entity selection: each group are issued independently; 2) inter-entity joins are performed to generate the final result of the basic graph pattern. After phase 1), each group will have multiple entities selected corresponding to the triple pattern specified in the group. Each entity corresponds to a tuple that includes the entity’s URI and all its properties. The tuples are then added to the table for possible future reuse.

Query Expansion and Triple Reducing

Entity caching effectively solves the problem of selecting multiple properties of an entity, but brings some other affections at the same time. An entity may be cached partially rather than totally. For instance, a product has the following properties: *type*, *label*, *comment*, *producer*, *productFeature*, and *productProperty*. Suppose query 1 select *type*, *label*, *comment* and *producer* of product1, then there will be only one tuple <product1, type:product, label:”product1”, comment:”comment1”, producer:producer1> in the cache after query 1 is answered. Values of *productFeature* and *productProperty* are both null in the cached tuple. At this time, if there is a query 2 selecting *producer*, *productFeature*, *productProperty* of product1, it will still require accesses to both the entity cache and the triple store. This will apparently affect the evaluation performance.

To solve the problem, we have to expand a triple pattern group to include all property values of an entity in the cache once the object is accessed. In this way, a single access to the entity cache item can complete entity selection whatever properties the selection includes. Since the entire entity description exists in the entity cache, there is no need to store those triples in triple store as long as the cache holds the entity. With the query expansion, all triples describing the entity are removed from the triple store to reduce the store size. And a triple write-back is performed when the entity is obsolete from the cache repository. Detailed experiment results are provided in Section 4 to show the performance improvement.

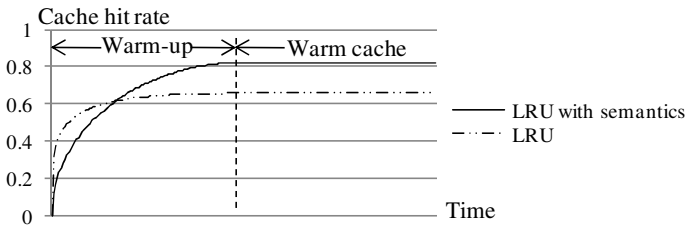


Fig. 4. Comparison of LRU with semantics and LRU

Cache Update

The latest SPARQL draft supports update operations on RDF graphs. Before the draft, updates to the RDF graph are converted atomic triple operations: triple addition, triple removal and triple lookup. Cache update is what to be concerned when triple addition/removal occurs. Our update strategy to the entity caching is quite simple. Suppose the current triple to be added/removed is $\langle s, p, o \rangle$, then when 1) entity s is in the triple store, the addition/removal operation of $\langle s, p, o \rangle$ is performed on the triple store; 2) entity s is in the entity cache, then the addition/removal operation of property $p:o$ is performed on the cache repository.

Replacement Strategy

The LRU algorithm and its variations have been proved to be effective and efficient in the cache replacement. In this paper, we enhance LRU with semantics. In our replacement approach, if a cache item is hit, then the cache item itself and all its referring items are moved to the head of the LRU list. For example, if item (product1, hasFeature:{feature1,feature2}, producer:{producer1}) is hit, then item (feature1, ...), (feature2, ...), (producer1, ...) should all be moved to the head of the LRU list. The principle behind this strategy is the locality of data access. In practical applications, adjacent operations usually have some relation in semantics (e.g. search for products that fulfill specified restrictions, and then view product details of some of them). So the data semantically related to the currently accessed data are more likely to be accessed in the near future. Compared to plain LRU, LRU with semantics has higher cache hit rate during warm cache phase (See Fig. 4).

4 Implementation

We implement our caching schemes on Sesame [1], an Java RDF framework.

The system architecture is shown in Fig. 5. Both of AET based caching and entity caching employ a two-level storage. The main memory stores the most frequently access cache items. Swapped out items are stored on the disk for future possible use.

After parsing, SPARQL queries are translated into AETs and sent to AET cache manager, a lookup in the AET cache repository is performed. On cache hit, the result is directly returned to its requester. Otherwise the AET is passed to entity cache manager, where AET are processed and an entity cache lookup is performed in the entity cache repository. On cache hit, the results are directly sent to its requester, and AET cache repository is updated. Otherwise the AET is passed to AET evaluator and is evaluated in a conventional way. Results are returned to all requesters with both AET cache repository and entity cache repository updated.

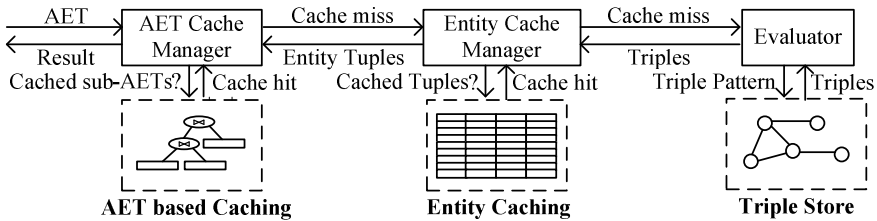


Fig. 5. Architecture of the System

AET based Caching stores cache items in a custom designed binary format. It uses a deflation compression algorithm to reduce the space consumption.

Entity Caching stores cached tuples as a key-value object in the main memory. For tuple persistence, MySQL database is used. Tuples are serialized in JSON format and indexed with Lucene.

5 Evaluation

We employ LUBM [11], SP²Bench [12] and BSBM [13] as benchmarks for evaluating our caching scheme. Before presenting the evaluation result, we first introduce metrics used.

5.1 Evaluation Metrics

Temporal Improvement. Three values are compared in this metric: 1) time consumption of query evaluation with conventional means without our caching scheme; 2) time consumption of query evaluation with our cache scheme at cache warm-up phase, when few cache hits happen; 3) time consumption of query

evaluation with our cache scheme at warm cache phase, when cache hits happen relatively more frequently and the time consumption tends to be stable.

Cache Hit. Cache Hit evaluates that to what extent the cached items is utilized.

Eliminated Join Operations. This metric evaluates how many joins are eliminated.

5.2 Experiment Setup

Datasets

LUBM: We test our caching scheme on three LUBM datasets: LUBM(10) (1.3M triples), LUBM(100) (13.8M triples) and LUBM(1000) (138M triples).

SP²Bench: We use SP²Bench datasets of 3 different scales: 2.5M triples, 10M triples and 40M triples.

BSBM: We use 3 different BSBM datasets: BSBM(1,000) (0.35M triples), BSBM(10,000) (3.5M triples) and BSBM(100,000) (35M triples).

Evaluation Process

For i from 1 to 10

Construct condition C

Issue query mix Q , which costs time period T_i

Next i

Output average value T of T_1, T_2, \dots, T_{10}

Where:

1) In the evaluation of no cache query evaluation performance,

$C = C_n$ = Disable our caching scheme and use original Sesame implementation;

In the evaluation of query evaluation performance at cache warm-up phase,

$C = C_u$ = Enable our caching scheme and clear up memory/disk cache repository;

In the evaluation of query evaluation performance at warm cache phase,

$C = C_w$ = We have already finished evaluation under C_u . Now go on without clearing up memory/disk cache repository.

2) In the evaluation on LUBM, $Q = Q_L = 14$ queries are provided by LUBM;

In the evaluation on SP²Bench, $Q = Q_S = 17$ queries are provided by SP²Bench;

In the evaluation on BSBM, $Q = Q_B = 25$ queries are provided by BSBM.

Runtime Resource

The evaluation was performed on a HP dx2390 workstation with Q8400 CPU (2.66GHz, quadruple core), 4GB RAM and 320GB hard disk drive, running Windows Server 2008 R2 Enterprise Edition. JVM version is JDK 1.6.0 Update 12 for x64 architecture. Java programs are run with $-Xmx1024M$ command argument.

5.3 Evaluation Result

Temporal Improvement

Experimental results of temporal improvement on LUBM and SP²Bench datasets are shown in Table 3. Query evaluation at cache warm-up time costs more time than running without our caching scheme. This can be explained as follows: 1) there are few cache hits during cache warm-up time, so most queries are evaluated in a conventional way like how they are evaluated in the original Sesame implementation.

2) Besides query evaluation time, there is still another expense: when the processing of a request is completed, a cache item will be added into the cache repository. This latter part of time consumption makes query evaluation at cache warm-up time a bit slower than that is in the original Sesame implementation without caching scheme.

Obviously AET based Caching contributes most to performance improvement on LUBM and SP²Bench. The reason is that LUBM and SP²Bench provide static query mixes, and hence they have the same queries included respectively.

But for BSBM, AET based Caching contributes little, while Entity Caching contributes most to performance improvement. That is because BSBM generates queries dynamically, which makes queries have little in common, and hence AET based Caching has few cache hits. While entity caching caches entities with similar semantics and indexes them on appropriate properties. This helps a lot in reducing the workload of BSBM query evaluation.

Table 3. Benchmark Test Result (Time in second)

Dataset {1,2,3}={LUBM(1000), D2=SP2Bench(40M), D3=BSBM(100k)}
TpQm=Time per Query Mix, **CHpQm**=Cache Hits per Query mix
AC=AET based Caching, **EC**=Entity Caching

			Dataset 1	Dataset 2	Dataset 3
No Cache		TpQm(ms)	36635.12	485.49	108.40
AC	Warm-up	TpQm(ms)	40828.61	509.31	124.52
		CHpQm	2	4	5
	Warm	TpQm(ms)	712.56	153.61	106.35
		CHpQm	14	10	7
AC+EC	Warm-up	TpQm(ms)	45135.86	523.71	132.65
		CHpQm	5	6	17
	Warm	TpQm(ms)	679.51	119.86	18.77
		CHpQm	14	10	43

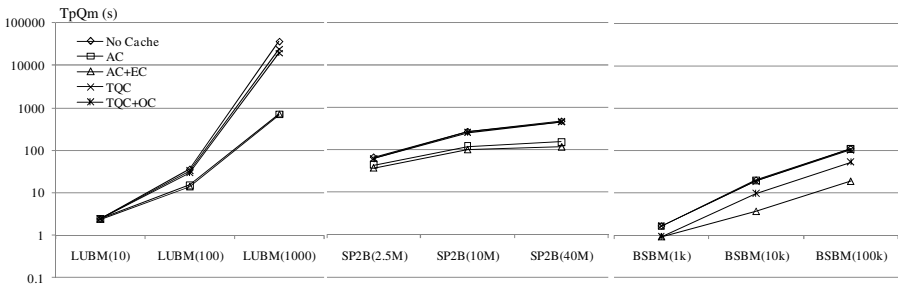


Fig. 6. Comparison with Existing Approaches

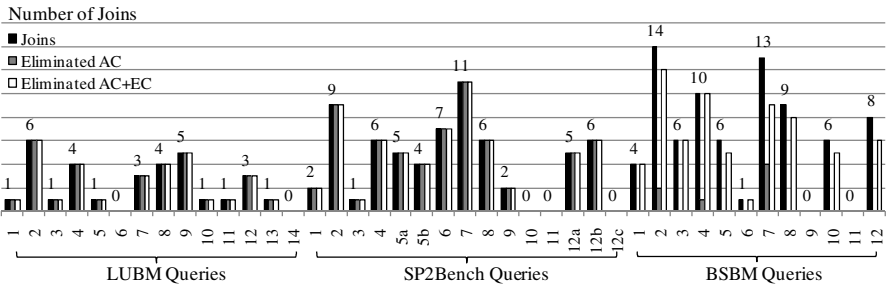


Fig. 7. Eliminated Join Operations

Cache Hit

1. LUBM and SP²Bench. As LUBM and SP²Bench provide static query mixes that contain fixed queries, all the three caching approaches have good cache hit rate on both of the benchmarks.

2. BSBM. AET based Caching has a low hit rate because BSBM generates dynamic SPARQL queries and few of these queries having whole or a part of their corresponding AETs in common. AET based Caching in warm cache phase has a reasonable hit rate.

Comparison with Existing Approaches

Fig. 6 shows the comparison with the caching approaches in [9]. TQC implements the triple-based query caching in [9], and OC implements the application object caching in [9]. For LUBM and SP²Bench, TQC caches triples, where join operations have to be reissued on cache hit, so TQC doesn't work well on LUBM and SP²Bench. OC is designed for applications, where adjacent queries usually have sub-queries in common. But LUBM and SP²Bench mainly focus on testing the throughput limit of RDF databases via SPARQL queries with complex semantics. These two benchmarks don't provide queries in which adjacent ones have a common sub-query. So OC won't work well on LUBM and SP²Bench either. For BSBM things become a little different: TQC doesn't perform well, while OC improves performance at a considerable degree (about 50%). BSBM is a benchmark simulating the application context of an e-store. So test queries in BSBM are intentionally designed to have coherent semantics in a small query context. So OC is applicable in helping improve performance on BSBM.

Fig. 6 also tells the scalability of our caching scheme. In all the 9 dataset at warm cache phase, the system with both of our caching schemes (AC+EC) has the best performance at warm cache phase. Our cache scheme effectively speeds up the processing capacity of Sesame RDF framework. Processing capacity on larger scale RDF datasets (LUBM(1000), SP²Bench(40M), BSBM(100k)) is promoted more than that on datasets of smaller scale. This ensures that RDF stores with our caching scheme have higher upper-bound in data scale.

Eliminated Join Operations

Our caching scheme speeds up query evaluation via elimination of join operations. According to Fig. 7, join operations in LUBM and SP²Bench are mainly eliminated by AET based caching because the test queries in these two benchmarks are static. For BSBM, queries are generated dynamically, thus AET based caching doesn't work well on join elimination. However, entity caching provides a more flexible caching effect, and as is shown, it is able to eliminate most join operations for BSBM test queries. This join elimination ensures the speedup in BSBM query evaluation.

6 Conclusion and Future Work

In this paper, we present a semantic caching scheme that exploit the semantics originated from both SPARQL query and RDF data.

AET based Caching caches intermediate result of SPARQL queries. Cache items are identified by the sub-AET structure, which ensures the identification ability of query semantics. AET based caching effectively improves the evaluation of SPARQL queries with identical sub-query. This caching scheme has a good performance on query context with similar adjacent queries.

Entity Caching extracts basic graph pattern (BGP) from SPARQL queries and groups triple patterns included in the BGP by the subject URIs/variables of the triple patterns. Triple pattern groups are extended to fully select the property values of the object specified by the subject URI. Entity caching is adaptive and efficient. It works well on almost all query contexts.

Future Work

This work is the first step of research work that focuses on improving performance of large-scale Semantic Web applications. Further work includes:

- 1) **Distributed Caching.** Distributed caching has been widely applied in mainstream web 2.0 applications. As the Semantic Web develops, centralized local caching will not work on very large-scale Semantic Web data. Distributed semantic caching will therefore become a must in distributed Semantic Web data infrastructure.
- 2) **Storage and Indexing Technique.** The entity caching speedup query evaluation via a new storage and indexing scheme of Semantic Web objects. This is applicable on the design of storage and indexing of an RDF database.

References

1. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
2. Wilkinson, K., et al.: Efficient RDFStorage and Retrieval in Jena2 (2003)
3. Owens, A., Seaborne, A., Gibbins, N., Schraefel, M.: Clustered TDB: A Clustered Triple Store for Jena (2008), <http://eprints.ecs.soton.ac.uk/16974/1/www2009fixedref.pdf>

4. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In: Conference on Social Semantic Web, pp. 59–68 (2007)
5. Neumann, T., Weikum, G.: RDF-3X: ARISC-Style Engine for RDF. Proc. VLDB Endow 1(1), 647–659 (2008)
6. 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)
7. Chong, E.I., et al.: An Efficient SQL-based RDF Querying Scheme. In: VLDB (2005)
8. Altinel, M., Luo, Q., Krishnamurthy, S., Mohan, C., Pirahesh, H., Lindsay, B.G., Woo, H.: DBCache: Database Caching for Web Application Servers. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (2002)
9. Martin, M., Unbehauen, J., Auer, S.: Improving the performance of Semantic Web Applications with SPARQL Query Caching. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 304–318. Springer, Heidelberg (2010)
10. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.J.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB (2007)
11. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics 3(2-3), 158–182 (2005)
12. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP²bench: A SPARQL Performance Benchmark (June 2008)
13. Castillo, R.: RDFMatView: Indexing RDF Data for SPARQL Queries. In: ISWC (2010)
14. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. International Journal on Semantic Web and Information Systems (2009)
15. Dar, S., Franklin, M.J., Jónsson, B.T., Srivastava, D., Tan, M.: Semantic Data Caching and Replacement. In: Proceedings of the 22th International Conference on Very Large Data Bases (VLDB 1996), pp. 330–341. Morgan Kaufmann Publishers Inc., San Francisco (1996)
16. Li, L., König-Ries, B., Pissinou, N., Makki, K.: Strategies for Semantic Caching. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, pp. 284–298. Springer, Heidelberg (2001)
17. Sakr, S., Al-Naymat, G.: Relational processing of RDF queries: a survey. SIGMOD Rec. 38(4), 23–28 (2010)

Evaluating Graph Traversal Algorithms for Distributed SPARQL Query Optimization

Xin Wang, Thanassis Tiropanis, and Hugh C. Davis

Electronics and Computer Science
University of Southampton
{xw4g08, tt2, hcd}@ecs.soton.ac.uk
<http://www.ecs.soton.ac.uk/>

Abstract. Distributed SPARQL queries enable users to retrieve information by exploiting the increasing amount of linked data being published. However, industrial-strength distributed SPARQL query processing is still at its early stage for efficiently answering queries. Previous research shows that it is possible to apply methods from graph theory to optimize the performance of distributed SPARQL. In this paper we describe a framework that can simulate arbitrary RDF data networks to evaluate different approaches of distributed SPARQL query processing. Using this framework we further explore the graph traversal algorithms for distributed SPARQL optimization. We present an implementation of a Minimum-Spanning-Tree-based (MST-based) algorithm for distributed SPARQL processing, the performance of which is compared to other approaches using this evaluation framework. The contribution of this paper is to show that a MST-based approach seems to perform much better than other non graph-traversal-based approaches, and to provide an evaluation framework for evaluating distributed SPARQL processing.

Keywords: SPARQL, Linked Date, distributed query processing, graph theory.

1 Introduction

Linked Data is interlinked RDF data that enables users to retrieve quality information from cross-domain data sources. With the emergence of the Semantic Web, a large volume of Linked Data is published by an increasing number of providers [2]. Meanwhile, frameworks for storing and querying Linked Data [28] are available for developers and users. As Linked Data is distributed in nature, it is inevitable to involve multiple RDF data sources when querying Linked Data. Linked Data can be queried using SPARQL [1], which is the standard query language of RDF having the capability of matching graph patterns [20]. SPARQL supports queries over multiple RDF graphs. However, it requires downloading all the data to a local storage which is not feasible for a number of reasons. For example, the total data size may be too large; it may be difficult to keep

¹ <http://www.w3.org/TR/rdf-sparql-query/>

the downloaded data up-to-date; the access to data may be restricted, and etc. [21]. In addition, the increasing scale of the Linked Data cloud poses further challenges to distributed SPARQL queries, for instance, the cost and latency of transferring data are considerably high, and also localisation of particular data is difficult. Therefore, it is necessary to investigate distributed SPARQL processing and explore optimization techniques to improve its performance.

Several approaches of distributed SPARQL query optimization have been proposed, including query rewriting [11], selectivity-based triple pattern reordering [1,25], scalable indexing strategy [7,14] and general methods for SPARQL optimization [24], and on distributed SPARQL query such as [26,21,15,27]. In spite of the approaches mentioned above, optimization techniques of distributed SPARQL queries are still at the early stage on both scalability and efficiency [8,10]. The problem of distributed query processing has also been well studied in the distributed database systems field [17,12]. It is not straightforward to apply optimization techniques of distributed database systems to distributed SPARQL because of the differences between the data representation (i.e. relations and triples) and the query languages (i.e. SPARQL and SQL) [6]. However, distributed SPARQL's particular characteristics, such as its graph structure, make it possible to benefit from other techniques such as graph traversal algorithms. The authors in [27] consider using graph-traversing algorithms, which are used to search minimum-weight path to traverse a graph, to construct the optimal execution plan of distributed SPARQL queries. In their approach, each Basic Graph Pattern (BGP) is regarded as a directed graph, in which subjects and objects in the BGP are nodes and predicates are edges. The weight of each edge is the cost of evaluating the corresponding triple pattern. The optimal query plan corresponds to the minimum spanning tree (MST) of this graph which is generated by graph-traversing algorithms.

The authors of [27] provide a brief idea of applying graph traversal algorithms for distributed SPARQL optimization. In this paper, we further explore the applicability of graph traversal algorithms for distributed SPARQL optimization. To achieve this goal, we implement a distributed SPARQL engine using a graph traversal algorithm and compare its performance with existing approaches. To perform the comparison and potentially other comparisons in the future, an evaluation environment for distributed SPARQL is required. Although several benchmarks have been proposed for SPARQL such as SP²Bench [23,22] and Berlin SPARQL Benchmark (BSBM) [3], only one [10] is designed for a distributed environment and it solely provides information of software aspects. Therefore, we also proposed and implemented an evaluation framework for our comparison purpose.

In summary, our contribution is in:

- Describing and deploying an evaluation framework which is capable of simulating a distributed environment of RDF datasets having different distribution, and measuring relevant costs.
- Implementing a distributed SPARQL engine building on the approach proposed in [27] using Prim's algorithms [19] to find the MST.

- Comparing our SPARQL engine with DARQ [21] using the evaluation framework. The result shows that graph traversal algorithms can significantly improve the performance of distributed SPARQL processing.

The remainder of this article is organized as follows. We introduce the design of the evaluation framework in section 2, followed by the implementation of our SPARQL engine in section 3. The comparison between our SPARQL engine and DARQ are described and analysed in section 4. Finally, we provide our conclusion and future plan in section 5.

2 The Evaluation Framework

Although several benchmarks, such as SP²Bench [22] and BSBM [3], have been proposed for evaluation of centralized SPARQL query performance, there is no satisfied methods to evaluate approaches of distributed SPARQL queries. In this section we present a general evaluation framework [2] which is capable of the following:

- Generating different sized data that obey particular distributions (e.g. normal distribution or the current distribution of data in Linked Data cloud).
- Simulating networks containing an arbitrary number of SPARQL endpoints that hold the generated datasets.
- Autonomously evaluating distributed SPARQL engines.
- Producing unified and comprehensive evaluation reports.

BSBM is a widely used benchmark for SPARQL which provides several sophisticated metrics, a set of queries, benchmark specification and tools that help run evaluation. In order to increase interoperability, our evaluation framework uses part of the BSBM metrics, datasets, and queries. In addition, several metrics that cover hardware aspects are contained. Besides metrics, the framework contains components to simulate RDF networks and evaluate distributed SPARQL engines. Furthermore, virtual machines are used instead of real machines to gain the ability of simulating arbitrary sized RDF networks. Evaluation reports generated by the framework follow the BSBM specification [3].

2.1 Design of the Evaluation Framework

The evaluation framework contains five components: *data generator*, *data splitter*, *statistics collector*, *data distributor* and *testdriver*. The structure of the evaluation framework is shown in figure 1 and the components are described below.

² Source code is available at

<http://code.google.com/p/dis-sparql-evaluation-framework/>

³ <http://www4.wiwiw.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/>

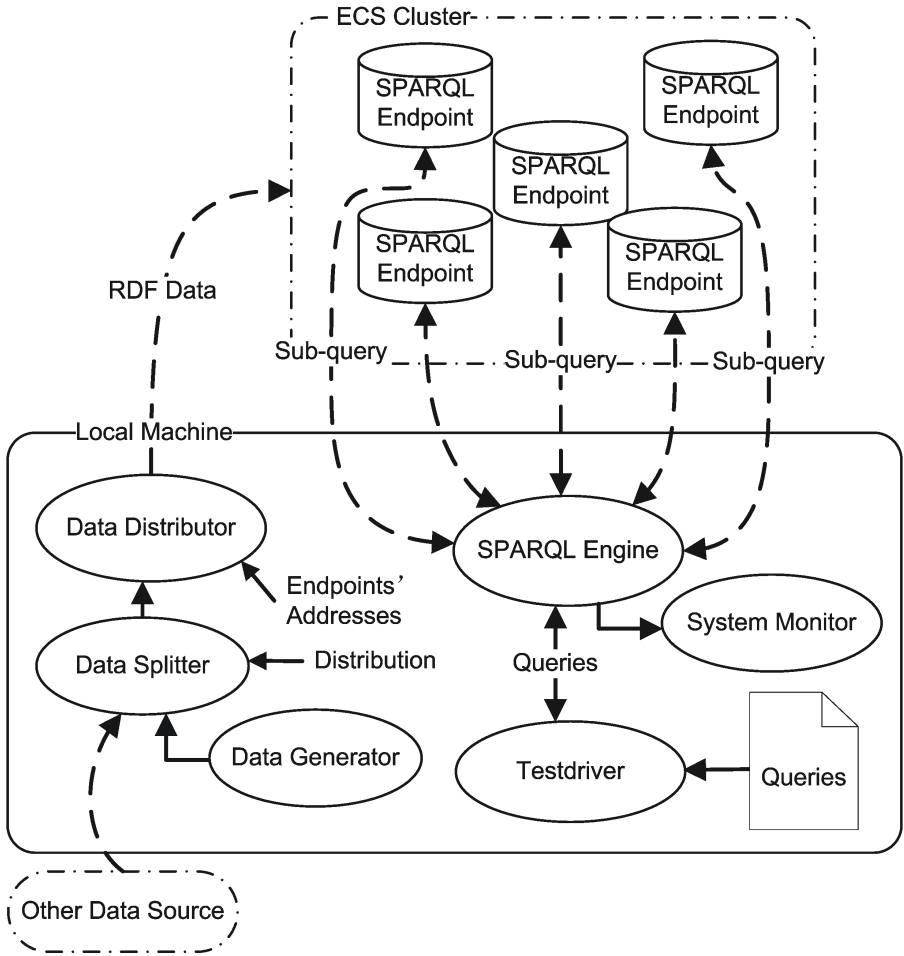


Fig. 1. Solid arrows denote local data flow while dotted arrows denote remote data flow. The *data splitter* splits data from either the *data generator* or the Linked Data cloud according to a given distribution. Then the *data distributor* uploads data to SPARQL endpoints. After the endpoints are ready, the *testdriver* reads queries from a file and calls the distributed SPARQL engine to process these queries. The engine processes the queries and returns results to the *testdriver*. The *testdriver* records the time of query processing and generates a performance report. Meanwhile the *system monitor* records the memory and CPU usage and the network flow.

Data Generator. The BSBM benchmark provides a *data generator* which can generate arbitrary sized datasets according to an abstract data model and data production rules. Using this *data generator* in our framework provides the ability to generate data in a controlled manner. Also, as we follow the BSBM specification, data generated by this *data generator* can be easily evaluated by the *testdriver* of the framework, which can evaluate the performance of SPARQL endpoints.

Data Splitter. The *data splitter* accepts any RDF data file and splits it into certain number of data files according to particular distributions. The supported distributions include uniform distribution, normal distribution and the current distribution of data in the Linked Data cloud in terms of dataset size; other considerations such as the distribution of referenced URIs were not taken into account as this stage.

Data Distributor. After the data is split, the *data distributor* is used to dispatch data to remote datasets. It accepts a list of pairs, each of which contains the address of a dataset and the path of data going to that dataset.

Statistics Collector. The *statistics collector* reads local RDF data files or accesses remote RDF datasets and then reports basic statistics of them. The statistics contain the number of triples and the number of each predicate in a data file/dataset. The *statistics collector* can be used to check the distribution of a RDF network and to provide statistics to distributed SPARQL engines.

Testdriver. The BSBM also provides a *testdriver* which can evaluate query performance of SPARQL endpoints. Since most approaches of distributed SPARQL processing are not implemented as SPARQL endpoints, we modify the *testdriver* to support issuing queries to and processing results from distributed SPARQL engines. Performance reports of query executing are automatically generated by the *testdriver*. Along with the *testdriver*, system and networks monitors are included in the framework to provide information of system and network performance.

Metrics. The first two which are part of BSBM's fundamental metrics⁴ are used to measure the speed of query execution. Meanwhile the evaluation framework considers the system loads of query evaluation. CPU usage is one important aspect of measuring system load. Besides, memory usage and networks data flow are considerable for distributed query processing [16,17]. Therefore CPU usage, memory usage and networks traffic are contained in our evaluation framework. The metrics we use are:

- Queries per Second (QpS).
The average number of queries that can be executed per second.
- Overall Runtime (oaRT).
The overall time to execute a certain amount of queries.

⁴ <http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/BenchmarkRules/>

- Memory Usage (MU).
The memory usage of the engine during the evaluation.
- CPU Usage (CU).
The CPU usage of the engine during the evaluation.
- Network Flow (NF).
Uploading and downloading data of the engine during the evaluation.

When we use the framework to evaluate an approach of distributed SPARQL, firstly the RDF data is obtained from the Linked Data cloud or generated by the *data generator*. Once the data is ready, it is split into pieces by the *data splitter* according to a certain distribution, and also basic statistics of each piece of data can be collected by the *statistics collector*. After that, the *data distributor* dispatches each piece of data to certain remote datasets. Then the distributed SPARQL approach can be evaluated by the *testdriver* against those remote datasets.

3 Graph Traversal Algorithm for Distributed SPARQL Query Optimization

BGP is the basic block of SPARQL queries. As a BGP can be represented as a graph, evaluating a BGP can be regarded as the same process as traversing its graph representation. Therefore distributed SPARQL optimization process equals to constructing the MST, which is the spanning tree of a graph with the minimum weight, of a BGP graph (refer to [27] for details). In this section, we briefly discuss several graph traversal algorithms. And then we discuss the evaluation strategy and decomposition strategy of distributed SPARQL processing in general. Finally we present the design and implementation of a Distributed SPARQL engine using Prim's algorithm, which is called DSP engine for convenience.

3.1 Graph Traversing Algorithms

There are several algorithms for constructing MST, such as Borůvka's algorithm, Prim-Jarník's algorithm, and Kruskal's algorithm. The MST grows from one or several disconnected components during construction. An edge with exactly one endpoint in a MST component is called a frontier edge of this component, and all of the three algorithms construct MST by adding the minimum frontier edge to the component(s) to which it belongs: Borůvka's algorithm grows the MST from several components and adds all minimum frontier edges simultaneously; Prim-Jarník's algorithm grows the MST from a single component and therefore adds one minimum frontier edge at a time; and Kruskal's algorithm grows the MST (probably) from multiple components and adds minimum frontier edges one by one in increasing weight order.

Prim-Jarník’s algorithm traverses a graph one node after another from a single node, which result in evaluating triple patterns as a chain (i.e. the triple patterns in every two adjacent steps have at least one part in common). In this case bind join [9] can be adopted to reduce intermediate results. On the contrary, Kruskal’s algorithm can not be used with bind join as it may not evaluate triple patterns as a chain. Borůvka’s algorithm is capable of executing several triple patterns simultaneously to reduce query execution time, however intermediate results may be increased which can lead to an increase in network traffic. In our implementation we focus on reducing network flow, and therefore we combine Prim’s algorithm and bind join together to achieve this purpose.

3.2 Distributed SPARQL Processing Evaluation Strategy

Intuitively, distributed SPARQL processing is equivalent to the union of individual SPARQL queries over multiple remote RDF datasets. A similar idea can be found in [13]. It regards SPARQL query processing over a set of Linked Data sources as evaluating the queries on the graph which is the union of graphs from all data sources. If we consider the fact that duplication may exist in different data sources, naturally we will ask whether the union between data sources is *bag union* (allows duplication) or *set union* (does not allow duplication)? In [4] the graph containing all the triples from involved datasets is constructed using RDF merge, which is a set union, and this satisfies the definition of RDF graph (a set of triples). However, most existing distributed SPARQL approaches such as [21][27][13] do not eliminate results from duplicate data. Furthermore, allowing results from duplicate data has the following advantages:

- It retains statistical information of certain triples.
- Eliminating duplications is simple after query execution.
- It makes implementation of distributed SPARQL engines easier.

Therefore, in this article definition [1] is applied for evaluation of a distributed SPARQL query. We use \bigcup^B to denote bag union.

Definition 1. *Evaluation of a distributed SPARQL query Q over a set of data sources \mathcal{G} , denoted by $[[Q]]_{\mathcal{G}}$, is defined as $[[Q]]_{\mathcal{G}} = [[Q]]_G$, where $G = \bigcup_{g_i \in \mathcal{G}}^B g_i$, i is the URI of graph g_i .*

3.3 Decomposition Strategy

We assume that individual dataset cannot provide all the data that a distributed SPARQL query needs. This assumption implies that sending a whole query to any single dataset probably returns no result. Therefore the major purpose of query decomposition is to decompose the original query into sub-queries [5] which has the chance to have one or more datasets that can provide data they require.

⁵ A sub-query may contain only one triple pattern.

Furthermore, by query decomposition there is a possibility to reduce query processing time, cost of sending out queries, and cost of transferring intermediate data. The reduction can be achieved by sending as many sub-queries as possible together to one dataset, which is adopted in most existing approaches of distributed SPARQL processing. However, the possibility of doing that is low. According to [18] we have:

Lemma 1. *The evaluation of two joined sub-queries denoted by $(q_1 \text{ AND } q_2)$ over two datasets d_1, d_2 denoted by $[[q_1 \text{ AND } q_2]]_{d_1 \cup d_2}$ is $([[q_1]]_{d_1} \cup [[q_1]]_{d_2}) \bowtie ([[q_2]]_{d_1} \cup [[q_2]]_{d_2})$.*

For convenience, we call a join that merges results from different datasets (e.g. $[[q_1]]_{d_2} \bowtie [[q_2]]_{d_1} \bowtie [[q_3]]_{d_1}$) a cross-dataset join. From lemma 1 we can have:

Proposition 1. *Sending n sub-queries $q_1 \dots q_n$ together to m datasets $d_1 \dots d_m$ does not affect the results iff all cross-dataset joins, denoted by $\bowtie_{i \in [1, n]} [[q_i]]_{d_{j_i}}$ for $j_i \in [1, m]$ and not all j_i have the same value, are empty.*

Proposition 1 implies that if a sub-query can be evaluated against more than one dataset, it cannot be sent with other sub-queries⁶. For instance, in [21] each triple pattern that can be answered by more than one data source are sent to those data sources separately. Furthermore, sending several triple patterns together to a data source does not guarantee reducing the intermediate results, because the intermediate results are produced by joining the results of each triple pattern. These two reasons (especially the former one) make it unfeasible to send more than one triple pattern together to a dataset. Therefore, the importance of query decomposition falls on reducing the number of datasets that a triple pattern is sent to. In addition, query optimization should take per triple pattern as the basic unit.

3.4 Implementation

The DSP engine⁷ is based on Prim's algorithm. A similar conceptual pseudo-code is presented in [27]. However, several issues exist in the proposed pseudo-code, such as the sequence of popping out a new triple pattern, pruning filler edges and retrieving new binding will not produce the right result⁸, and also it is not aware of keeping separate data copies for different iterations. Our algorithm (shown in algorithm 1) adopts the evaluation and decomposition strategy discussed above and addresses the issues of the pseudo-code proposed in [27].

The algorithm accepts triple patterns and returns a set of bindings, each of which is a mapping from variables to their values (i.e. a solution mapping). Edges

⁶ In some cases sub-queries that can be evaluated against more than one dataset may be able to be sent together, however these cases are complex for computation.

⁷ Source code of our implementation can be found at

<http://code.google.com/p/gdsparal/>

⁸ Pruning should be done after retrieving new binding and before popping out a new triple.

containing both visited and unvisited nodes are called frontier edges. All frontier edges are kept in a min-heap (i.e. whose root is the minimum node) (line 13). In each iteration, the frontier triple pattern having the minimum cost is selected and then evaluated (line 6). A binding which is produced in previous iterations is kept for following iterations. Bind join is adopted, and therefore the triple pattern is firstly bound with the current binding (i.e. replacing variables in the triple pattern by corresponding values in the current binding), and then evaluated in remote repositories (line 9). The cost of new frontier edges connecting with the current edge e is calculated using the actual number of bindings of e (line 10 to line 12). In each iteration, the binding is extended to contain new results, and separate data copies are maintained for the following iterations (line 18). If a triple pattern in the heap has all its variables bound by current binding, the bound triple pattern is evaluated as an *ASK* query to verify the binding. The iteration is terminated if the binding fails verification, otherwise the triple pattern is removed (line 11). Contrary to the pseudo-code presented in [27], our pruning process is after retrieving new binding and before popping out a new triple, which guarantees that the newly popped triple is not bound by the current binding. The current binding is added to the final solution set once the heap is empty.

Our algorithm optimizes the evaluation of BGPs, and therefore it is applicable for any kind of query with all operators (e.g. *UNION*, *OPTIONAL* etc.). The query interface of the DSP engine is the same as DARQ's, which accepts SPARQL query strings and returns the results as a table. The *testdriver* is modified accordingly, which enables us to evaluate the DSP engine and DARQ as evaluating normal SPARQL endpoints.

4 Evaluation

In order to explore the potential of the DSP engine, we compare it with DARQ which is the only implementation available among relevant research. The comparison is carried out in an environment that contains a small scale network, and moderate sized data and queries. Initially we use all 12 queries of BSBM to test DARQ and the DSP engine. However, we find that DARQ does not always return correct results for queries with *OPTIONAL* and *UNION* clauses while DSP engine does. Query 1 and query 10 are the only queries of BSBM for which we can confirm the accuracy of results of both engines. Furthermore, DARQ does not return result for query 10 after a large amount of time, therefore we present detailed results of query 1 and brief results of query 10. The comparison of the DSP engine and DARQ was carried out in the evaluation framework shown in figure 1 on page 213. The following sections detail the evaluation and results.

4.1 Configuration of the Evaluation Environment

In the evaluation environment, 6 SPARQL endpoints (Sesame 2.3.2⁹) are hosted on 5 VMs, containing 104046, 93071, 74205, 53556, 34045 and 19356 triples

⁹ <http://www.openrdf.org/>

Algorithm 1. Pseudo-code of the DSP engine

```

input : unvisited stores unvisited triple patterns
input : heap stores all the frontier edges
input : binding stores a solution mapping of several variables
input : bindingNum stores nodes and corresponding number of bindings
output: solutions stores all valid bindings
1  prune(heap);
   // remove all bound triple patterns in heap
2  if heap.isEmpty() then
3    solutions.add(binding);
4    return;
5  end
6  minTriple ← heap.pop();
7  preNode ← minTriple.getBoundNode();
   // the subject or object that is bound in current triple pattern
8  currentNode ← minTriple.getFreeNode();
9  currentBindings ← execSel(minTriple, binding);
   // evaluate minTriple according to existing binding
10 bindingNum ← (currentNode,
    currentBindings.size()*bindingNum.get(preNode));
11 for each triple pattern t connected with currentNode do
12   t.setCost(bindingNum.get(currentNode)*t.getWeight());
13   heap.insert(t);
14   unvisited.remove(t);
15 end
16 for each binding b in currentBindings do
17   nextHeap ← heap.clone();
   // keep separate copies for each iteration
18   nextBinding ← binding.add(b).clone();
19   nextUnvisited ← unvisited.clone();
20   this(nextHeap, nextBinding, bindingNum, nextUnvisited);
   // recurs
21 end
22 return;

```

Table 1. The queries for testing*Query 1*

```

SELECT ?product ?label
WHERE {
  ?product rdfs:label ?label .
  ?product a bsbm:Product .
  ?product bsbm:productFeature %ProductFeature1% .
  ?product bsbm:productPropertyNumeric1 ?value1 .
  FILTER (?value1 > %x%)
}
ORDER BY ?label
LIMIT 10

```

Query 10

```

SELECT DISTINCT ?offer ?price
WHERE {
  ?offer bsbm:product %ProductXYZ% .
  ?offer bsbm:vendor ?vendor .
  ?offer dc:publisher ?vendor .
  ?vendor bsbm:country <http://downlode.org/rdf/iso-3166/countries#US> .
  ?offer bsbm:deliveryDays ?deliveryDays .
  FILTER (?deliveryDays <= 3)
  ?offer bsbm:price ?price .
  ?offer bsbm:validTo ?date .
  FILTER (?date > %currentDate% )
}
ORDER BY xsd:double(str(?price))
LIMIT 10

```

respectively, which obey normal distribution. The two distributed SPARQL engines under testing are run on a machine having an Intel Xeon W3520 processor, 12 GB memory and 1Gbps LAN.

The queries in table 1 are used for testing. The values enclosed by “%” are generated randomly during testing. The evaluation framework is configured to perform 5 warm up runs and 50 testing runs. Query results of the last 50 runs (i.e. the testing runs) are recorded, and system resource load and networks traffic are recorded for all the 55 runs.

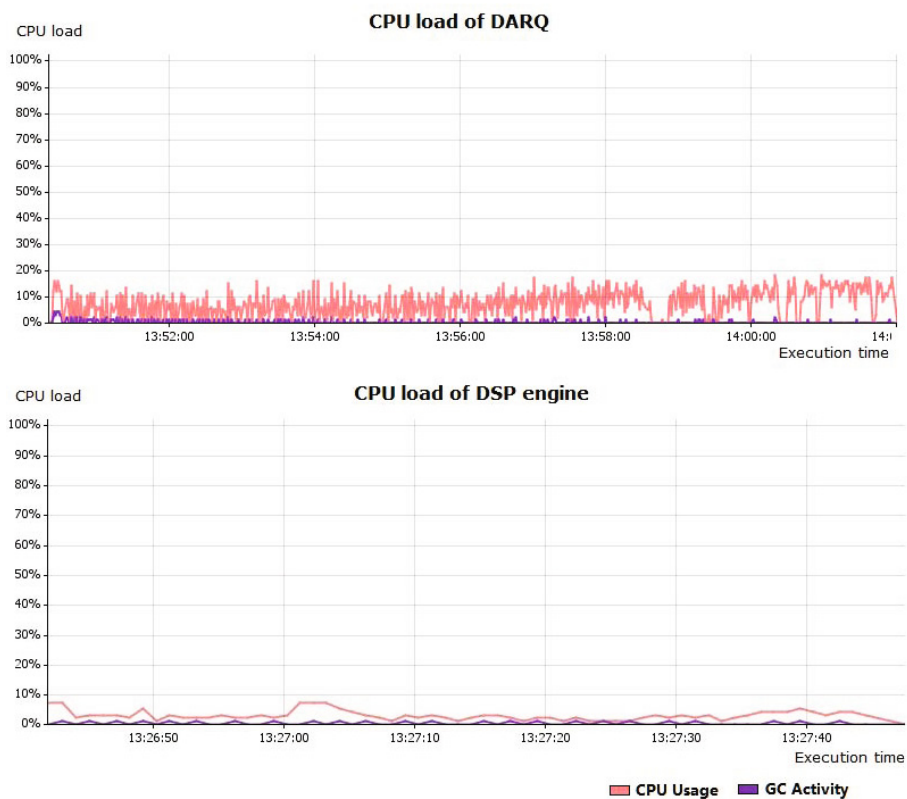
4.2 Evaluation Results

The evaluation result of query 1 is shown in table 2. Details of CPU and memory usage is shown in figures 2 and 3.

Query 10 takes both engines quite a long time to process. On average, the DSP engine spends 220 seconds to process one instance of query 10, while DARQ does not finish processing within 400 seconds and it times out. Meanwhile, DARQ’s cost on CPU, memory and network bandwidth overruns DSP engine.

Table 2. Evaluation result

	DARQ	DSP engine
Overall runtime (s)	606.273	41.826
Queries per Second	0.08	1.20
Min/Max query runtime (s)	6.7576/26.3826	0.0148/3.3272
Average result count	15.82	15.82
CPU usage (%)	15	5
Memory usage (MB)	20 - 60	9
Up/Down network flow (MB)	39.32/14.09	0.88/0.42

**Fig. 2.** CPU load

4.3 Analysis

From the above results it can be concluded that the DSP engine performs much better than DARQ on both query processing time and system resource usage for both BSBM queries 1 and 10, for which a comparison with DSP was possible.

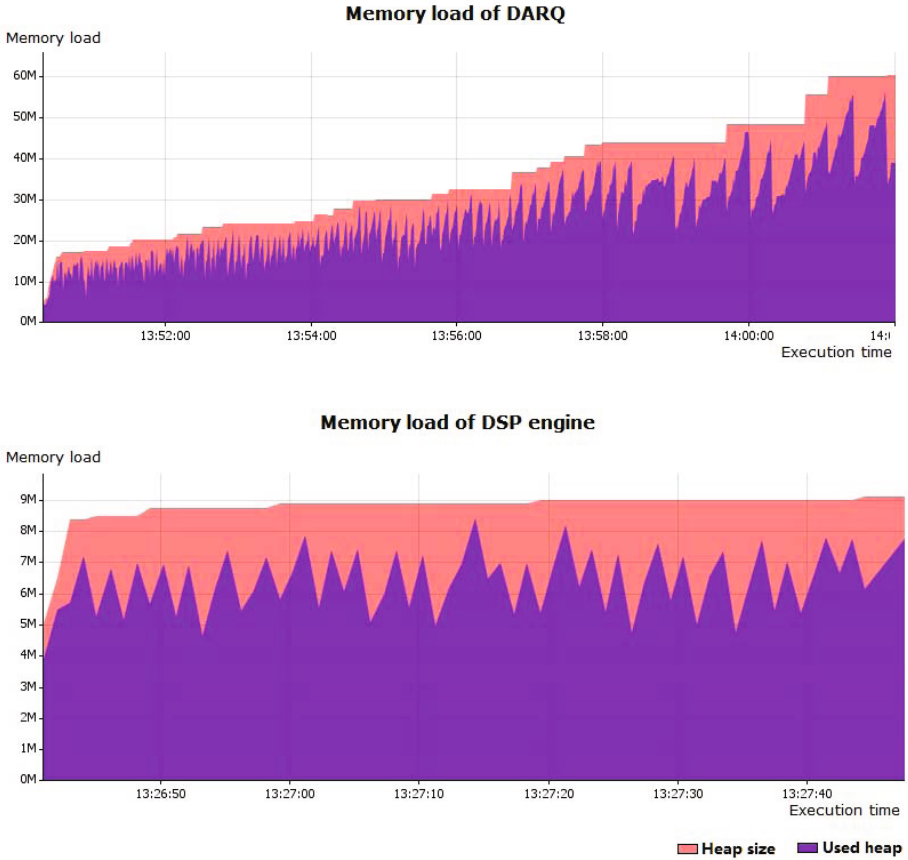


Fig. 3. Memory load

One probable reason is that our decomposition strategy does not group triple patterns, therefore all triple patterns together rather than sub-groups of triple patterns are considered during optimization. In addition, the use of bind join and the pruning mechanism (line 11 of algorithm 1) together can eliminate invalid results at an early stage during the evaluation. The high system load of DARQ may be due to the fact that it generates all possible query execution plans and then searches for the optimal one. On the contrary, our engine computes the optimal query plan using Prim’s algorithm rather than enumerating all possible query plans. Both DARQ and our implementation show that the outgoing data is more than the incoming data under current testing environment. This implies that reducing the number of sub-queries is also an important aspect to decrease the traffic, and availability of more accurate service descriptions will be useful for reducing sub-queries.

5 Discussion and Future Work

This work aims to address performance in environments of varying scale and data distribution and to provide an framework on which different approaches can be evaluated. In this paper, we have presented an evaluation framework which can be used to analyze and compare various approaches of distributed SPARQL processing, the design and implementation of DSP engine which is a distributed SPARQL engine using Prim's algorithm, and a comparison between DARQ and DSP engine which shows the potential of MST algorithms for distributed SPARQL processing. Our comparison shows encouraging results to support that graph traversal algorithms are applicable for distributed SPARQL query optimization. However there are still several aspects that require further investigation. The lack of accurate service descriptions and cost models is one drawback of algorithm performance.

In addition, the algorithms themselves have limitations. MST algorithms such as Prim's algorithm traverses a graph node by node, which implies the lower bound of query processing time is the sum of each triple patterns in the query. Unfortunately, we observed that executing triple patterns in remote datasets is the most time consuming process. Therefore the execution time may be unacceptable when executing a complex query having many triple patterns. One possible solution is to explore mechanisms that enable executing multiple triple patterns in parallel.

Another challenge is from co-reference phenomenon. In the Linked Data cloud a particular URI identifies one unique resource, however, one resource can be identified by multiple URIs. The more data are published in Linked Data cloud, the more common co-reference is [5]. There is an urgent requirement to develop solutions that do support queries with co-references. As future work, we plan to explore graph traversal algorithms to optimize distributed SPARQL queries having co-references and to explore parallel query execution.

References

1. Bernstein, A., Kiefer, C., Stocker, M.: OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation (2007)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. *International Journal on Semantic Web and Information Systems* 5(3), 1–22 (2009)
3. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. *International Journal On Semantic Web and Information Systems-Special Issue on Scalability and Performance of Semantic Web Systems* (2009)
4. Bouquet, P., Ghidini, C., Serafini, L.: Querying the Web of Data: A Formal Approach. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 291–305. Springer, Heidelberg (2009)
5. Correndo, G., Salvadores, M., Millard, I., Glaser, H., Shadbolt, N.: SPARQL query rewriting for implementing data integration over linked data. In: *Proceedings of the 2010 EDBT Workshops*, pp. 1–11. ACM (2010)
6. Cyganiak, R.: A relational algebra for SPARQL. Digital Media Systems Laboratory, HP Laboratories Bristol, pp. 2005–170 (2005)

7. Fletcher, G., Beck, P.: Scalable indexing of RDF graphs for efficient join processing. In: *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pp. 1513–1516. ACM (2009)
8. Gray, A., Gray, N., Ounis, I.: Can RDB2RDF Tools Feasibly Expose Large Science Archives for Data Integration? 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. 491–505. Springer, Heidelberg (2009)
9. Haas, L., Kossmann, D., Wimmers, E., Yang, J.: Optimizing queries across diverse data sources. In: *Proceedings of The International Conference on Very Large Data Bases*, pp. 276–285. Citeseer (1997)
10. Haase, P., Mathäß, T.: An evaluation of approaches to federated query processing over linked data. In: *Proceedings of the 6th International Conference on Semantic Systems*, pp. 1–9. ACM (2010)
11. Hartig, O., Heese, R.: The SPARQL Query Graph Model for Query Optimization. In: Franconi, E., Kifer, M., May, W. (eds.) *ESWC 2007*. LNCS, vol. 4519, pp. 564–578. Springer, Heidelberg (2007)
12. Kossmann, D.: The state of the art in distributed query processing. *ACM Computing Surveys (CSUR)* 32(4), 422–469 (2000)
13. Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I*. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
14. McGlothlin, J., Khan, L.: RDFJoin: A Scalable Data Model for Persistence and Efficient Querying of RDF Datasets. *Database* (2009)
15. Obermeier, L., Nixon, L.: A Cost Model for Querying Distributed RDF-Repositories with SPARQL. In: *Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense Tenerife*. Citeseer (2008)
16. Özsu, M.: Distributed and parallel database systems. *ACM Computing Surveys (CSUR)*, 1–21 (1996)
17. Özsu, M., Valduriez, P.: *Principles of distributed database systems*. Prentice Hall (1999)
18. Pérez, J., Arenas, M.: *Semantics and Complexity of SPARQL*. *ACM Transactions on Database Systems, TODS* (2009)
19. Prim, R.: Shortest connection networks and some generalizations. *Bell System Technical Journal* 36(6), 1389–1401 (1957)
20. Prud’Hommeaux, E., Seaborne, A.: *SPARQL query language for RDF* (2008)
21. Quilitz, B.: Querying Distributed RDF Data Sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
22. Schmidt, M., Hornung, T., Küchlin, N., Lausen, G., Pinkel, C.: An Experimental Comparison of RDF Data Management Approaches in a SPARQL Benchmark Scenario. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 82–97. Springer, Heidelberg (2008)
23. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: A SPARQL Performance Benchmark. In: *IEEE 25th International Conference on Data Engineering, ICDE 2009*, pp. 222–233. IEEE (2009)
24. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL query optimization. In: *Proceedings of the 13th International Conference on Database Theory*, pp. 4–33. ACM (2010)

25. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: Proceeding of the 17th International Conference on World Wide Web, pp. 595–604. ACM (2008)
26. Stuckenschmidt, H., Vdovjak, R., Houben, G., Broekstra, J.: Index structures and algorithms for querying distributed RDF repositories. In: Proceedings of the 13th International Conference on World Wide Web, pp. 631–639. ACM (2004)
27. Vandervalk, B.P., McCarthy, E.L., Wilkinson, M.D.: Optimization of Distributed SPARQL Queries Using Edmonds' Algorithm and Prim's Algorithm. In: International Conference on Computational Science and Engineering, CSE 2009, vol. 1, pp. 330–337. IEEE (2009)
28. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D., et al.: Efficient RDF storage and retrieval in Jena2. In: Proceedings of SWDB, vol. 3, pp. 7–8 (2003)

BipRank: Ranking and Summarizing RDF Vocabulary Descriptions

Gong Cheng¹, Feng Ji², Shengmei Luo², Weiyi Ge¹, and Yuzhong Qu¹

¹ State Key Laboratory for Novel Software Technology, Nanjing University, China

² Communication Services R&D Institute, ZTE Corporation, China
{gcheng,yzqu}@nju.edu.cn, {ji.feng,luo.shengmei}@zte.com.cn,
geweiyi@gmail.com

Abstract. When searching for RDF vocabularies, users often feel hindered by the lengthy description of a retrieved vocabulary from judging its relevance. A natural strategy for dealing with this issue is to generate a summary of the vocabulary description that compactly carries its main theme and reveals its relevance to the user's information need. In this paper, we present a new solution to this problem of vocabulary summarization, which has been defined as ranking and selecting RDF sentences in our previous work. Firstly, we propose a novel bipartite graph representation of vocabulary description, on which we carry out a stochastic analysis of a random surfer's behavior, from which we derive a new centrality measure for RDF sentences called BipRank. Further, we improve it by investigating the patterns of RDF sentences and employing their statistical features. Then, we combine BipRank with query relevance and cohesion metrics into an aggregate objective function to be optimized for the selection of RDF sentences. Our experiments on real-world vocabularies demonstrate the superiority of our approach to the baseline, and also validate its scalability in practice.

Keywords: Cohesion, query relevance, random surfer model, ranking, vocabulary summarization.

1 Introduction

An RDF vocabulary defines a collection of terms, namely classes and properties, which convey data semantics between Web applications. Application developers are encouraged to reuse existing vocabularies for achieving better interoperability. However, when they interact with a vocabulary repository, e.g. browsing the results of a vocabulary search, they are often overloaded with many candidates and their lengthy descriptions, and can hardly make appropriate selections in an effective yet efficient manner. By comparison, concise vocabulary summaries could facilitate vocabulary understanding and then support decision making. Existing approaches mainly operate on two kinds of unit of vocabulary. On the one hand, fruitful research has been carried out into ranking and selecting key terms from a vocabulary as a summary [9,11,15,17]. This paradigm is followed by

some major search engines such as Swoogle¹ and Watson². On the other hand, vocabulary description — represented as an RDF graph — is partitioned into a collection of so-called “RDF sentences”, from which salient ones are selected as a summary [10,16] that aims at reflecting the main theme of the vocabulary with central components. Although having not been extensively investigated, the latter is preferred in this study because it captures not only terms but also relationships between them.

Following this line of research, we (1) propose BipRank, a new variant of the random surfer model for characterizing the centrality (as a kind of salience) of RDF sentence, based on a novel bipartite graph representation of vocabulary description. Different from our previous work [16] which examines the syntactic roles of terms in RDF sentences/triples (i.e. as subject, predicate or object), this new model differentiates between intensional and extensional descriptions of terms, and further exploits the patterns of RDF sentences. In the evaluation, we observe a strong positive correlation between the salience given by our measure and handcrafted gold standards. Then, for supporting vocabulary search, we (2) devise an approach to vocabulary summarization that combines BipRank with two other metrics, namely query relevance and cohesion. In a search-based evaluation comprising real-world vocabularies and users, the summaries generated by our approach are favored by more participants. We have incorporated the approach into a practical vocabulary search engine³.

This work is an extension of our poster presentation [2]. In the remainder of this paper, Sect. 2 gives the problem statement. Section 3 introduces our salience measure. Section 4 describes our approach to vocabulary summarization. Section 5 presents experimental results. Section 6 discusses related work. Finally, Sect. 7 concludes the paper with future work.

2 Problem Statement

An RDF vocabulary v defines a set of terms, denoted by $\text{Terms}(v)$. All their descriptions are collectively characterized as an RDF graph, denoted by $\text{Graph}(v)$. An RDF graph, as a set of RDF triples, has a unique finest partition satisfying that RDF triples sharing common blank nodes are in the same part [13,16], which corresponds to an equivalence relation. Each part, being an RDF graph by itself, is called an RDF sentence [16], a.k.a. minimum self-contained graph [13]. Let $\text{Sents}(v)$ be such partition of $\text{Graph}(v)$, which is a set of RDF sentences and can be computed by traversing $\text{Graph}(v)$ only once. For a toy vocabulary vin , $\text{Graph}(vin)$ and $\text{Sents}(vin)$ are illustrated in Fig. 1.

Following [16], a summary of a vocabulary v , denoted by S , is defined as a subset of $\text{Sents}(v)$. Summaries are usually under some length constraint. Here, the number of RDF sentences is a natural but inappropriate constraint, because one single RDF sentence may comprise one or many RDF triples, as shown in

¹ <http://swoogle.umbc.edu/>

² <http://watson.kmi.open.ac.uk/>

³ <http://ws.nju.edu.cn/falcons/ontologysearch/>

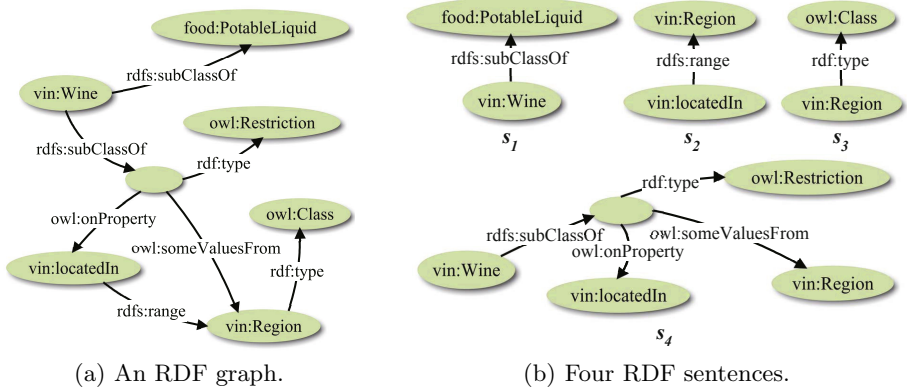


Fig. 1. An RDF graph and the corresponding RDF sentences

Fig. 1(b). Instead, we impose constraints on the total number of RDF triples involved, i.e. $|\cup S| \leq k$, given k a positive integer. In addition, we are also aware of that from the point of view of presentation, RDF triples may not appear equally long because literals and labels of terms may be of different lexical lengths. However, this issue will not be addressed in this work.

3 Salience Measurement

In this section, firstly we review centrality-based salience measurement for summarization tasks. Then, we present a novel bipartite graph representation of vocabulary description, and study centrality within this graph via simulating a random surfer’s behavior. Finally, we extend the solution with an investigation into the patterns of RDF sentences.

3.1 Centrality-Based Salience

Centrality indicates the importance of a node within a graph. To apply centrality-based methods to measure salience of data units for a summarization task, a graph needs to be constructed where nodes correspond to data units and edges represent their similarity or relatedness. This idea, after being successfully tested on text summarization [5], has also been exploited for summarizing vocabularies [16]. To be specific, given a vocabulary v , each RDF sentence in $Sents(v)$ is mapped to a node within a so-called “RDF sentence graph”, and two RDF sentences are connected by an edge if they contain common RDF resources, in particular common URIs in this context. A weight is assigned to each edge based on the syntactic roles of these common URIs, i.e. as subject, predicate or object of the RDF triples involved. Then, many different kinds of centrality such as degree, betweenness and eigenvector (e.g. PageRank) could be computed.

However, we observe that the weakness of this approach is threefold. (1) Since the construction of this graph employs merely the RDF syntax, e.g. determining edges and their weights only based on the syntactic roles of URIs, the resulting relationships captured by the graph seem too general and ambiguous. (2) Whereas terms are the first-class citizens of a vocabulary, they are not directly modeled in this graph, but only contribute to the weights of the edges, thereby failing to make the principle behind the model explicit. (3) Whereas the approach focuses on the relationships between RDF sentences, little attention is paid to their internal structure, which may have an effect on the salience measurement.

To remedy the first and second flaws, in the following we will study centrality within a bipartite graph that models a specific kind of relationship between RDF sentence and term. After that, we will investigate the patterns of RDF sentences for refining the model and remedying the third flaw.

3.2 BipRank: Centrality within a Bipartite Graph

To measure the salience of an RDF sentence $s \in \text{Sents}(v)$, we choose to look at how likely it is that s will be visited when v is under investigation. To this end, we simulate a hypothetical surfer’s behavior of exploring v ’s description, i.e. $\text{Sents}(v)$, which has two types. On the one hand, when the surfer has just visited an RDF sentence (e.g. s_1 in Fig. 1(b)), she might focus on a term described there (e.g. `vin:Wine`). Then, she intends to explore its description (s_1 and s_4). To start with, she chooses one (e.g. s_4) from these RDF sentences. As a result, the surfer turns from one RDF sentence (s_1) to another (s_4). On the other hand, the surfer might directly choose another RDF sentence from $\text{Sents}(v)$ at random (e.g. s_2) for further exploration (after visiting s_1).

Such characterization of a surfer’s behavior is quite similar to the one under the well-known PageRank algorithm. However, since two kinds of units are discussed here, it inspires us to consider a bipartite graph representation.

Before that, we notice that in the first type of behavior, not all the terms occurring in an RDF sentence will equally have a chance of being focused on. For instance, given s_1 in Fig. 1(b), it is most unlikely that the description of `rdfs:subClassOf` will be looked up. That is, we should not assume a homogeneous relationship between RDF sentence and term. In fact, here `rdfs:subClassOf` is only used but not essentially described. To convey this message, we differentiate between intensional and extensional descriptions. An RDF sentence s is an *extensional* description of a term u if there is an RDF triple $t \in s$ satisfying that its predicate is `rdf:type` and its object is u , or its predicate is u . In other cases of occurrence, u is *intensionally* described in s . For instance, s_4 in Fig. 1(b) is an intensional description of `vin:Wine`, `vin:locatedIn` and `vin:Region`, but is an extensional description of `rdfs:subClassOf`, `owl:Property`, `owl:someValuesFrom`, `rdf:type` and `owl:Restriction`. We only consider intensional description when characterizing a surfer’s behavior.

Then, we propose to represent a vocabulary v ’s description as a bipartite graph called *sentence-term graph*, denoted by $\text{STG}(v)$, whose nodes comprise $\text{Sents}(v)$ and a set of terms T having some $s \in \text{Sents}(v)$ as an intensional description,

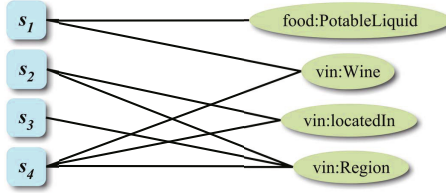


Fig. 2. A sentence-term graph

and whose edges E connect terms with their intensional descriptions. For our toy vocabulary vin , Fig. 2 illustrates $STG(vin)$ corresponding to Fig. 1(b).

Operating on this bipartite graph, we present a stochastic analysis of a surfer’s behavior as characterized previously, called *BipRank*. Let d be the probability of choosing the first type of behavior and thus $1 - d$ the probability of choosing the second type, since we assume only two types. For the first type, for each edge $\langle s, u \rangle \in E$ connecting an RDF sentence s with a term u , let $p(u|s)$ be the probability of focusing on u at s , and let $p(s|u)$ be the probability of exploring s at u . They must satisfy the following normalization constraints:

$$\begin{aligned} \forall s \in \text{Sents}(v), \quad \sum_{\langle s, u \rangle \in E} p(u|s) &= 1, \text{ and} \\ \forall u \in T, \quad \sum_{\langle s, u \rangle \in E} p(s|u) &= 1. \end{aligned} \tag{1}$$

For the second type, we simply assume the surfer chooses with uniform probability which RDF sentence to explore. Then, let $BR_r(v, s)$ be the probability that the surfer visits $s \in \text{Sents}(v)$ at step r , which satisfies:

$$\sum_{s \in \text{Sents}(v)} BR_r(v, s) = 1. \tag{2}$$

By taking all the possibilities of the surfer’s behavior into account, $BR_{r+1}(v, s)$ is iteratively updated as follows:

$$\begin{aligned} BR_{r+1}(v, s) &= d \sum_{\langle s, u \rangle \in E} p(s|u) \sum_{\langle s', u \rangle \in E} p(u|s') BR_r(v, s') \\ &+ (1 - d) \sum_{s' \in \text{Sents}(v)} \frac{1}{|\text{Sents}(v)|} BR_r(v, s'). \end{aligned} \tag{3}$$

In fact, *BipRank* can be regarded as a specific case of a general probabilistic framework for random walks [3], in which it has been proved that $BR_r(v, s)$ will converge to a constant $BR^*(v, s)$ that does not depend on any initial values of BR if $1 - d \neq 0$. Similar to PageRank, BR^* also characterizes a certain kind

of centrality, and in this context, it indicates the salience of RDF sentence in a vocabulary description.

Before applying BipRank, some parameters require specification. Whereas d is usually tuned based on experience or experimental results, in this work we simply assume $p(u|s)$ to follow a uniform distribution. We could make a similar assumption on $p(s|u)$, and we refer to the resulting salience measure as BipRank-U. However, in the next, we will discuss other ways of estimating $p(s|u)$.

3.3 Patterns of RDF Sentences

So far, BipRank only models the relationship between term and intensional description (i.e. RDF sentence). We argue that the schema underlying an intensional description should also contribute to its salience, because different surfers may favor different kinds of descriptions, e.g. subclass/superclass, domain/range or label/comment. That is, $p(s|u)$ may not follow a uniform distribution, but depend on from which aspect s describes u , i.e. the pattern of s .

To represent the pattern of an RDF sentence, we substitute placeholders for the terms intensionally described. Blank nodes are suitable placeholders since they are treated as existential variables in RDF semantics. More formally, the *pattern* of an RDF sentence s , denoted by $\text{Pattern}(s)$, is an RDF graph derived from s by substituting a fresh blank node for each occurrence of a literal or URI that is not a term extensionally described. For instance, Fig. 3 illustrates the patterns of the four RDF sentences in Fig. 1(b).

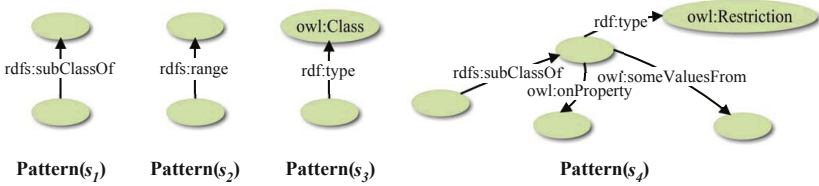


Fig. 3. Four patterns of RDF sentences

Rather than relying on surfer-specific preference for pattern, we intend to develop a generic method for estimating $p(s|u)$ that can serve a broad range of needs. Therefore, we refer to the statistical properties of patterns for clues.

Firstly, given a vocabulary v , if a pattern is observed in many RDF sentences in $\text{Sents}(v)$, it is believed to capture a key structural characteristic of v , and thus all the RDF sentences in $\text{Sents}(v)$ that have this pattern are supposed to have a high centrality. More formally, we have:

$$p(s|u) \propto |\{s' \in \text{Sents}(v) \mid \text{Pattern}(s) \preceq \text{Pattern}(s')\}|, \tag{4}$$

where $\text{Pattern}(s) \preceq \text{Pattern}(s')$ means that $\text{Pattern}(s)$ is isomorphic to a subgraph of $\text{Pattern}(s')$. That is, not only pattern equivalence (which amounts to equivalence between RDF graphs) but also pattern inclusion (i.e. subgraph isomorphism) are considered in our analysis. This extension of BipRank employs the *frequency* of each pattern within a vocabulary, and we call it BipRank-F.

Secondly, given a vocabulary repository V , if a pattern is observed in RDF sentences distributed in many vocabularies, i.e. being favored by many vocabulary publishers, we consider that it may also be favored by many surfers, and thus all the RDF sentences having this pattern should have a high centrality. Then we have:

$$p(s|u) \propto |\{v \in V \mid \exists s' \in \text{Sents}(v) : \text{Pattern}(s) \preceq \text{Pattern}(s')\}|. \quad (5)$$

This extension of BipRank leverages the *popularity* of each pattern, and we call it BipRank-P.

To sum up, we have presented three implementations of our BipRank model, functioning as different salience measures. In the next, we will apply BipRank to summarizing vocabularies, and also compare these measures empirically.

4 Vocabulary Summarization

This section addresses the problem of vocabulary summarization. We firstly discuss how to characterize a good summary, then we present our metrics, and finally discuss implementation.

4.1 Goodness of a Vocabulary Summary

As introduced in Sect. 2, we have defined a summary of a vocabulary v as a subset of $\text{Sents}(v)$ with an upper limit k on the total number of RDF triples involved. There could be many summary candidates under this constraint, and the criterion for accessing their “goodness” really depends on the application. In this work, we aim at supporting vocabulary search. A good summary in this context (a.k.a. snippet) is expected to facilitate effective and efficient human judgment on the relevance of the corresponding search result (i.e. a vocabulary) to the information need expressed in the query. Inspired by the studies of text summarization [12], we propose to measure three dimensions of goodness:

Salience. Summarization should extract the most salient units that capture the main theme of the original data. In our problem, since vocabulary description is given in a structured manner, it boils down to identifying those RDF sentences that have a high centrality within graphical representation.

Query relevance. Query-oriented summaries have been widely used in information retrieval. Similarly, for supporting vocabulary search, the selection of RDF sentences should be biased towards the query for effectively reflecting the relevance of the underlying vocabulary to the information need.

Cohesion. Text summarization considers a summary more than a collection of separate sentences, but requiring exhibiting user-perceived unity grounded on lexical and grammatical relationships between sentences that hold the text together. Analogously, a good vocabulary summary should present interconnections between its constituent RDF sentences.

Since all these dimensions may or may not be optimized simultaneously, we formalize vocabulary summarization as a multi-objective optimization problem, and solve it by constructing a single aggregate objective function:

$$\text{Goodness}(v, S, Q) = (1 - \alpha - \beta) \text{Sal}(v, S) + \alpha \text{Rel}(S, Q) + \beta \text{Coh}(S), \quad (6)$$

where v is the vocabulary to be summarized, $S \subseteq \text{Sents}(v)$ a summary candidate, Q a set of query keywords, and $\alpha, \beta, \alpha + \beta \in [0, 1]$ are weighting coefficients. The three constituent objective functions will be discussed in the following.

4.2 Metrics

Saliency. In Sect. 3 we have described three variants of our BipRank model, any of which could be employed to measure the saliency of RDF sentences in a vocabulary description (i.e. BR^*). Accordingly, we have

$$\text{Sal}(v, S) = \sum_{s \in S} \text{BR}^*(v, s). \quad (7)$$

Query Relevance. Vocabulary description involves not only structure but also text. The lexical feature of a summary S can be characterized by a vector, denoted by $\text{KW}(S)$. Each dimension corresponds to a separate keyword, whose value is given by the total frequency of the keyword in the local name of each term intensionally described in S and the lexical form of each literal occurring in S . Analogously, a keyword query Q can also be represented by a keyword vector. Then, the relevance of S to Q is defined as their cosine similarity:

$$\text{Rel}(S, Q) = \text{Cosine}(\text{KW}(S), Q). \quad (8)$$

Cohesion. The cohesion of a summary is mainly embodied in the common topics shared by its units. In vocabulary summarization, these amount to the common terms intensionally described in the selected RDF sentences. Accordingly, given $\text{ITerms}(s)$ the set of terms intensionally described in s , we have

$$\text{Coh}(S) = \sum_{\substack{s_i, s_j \in S \\ s_i \neq s_j}} |\text{ITerms}(s_i) \cap \text{ITerms}(s_j)|. \quad (9)$$

4.3 Notes on Implementation

We would like to briefly discuss two trade-offs we make in the implementation.

Firstly, the optimization problem defined by (6) is at least NP-complete, because the salience part boils down to the 0-1 knapsack problem. Therefore, we employ a greedy strategy to find a locally optimal solution, which starts with an empty summary, and iteratively adds one RDF sentence that leads to the highest goodness until no more can be added without violating the length constraint.

Secondly, even though salience measurement can be carried out offline, we observe that subgraph isomorphism testing in (4) and (5) is NP-complete and needs to be performed many times. Hence we simplify this task by representing each pattern as the set of all the URIs occurring in it, and then testing subgraph isomorphism is approximately reduced to testing set inclusion.

5 Experiments

Our experiments were based on a real-world repository indexing 2,029 vocabularies crawled by the Falcons search engine⁴ from February to September 2010, collectively containing 381,317 terms and 2,328,091 RDF sentences derived from 2,478,085 RDF triples.

5.1 Evaluation of Salience Measures

In the first experiment, we compare the salience of RDF sentences given by our BipRank measures and a baseline measure with handcrafted gold standards.

We identified 312 moderate-sized vocabularies from our repository, each containing 20–50 RDF sentences describing 5–20 terms, which are neither too small to be significant for a summarization task nor too large for manual investigation. Then, nine of them were randomly selected as test cases, as shown in Table 1. We invited six human experts to independently rate each RDF sentence in each vocabulary with a 10-point scale indicating its salience to the main theme of the vocabulary. We observed that the Pearson product-moment correlation coefficients (ρ) between their ratings ranged from 0.29 to 0.82, and averaged 0.57, showing a strong (i.e. $|\rho| > 0.5$) agreement (i.e. $\rho > 0$) between their opinions.

In the literature, only [16] has proposed several salience measures for RDF sentence as described in Sect. 3.1, from which the one based on the PageRank centrality (denoted by C_p) was implemented as a baseline, since it was empirically among the best according to [16]. In the implementation of the three variants of our BipRank model (i.e. BipRank-U, BipRank-F and BipRank-P), d in (3) was set to 0.85, a widely used empirical value in related research, and all the iterative computation would stop after 20 steps.

In each test case, since we received a gold standard from each expert, we compared computed salience with each of them and then took the average ρ value. The results are presented in Table 1, where the highest value(s) in each case are highlighted. We observe that, on the one hand, our basic BipRank-U measure consistently and considerably outperformed C_p in all the cases, leading

⁴ <http://ws.nju.edu.cn/falcons/>

Table 1. ρ between computed salience of RDF sentences and gold standards

	C_p	BipRank		
		-U	-F	-P
v_1 http://data.ordnancesurvey.co.uk/ontology/spatialrelations/	-0.07	0.20	-0.58	-0.19
v_2 http://ebiquity.umbc.edu/ontology/news.owl	0.49	0.61	0.47	0.56
v_3 http://metadata.net/WildNET/Geography.owl	-0.02	0.71	0.63	0.71
v_4 http://sw.deri.org/2005/08/conf/cfp.owl	-0.13	0.66	0.46	0.71
v_5 http://www.csc.ncsu.edu/faculty/mpsingh/books/SOC/1st/description/life.owl	-0.28	0.80	0.62	0.86
v_6 http://www.cse.sc.edu/research/cit/projects/DAML/Guo.daml	-0.18	0.62	0.62	0.61
v_7 http://www.daml.org/researchers-ont	-0.23	0.37	0.30	0.35
v_8 http://www.ling.helsinki.fi/kit/2004k/ctl310semw/GATE/Exporter.daml	0.05	0.90	0.90	0.90
v_9 http://www.mindswap.org/2004/multipleOnt/FactoredOntologies/FactoredPeoplePets/FactoredVehicle.owl	-0.15	0.75	0.55	0.76
Average	-0.06	0.62	0.44	0.59

by 0.12–1.08 or averaging 0.68. That is, our combination of sentence-term graph and BipRank is a much more appropriate model than RDF sentence graph and PageRank proposed in [16] for characterizing the salience of RDF sentence. On the other hand, when further employing pattern of RDF sentence, the results varied from case to case. Generally, we didn’t obtain better results with BipRank-F. One illustrative case is v_4 , where a pattern about “term status” is used many times, which nevertheless was not deemed that important by the experts. By comparison, BipRank-P exactly showed its usefulness in this case, which assigned relatively low salience to the RDF sentences having this unpopular pattern. However, both BipRank-F and BipRank-P met opposite ratings in v_1 , where the experts preferred the RDF sentences declaring reflexive/transitive/symmetric properties to those more general declarations of object properties, despite the relatively infrequent and unpopular patterns of the former. This finding inspires us to study a trade-off between “informativeness” and frequency/popularity of pattern in future work. To sum up, our BipRank measure has exhibited a definite advantage over the state of the art, whereas the employment of pattern of RDF sentence still needs further investigation.

5.2 Evaluation of Vocabulary Summaries

In the second experiment, we generate summaries by using several variants of our approach and a baseline approach, and compare their usefulness in search applications based on human ratings.

We developed a keyword search interface based on our vocabulary repository (excluding 17 huge vocabularies containing more than 10,000 RDF sentences), which retrieves all the vocabularies in the repository whose contents can match all the query keywords, and returns one of them that is selected at random. For the returned vocabulary, each of the approach settings to be evaluated was employed to generate a summary under the same constraint of $k = 5$. Accordingly, tests on different values of k were outside the scope of this work. These summaries were listed in random order, and their underlying approach settings were not given to users. Each summary was presented as a merge of its constituent RDF sentences — amounting to a single RDF graph, and was visualized as a node-link diagram by using Graphviz⁵.

We invited 18 human experts (P_1 – P_{18}) to this experiment, each carrying out at least 10 searches with different keyword queries. For each search, the expert was asked to think about an arbitrary domain (e.g. sports or news), describe it by using some keywords which compose a keyword query, and judge — based on the expert’s own opinion and each of the presented summaries — how well it is characterized by the returned vocabulary (if any). However, we were not interested in and thus did not ask for the judgment itself. Instead, the expert needed to rate each summary with a 10-point scale indicating its usefulness in assisting in making the judgment.

In the literature, we found that [16] and [10] are comparable to our approach since they also treat the RDF sentence as the unit of vocabulary summary. In [16], a generic (i.e. not query-biased) approach to vocabulary summarization (denoted by Generic) is proposed, which firstly ranks RDF sentences by C_p and then performs a diversity-oriented re-ranking. Whereas we implemented Generic as a baseline approach, we did not choose the query-biased approach presented in [10] because we were not sure how to adapt it appropriately for our problem definition (i.e. ranking and selecting RDF sentences). In fact, it would firstly produce a clustering of RDF sentences, and then compute a ranking of RDF sentences within each cluster. However, a reasonable way to select RDF sentences that collectively contain not more than k RDF triples from such kind of result is not explicit. We also implemented three variants of our approach by setting the parameters in (6) to different values: (1) QR that considers only query relevance ($\alpha = 1, \beta = 0$), (2) QR+S that considers query relevance and salience ($\alpha = 0.5, \beta = 0$, using BipRank-P), and (3) QR+C that considers query relevance and cohesion ($\alpha = \beta = 0.5$).

We received 190 valid searches, each returning a vocabulary of which all the summaries generated by each approach were associated with human ratings. Table 2 summarizes these ratings given by each expert to each approach, averaged over all the searches performed by the expert. Firstly, between Generic and QR, all the 18 experts except P_5 (94%) gave higher ratings to the summaries generated by our approach with only the query relevance metric enabled than to the baseline approach, leading by 2.37 on average on a 10-point scale, which suggested that our query-biased vocabulary summarization considerably

⁵ <http://www.graphviz.org/>

outperforms a state-of-the-art generic approach. Secondly, between QR and QR+S, 9 experts (50%) believed that simultaneously using our salience metric is superior to only using query relevance, whereas only 6 (33%) believed the opposite. Finally, between QR and QR+C, the superiority of incorporating our cohesion metric was largely supported by 14 experts (78%). Overall, QR+C was also the best-performing approach in this evaluation. To sum up, when our query-biased summaries have proven to be more useful in vocabulary search than the baseline, they could be further enhanced by our salience and cohesion metrics. However, more parameter combinations deserve to be tested in future work.

Table 2. Usefulness of computed summaries in search rated by each expert

	Generic	QR	QR+S	QR+C		Generic	QR	QR+S	QR+C
P_1	1.80	5.90	6.70	6.20	P_{10}	5.40	6.70	7.10	6.90
P_2	1.58	6.89	7.00	7.47	P_{11}	2.09	4.82	4.82	4.36
P_3	3.80	9.10	9.10	9.00	P_{12}	4.20	5.50	5.70	5.70
P_4	5.30	7.50	7.80	7.70	P_{13}	4.10	6.20	6.00	5.70
P_5	7.10	5.70	6.10	7.80	P_{14}	2.20	7.40	7.40	7.60
P_6	5.80	6.90	6.30	8.80	P_{15}	3.00	6.30	5.90	7.30
P_7	3.60	6.00	6.50	6.30	P_{16}	5.40	6.60	6.00	7.10
P_8	5.45	5.91	5.55	6.09	P_{17}	2.08	5.00	5.50	5.92
P_9	1.36	4.09	3.82	3.82	P_{18}	4.91	5.27	5.55	5.55
					Average	3.84	6.21	6.27	6.63

Figure 4 summarizes all the human ratings in a different way, to show the relationship between the usefulness of a summary and the number of keywords contained in a query. Whereas the three variants of our approach performed robustly in different settings, the rating received by the baseline approach consistently decreased when increasing the number of query keywords. One explanation could be that a query containing more keywords usually indicates a more specific information need, thereby expecting more indications of query relevance from the summaries, which failed to be achieved by a generic approach. When our approach under all the three parameter combinations largely outperformed the baseline approach, QR+C seemed to be the best among them, in particular for queries comprising single keyword. That is, when users aimed at exploring an unfamiliar domain without specific needs, they were more biased towards starting from a cohering portion of a vocabulary than from those separate — even if salient — particles.

5.3 Performance Testing

When being used in vocabulary search, the efficiency of a summarization approach becomes a critical issue since a high latency would badly hurt user experience. In this third experiment, we test the performance of our approach in real-world settings.

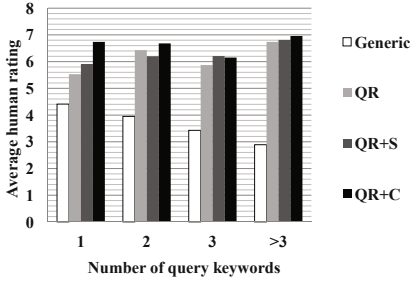


Fig. 4. Human-rated usefulness of computed summaries in search with queries comprising different numbers of keywords

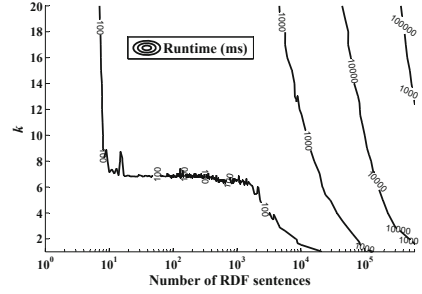


Fig. 5. A contour map view of the runtime of our approach applied to vocabularies containing different numbers of RDF sentences (on a log scale) under different k values (on a linear scale)

Our implementation was written in Java and ran on a 64-bit server with two 4-core Xeon 2.4G and 4GB memory for JVM. In particular, at each iteration of our greedy implementation as described in Sect. 4.3, we employed 20 threads to concurrently look for a locally optimal solution. On each vocabulary in our repository, we run our approach 5 times (after a warm-up run) under each $k \in [1, 20]$, each time with a query comprising up to 10 keywords picked at random from the vocabulary. The contour map shown in Fig. 5 presents the average runtime of one single execution of our approach under different k values and different numbers of RDF sentences contained in a vocabulary. Under $k = 6$ — a practical setting adopted by our vocabulary search engine, our approach could scale to 1,743 RDF sentences (covering 98.3% of all the vocabularies in our repository) within 100 milliseconds, to 19,420 (covering 99.3%) within 1 second and to 608,235 (covering all) within 45 seconds. That is, our approach could deal with almost all the real-world vocabularies in our repository in real time, except for only a few very large ones, which require developing more efficient solutions in the future. From another point of view, given 206 RDF sentences — covering 80% of all our vocabularies, our approach costed 142 milliseconds under $k = 10$ and 272 milliseconds under $k = 20$. That is, even for larger k values, our approach still scaled well to most real-world cases. In fact, the runtime basically increased linearly when increasing k or increasing the number of RDF sentences.

6 Related Work

6.1 Vocabulary Summarization: Extracting RDF Sentences

Vocabulary (or ontology) summarization was firstly conceived as the problem of ranking RDF sentences in our previous work [16]. We defined two kinds of weighted links between RDF sentences based on the RDF resources identified by URIs they have in common and their syntactic roles in RDF triple.

This leads to an RDF sentence graph, on which various centrality measures such as degree, betweenness and PageRank were leveraged to induce a ranking of RDF sentences. We further developed a re-ranking strategy for generating more coherent and diverse summaries, also operating on the two kinds of links.

After that, [10] extended [16] by deriving a topic graph from an RDF sentence graph contracted on similar nodes. The similarity between RDF sentences is measured by the common textual information they have in common. Then, query-relevant topics are ranked according to their in-degree centralities within the topic graph, and top-ranked topics are selected as a summary. In each topic, only top-ranked RDF sentences are retained, and the ranking is based on the in-degree centrality within the RDF sentence graph.

This paper follows this line of approach. However, we extend the state of the art in several directions. Firstly, we propose a new graphical representation for characterizing the relationship between RDF sentences via terms, namely the notion of sentence-term graph, which is a bipartite graph grounded on not the ambiguous syntactic roles of terms but the more principled intensional description relationship between RDF sentence and term. Secondly, besides measuring the centralities of RDF sentences by using a variant of the random surfer model, we also discuss the impact of their patterns from two angles: frequency and popularity. Thirdly, we are the first to formalize vocabulary summarization (i.e. ranking and selecting RDF sentences) as a multi-objective optimization problem, in which salience, query relevance and cohesion are investigated.

6.2 Vocabulary Summarization: Identifying Key Terms

Another form of vocabulary summary is a set of key terms selected, which has been adopted by several search engines including Swoogle and Watson. Approaches thus far operate on various kinds of graphical representation of vocabulary description such as class hierarchy [11], class-property graph [15] and term dependence graph [17], and ranking criteria range from degree-based centrality measures such as density [11] and betweenness [14] to sophisticated measures based on different variants of the random surfer model [15,17]. Some other approaches such as [4,6] also consider the popularity of terms on the Web.

Choosing the term or the RDF sentence as the unit of summary are two different types of extractive vocabulary summarization. In fact, an empirical analysis [8] reveals that the two sides may generate quite close results in terms of the terms involved. We choose the latter because a subset of RDF sentences could slice off not only terms but also their semantic relationships, which could be leveraged by users to make more accurate relevance judgments in search applications.

6.3 Keyword Search on Graphs

Keyword search on graphs firstly attracted interests from the database community [1], and later was studied on RDF graphs [7]. The basic idea is to map query keywords to graph elements (i.e. nodes and edges) and then search for and rank

subgraphs that can connect these elements. When being applied to vocabularies in search applications, such a resulting subgraph could also be regarded as a vocabulary summary.

However, these approaches are designed for serving query answering, where a resulting subgraph is supposed to carry an answer to the question underlying a keyword query, which is different from the goal pursued in vocabulary search, namely to support efficient human inspection and relevance judgment. Accordingly, vocabulary summarization usually specifies a length constraint, whereas keyword search on graphs mainly requires every resulting subgraph to be connected.

7 Conclusions and Future Work

We have proposed the notion of sentence-term graph to characterize the intensional description relationship between RDF sentence and term in a vocabulary. On this graph, a variant of the random surfer model called BipRank is designed for measuring the centrality of RDF sentence as a proxy for salience, which is much closer to the gold standards given by human experts than a state-of-the-art approach. Further, the effect of pattern of RDF sentence on this model has been empirically analyzed, which is positive in some cases but generally still has room for being exploited in more appropriate ways. With BipRank and other two metrics for query relevance and cohesion, we have developed a new solution to vocabulary summarization, which is formalized as a multi-objective optimization problem. The results of a search-based evaluation show that the summaries generated by our approach are more useful than those computed by a baseline approach, and all the three metrics considered are effective. We have also demonstrated the scalability of our approach in real-world settings.

The evaluation results have suggested other factors such as informativeness that could be employed to weight patterns of RDF sentences in BipRank, which will be our primary work in the future. Besides, it would be interesting to investigate whether BipRank can be adapted for ranking not only RDF sentences but also terms. As to vocabulary summarization, we are also interested in extending this problem to the more generalized case of multiple vocabularies.

Acknowledgments. This work was supported in part by the NSFC under Grant 60973024 and 61100040, and in part by ZTE Corp. (R&Dcon1105160003). We thank Dr. Xiang Zhang for his invaluable advice on this work, thank Saisai Gong for his time and effort in supporting the experiments, and thank all the students that participated in the experiments.

References

1. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword Searching and Browsing in Databases Using BANKS. In: 18th International Conference on Data Engineering, pp. 431–440. IEEE Computer Society, Washington, DC (2002)

2. Cheng, G., Ge, W., Qu, Y.: Generating Summaries for Ontology Search. In: 20th International Conference Companion on World Wide Web, pp. 27–28. ACM, New York (2011)
3. Diligenti, M., Gori, M., Maggini, M.: A Unified Probabilistic Framework for Web Page Scoring Systems. *IEEE Trans. Knowl. Data Eng.* 16(1), 4–16 (2004)
4. Ding, L., Pan, R., Finin, T., 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)
5. Erkan, G., Radev, D.R.: LexRank: Graph-based Centrality as Salience in Text Summarization. *J. Artif. Intell. Res.* 22, 457–479 (2004)
6. 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)
7. Ladwig, G., Tran, T.: Combining Query Translation with Query Answering for Efficient Keyword Search. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) *ESWC 2010, Part II. LNCS*, vol. 6089, pp. 288–303. Springer, Heidelberg (2010)
8. Li, N., Motta, E.: Evaluations of User-Driven Ontology Summarization. In: Cimi-ano, P., Pinto, H.S. (eds.) *EKAW 2010. LNCS*, vol. 6317, pp. 544–553. Springer, Heidelberg (2010)
9. Li, N., Motta, E., d’Aquin, M.: Ontology Summarization: An Analysis and An Evaluation. In: *Proceedings of the International Workshop on Evaluation of Semantic Technologies, CEUR* (2010)
10. Penin, T., Wang, H., Tran, T., Yu, Y.: Snippet Generation for Semantic Web Search Engines. In: Domingue, J., Anutariya, C. (eds.) *ASWC 2008. LNCS*, vol. 5367, pp. 493–507. Springer, Heidelberg (2008)
11. Peroni, S., Motta, E., d’Aquin, M.: Identifying Key Concepts in an Ontology, through the Integration of Cognitive Principles with Statistical and Topological Measures. In: Domingue, J., Anutariya, C. (eds.) *ASWC 2008. LNCS*, vol. 5367, pp. 242–256. Springer, Heidelberg (2008)
12. Spärck Jones, K.: Automatic Summarising: The State of the Art. *Inf. Process. Manag.* 43(6), 1449–1481 (2007)
13. Tummarello, G., Morbidoni, C., Bachmann-Gmür, R., Erling, O.: RDFSyc: Efficient Remote Synchronization of RDF Models. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007. LNCS*, vol. 4825, pp. 537–551. Springer, Heidelberg (2007)
14. Tzitzikas, Y., Kotzinos, D., Theoharis, Y.: On Ranking RDF Schema Elements (and its Application in Visualization). *J. Univers. Comput. Sci.* 13(12), 1854–1880 (2007)
15. Wu, G., Li, J., Feng, L., Wang, K.: Identifying Potentially Important Concepts and Relations in an Ontology. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008. LNCS*, vol. 5318, pp. 33–49. Springer, Heidelberg (2008)
16. Zhang, X., Cheng, G., Qu, Y.: Ontology Summarization Based on RDF Sentence Graph. In: 16th International Conference on World Wide Web, pp. 707–716. ACM, New York (2007)
17. Zhang, X., Li, H., Qu, Y.: Finding Important Vocabulary Within Ontology. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) *ASWC 2006. LNCS*, vol. 4185, pp. 106–112. Springer, Heidelberg (2006)

Operational Semantics for SPARQL Update

Ross Horne, Vladimiro Sassone, and Nicholas Gibbins

Electronics and Computer Science
University of Southampton, UK
{rjh06r,vs,nmg}@ecs.soton.ac.uk

Abstract. Concurrent fine grained updates are essential for using RDF stores in dynamic modern Web applications, where users increasingly contribute content as often as they read content. SPARQL Update is a language proposed by the W3C for fine grained updates for RDF stores. In this work we propose an operational semantics for an update language for RDF, which models core features of SPARQL Update. Firstly, an abstract syntax for RDF and updates is presented. Secondly, the operational semantics is defined using relations over the abstract syntax. The operational semantics specifies all possible operational behaviours of updates in the presence of an RDF store. The specification is useful as a common reference for compiler engineers and as a foundation for the static analysis of updates.

1 Introduction

An open problem is to provide an operational semantics for the W3C SPARQL Update working draft [9]. SPARQL Update is a development of an earlier proposal from Hewlett-Packard Labs [18]. The language is introduced to extend the SPARQL Query language [17] to enable fine grained updates over an RDF store.

The recommended semantics for SPARQL Query are influenced by the work of Pérez et al. [15], which provides a set-based denotational semantics for queries. In contrast, the semantics presented here for updates are operational in nature. The difference between a denotational semantics and an operational semantics is that the former builds an external model (typically a static set in which behaviours may exist), whereas the later is defined directly over an abstract syntax for the language.

There are several advantages of operational semantics. An operational semantics works like an interpreter, so is at an appropriate level for compiler engineering. Operational semantics is also suited to ad-hoc features which appear in real programming languages, which SPARQL Update intends to be. Furthermore, operational semantics are suited to specifying the complex long term behaviour of systems, including concurrency as required by servers. Denotational semantics for both application driven ad-hoc features and long term behaviour are notoriously difficult [1]. Thus, operational semantics can easily and insightfully be adapted to queries [12]; but denotational semantics do not extend easily to updates, since non-standard mathematics would be required.

Consider an analogy. All readers are familiar with the concept of a regular expression or use tools which involve regular expressions. For instance, the replace tool in your text editor is appropriate for every day updates in text documents. This update

language extends the power of regular expressions generalised appropriately to RDF triples (instead of characters), with quantifiers to access URIs and literals in triples. For the sake of clarity, here a core update language, rather than full SPARQL Update, is presented where only the default RDF graph is updated. The model can be extended to handle named graphs [6]. Also, the model can accommodate updates with respect to entailments, such as those defined in RDFS [5].

In Section 2 further motivation is provided to emphasise the importance of an operational semantics for SPARQL Update. In Section 3 a syntax for RDF Data and Updates is established. In Section 4 an equivalence is provided over RDF Data. This equivalence defines when two pieces of RDF Data have the same meaning. In Section 5 the behaviour of Updates is specified using a deductive system which derives relations over the syntax. The relations indicate how some RDF Data is transformed by a Update into some other RDF Data. Examples of each feature of Updates are provided along with the rules of the operational semantics.

2 Background and Motivation

Before defining the operational semantics, some motivation is provided in this section. The tradition of using operational semantics to specify programming languages is discussed, highlighting benefits offered to Web standards. A short sketch of the relationship between this work and the draft specification is also provided, including a preview of the abstract syntax.

2.1 A Case for an Operational Semantics for SPARQL Update

A structural approach to operational semantics was first introduced in a seminal note by Plotkin [16]. At the time, the behaviour of languages tended to be specified using a reference compiler. The correctness of an implementation of a language would be verified by checking that its behaviour matched the behaviour of the reference compiler. Plotkin's work introduced a methodology for precisely specifying the operational behaviour of languages directly over an abstract syntax.

Web standards, such as SPARQL Update, require clear specifications of their behaviour. Structural operational semantics are designed precisely for this scenario. A clear operational semantics can be concisely communicated in a document, which can be used as a reference to ensure that all implementations of a standard have a common operational behaviour. For this reason, an operational semantics for SPARQL Update is a valuable contribution to the current standardisation process at the W3C [9].

An operational semantics for a language has further advantages. It allows clarity and methodology when design decisions are considered. Furthermore, operational semantics can be used as the basis for powerful techniques and tools for the language, such as a static type checker or algebra for composition and optimisation of updates [12, 11]. Such tools cannot be confidently developed without an operational semantics. Furthermore, the distinction between static and dynamic types is not evident until updates are considered.

2.2 A Comparison of SPARQL Update to This Work

The intention of the update language introduced is to model the core of SPARQL Update. A basic comparison between this update language and SPARQL Update is provided here.

The following example is adapted from the current working draft [9]. The update deletes zero or more RDF triples where the literal "Bill" appears and inserts a triple where "William" appears. The update can only occur if the subject of the triple is of type person.

Concrete Update:

```
DELETE { ?person foaf:givenName "Bill" }
INSERT { ?person foaf:givenName "William" }
WHERE { ?person rdf:type foaf:Person }
```

The above update can be expressed in an abstract syntax as follows.

Abstract Update:

```
DO SELECT :person {
  DELETE { :person foaf:givenName "Bill" }
  INSERT { :person foaf:givenName "William" }
  WHERE { :person rdf:type foaf:Person }
}
```

The abstract syntax above is more explicit than the concrete syntax. The select quantifier explicitly indicates the scope of the bound variable. Also, the abstract syntax of the update language explicitly indicates that the update may be applied zero or more times. This makes the definition of the operational semantics cleaner.

3 A Concise Abstract Syntax

This section presents an abstract syntax used to define an operational semantics for Updates. The abstract syntax is intended for the purpose of compiler engineering (as opposed to defining an exchange format). Three generators are sufficient to specify this abstract syntax: one for RDF Data; one for constraints; and a third for Updates.

3.1 Syntactic Conventions

The following namespace prefix bindings are used throughout this document.

```
dc: http://purl.org/dc/terms/
dc11: http://purl.org/dc/elements/1.1/
foaf: http://xmlns.com/foaf/0.1/
eg: http://example.org/
```

Prefixes abbreviate URIs, for readability in examples. Curly brackets are used to resolve ambiguity in examples.

3.2 A Syntax for RDF Data

The following grammar presents an abstract syntax for RDF. Several concrete syntaxes have been proposed for RDF, such as Turtle and N3 for the purpose of tersely presenting RDF to humans [24]. In contrast, the following abstract syntax for RDF Data is designed for compiler engineering.

<i>object</i> ::= "literal" a literal <i>URI</i> a URI <i>?variable</i> a variable	<i>Data</i> ::= { } nothing { <i>URI URI object</i> } a triple <i>Data</i> , <i>Data</i> par <i>BNODE URI Data</i> blank node
---	--

Two forms of triple represent RDF triples, with either a URI or a literal as the object. A variable indicates an unknown literal (for pattern matching in queries). Nothing represents the absence of any RDF triples. The operator ‘par’ composes RDF Data, thus for instance two triples can be composed using par. The Blank node quantifier binds a name which can only be referred to locally (as opposed to a URI which is a global name). For elegance, the same syntax is used to name URIs and blank nodes; they are distinguished only by quantifiers, which give local names scope. Many examples of RDF Data are presented throughout this work.

3.3 A Syntax for Constraints

Constraints are defined fully in the SPARQL Query recommendation [17], hence only an outline grammar for constraints is provided here. The following is enough to suggest that constraints form a Boolean algebra with built in primitives. Constraints may contain variables and URIs.

<i>Constraint</i> ::= true false <i>Constraint</i> && <i>Constraint</i> <i>Constraint</i> <i>Constraint</i> ! <i>Constraint</i> regex(<i>?variable</i> , <i>RegularExpression</i>) ...	true false and or not regular expression etc.
---	---

A constraint is satisfied if and only if it evaluates to true. The evaluation of constraints is detailed in the SPARQL Query recommendation [17]. Examples of constraints include regular expressions parametrised on a variable and inequality tests on numbers.

3.4 A Syntax for Updates

The following grammar proposes an abstract syntax for updates. The language is inspired by core features of SPARQL Update. A successful update results in an atomic change to an RDF store. This abstract syntax allows constructs to be nested. Nesting is useful for expressiveness and optimisation purposes.

<i>Update ::= DELETE Data</i>	delete a term
<i>INSERT Data</i>	insert a term
<i>FILTER Constraint</i>	impose a constraint
<i>Update CHOOSE Update</i>	choose a branch
<i>Update JOIN Update</i>	synchronise updates
<i>SELECT URI Update</i>	select a URI
<i>SELECT ?variable Update</i>	select a literal
<i>DO Update</i>	iteratively apply an update

Delete removes the indicated RDF Data from the store. Insert introduces some RDF Data to the store. Filter imposes a constraint on an update. Choose offers the choice of either a left or right update. Join ensures that two updates happen in the same atomic update. Select parametrises an update on either a URI or a literal which is not known in advance. (Note that in this abstract syntax, URIs and literals are distinguished in Selects for clarity.) Iteration (DO) performs an update zero, one, two or more times, in the same atomic update. Without iteration an update is applied once.

Examples of each construct are provided along with the operational semantics for the construct in Section 5.

3.5 Abbreviations for Common Updates

A number of common updates can be defined using the basic updates above. The use of abbreviations avoids redundancy in the operational semantics. Identifying redundant operators is useful for compiler engineering, since the number of operators to implement directly is reduced.

An optional update gives the choice of performing an update or not performing an update. The optional update can be defined by a choice between an update and the true constraint which always holds, as follows. (This avoids a left outer join operator, which is used to provide the semantics of ‘optional’ in SPARQL Query [15].)

$$\text{OPTIONAL Update} \triangleq \text{Update CHOOSE FILTER true} \quad \text{optional update}$$

Successive select queries are can be combined. The combined variables are listed in a single select quantifier, as follows.

$$\text{SELECT ?variable}_0 \text{ ?variable}_1 \text{ Update} \triangleq \text{SELECT ?variable}_0 \text{ SELECT ?variable}_1 \text{ Update} \quad \text{multiple selects}$$

In this paper queries are encoded naïvely, using the keyword WHERE. The effect of a query can be achieved by joined insert and delete, as follows.

$$\text{WHERE Term} \triangleq \text{DELETE Term JOIN INSERT Term} \quad \text{queries}$$

The joined delete and insert has the effect of a querying for a term: The term deleted must exist for the delete to be applied, but the insert immediately replaces the deleted term in the same atomic step. Queries could alternatively be defined as primitives of the language, as in [12].

4 An Equivalence over RDF Terms

This section identifies equivalent syntax. A syntactic equivalence imposes less constraints on RDF than any requirement that collections of triples are sets. Instead, obviously equivalent syntax is considered to serve the same purpose, as defined by a structural congruence.

4.1 A Structural Congruence

A structural congruence, written = below, is a relation between RDF Terms. A congruence is an equivalence relation (reflexive, symmetric and transitive) which holds in all contexts. The structural congruence satisfies the following equations — associativity, unit and commutativity.

Associativity: $Data_0 , \{Data_1 , Data_2\} = \{Data_0 , Data_1\} , Data_2$

Unit: $Data , \{\} = Data$

Commutativity: $Data_0 , Data_1 = Data_1 , Data_0$

The structural congruence can be applied at any point, when evaluating the operational semantics in Section 5.

Example of Applying the Structural Congruence. The following RDF Data can be used interchangeably. If the RDF on the left appears in a rule in the next section, then it can be replaced by the RDF on the right.

<pre>{ eg:book1 eg:price "£10" } , { } , { { eg:book2 dc:title "Linked Data" } , { eg:book1 dc:title "Web of Data" } } }</pre>	<pre>{ { eg:book1 dc:title "Web of Data" } , { eg:book1 eg:price "£10" } } , { eg:book2 dc:title "Linked Data" }</pre>
--	--

Brackets are used similarly for Group Graph Patterns in SPARQL Query [17]. Associativity of par allows most brackets to be omitted for readability.

Alpha Conversion. The standard notion of alpha conversion can be applied to blank node quantifiers. Alpha conversion allows a bound name to be replaced by a fresh name, to avoid name clashes. For instance the following RDF Data is equivalent.

<pre>BNODE :a { { :a foaf:familyName "Carrol" } , { :a foaf:knows eg:Klyne } }</pre>	<pre>BNODE :b { { :b foaf:familyName "Carrol" } , { :b foaf:knows eg:Klyne } }</pre>
--	--

Alpha conversion captures the isomorphisms in the RDF specification [13].

5 A Commitment Relation for Updates

The commitment relation specifies atomic changes which can be made to an RDF store. Atomicity focuses on the local effect of an update. The RDF Data which is required to perform an atomic update is accounted for. An advantage of this approach is that the data indicated by a commitment relation can be locked to ensure that an update occurs atomically.

A commitment relation consists of the RDF Data before an update, an Update and the resulting RDF Data after the update. Thus commitment relations are relations of the following form.

Before: *Data* Update: *Update* After: *Data*

Commitment relations can also be derived from rules. The premises of a rule are one or more commitment relations and the conclusion is a single commitment relation. The conclusion holds only if all the premises hold. The axioms and rules which specify operational semantics for Updates are defined throughout this section.

5.1 The Delete Axiom

The Delete Axiom removes some RDF Data from the store. The committed RDF Data, *Data*, and committed delete update, *DELETE Data*, interact. After the interaction both the Data is removed from the store. This results in the empty process.

Before: *Data* Update: *DELETE Data* After: {}

Example of the Delete Axiom. The following triple can be removed by the following update due to the following commitment relation. This commitment relation is an instance of the Delete Axiom.

Before: { *eg:book1 dc:title "The Semantic Web"* }
 Update: *DELETE*{ *eg:book1 dc:title "The Semantic Web"* }
 After: {}

5.2 The Insert Axiom

The Insert Axiom adds some designated RDF Data to the store. The designated RDF Data is indicated by the *INSERT* keyword. The result of this update is to make the designated RDF Data available after the commitment.

Before: {} Update: *INSERT Data* After: *Data*

Example of the Insert Axiom. The two triples below can be inserted into anything (since nothing is required), due to the following commitment relation. This commitment relation is an instance of the Insert Axiom.

Before: {}
 Update: *INSERT*{ *eg:book1 dc:title "The Web of Linked Data"* }
 After: { *eg:book1 dc:title "The Web of Linked Data"* }

5.3 The Join Rule

The Delete Axiom and the Insert Axiom allow basic updates to take place where either the exact RDF Data to be deleted is known, or the exact RDF Data to be inserted is known, respectively. For more substantial updates, rules are required to build commitment relations. The first of these rules is the Join Rule.

The Join Rule ensures that two updates occur atomically, in the same commitment relation. If one update has one effect and another update has another effect, then the join of the updates is their combined effect. The rule ensures that both updates act simultaneously on separate RDF Data. Suppose that the following commitment relation holds.

Before: $Data_0$ Update: $Update_0$ After: $Data_2$

Also, suppose that the following commitment relation holds.

Before: $Data_1$ Update: $Update_1$ After: $Data_3$

The two commitment relations above can be combined to produce the following commitment relation.

Before: $Data_0$, $Data_1$ Update: $Update_0$ JOIN $Update_1$ After: $Data_2$, $Data_3$

Example of Joined Updates. The update below demonstrates two joined updates. The update combines the examples of Sec. 5.1 and Sec. 5.2 using the join rule. Thus the combined update removes a triple and adds a new triple atomically.

Before: { *eg:book3 dc:title "The Semantic Web"* }
 Update: DELETE { *eg:book3 dc:title "The Semantic Web"* }
 JOIN
 INSERT { *eg:book3 dc:title "The Web of Linked Data"* }
 After: { *eg:book3 dc:title "The Web of Linked Data"* }

Notice that the rules ensure that join is commutative (this is not sequential composition). Also, from here onwards, the join keyword is omitted from examples. This makes examples more readable and closer to the SPARQL Update syntax.

5.4 The Select Literal Rule and Select URI Rule

The Select Literal Rule is parametrised on a variable. The variable is bound to the update indicated (so cannot be referred to from outside the select). The Select Rule allows any literal which enables a commitment to be substituted for the variable. The result of the commitment with the variable substituted for a literal, becomes the result of the commitment with the variable bound by a Select. Note that substitution is indicated by square brackets where the literal on the left replaces the variable on the right. Suppose that the following commitment relation holds.

Before: $Data_0$ Update: $Update["literal"/?variable]$ After: $Data_1$

Given the commitment relation above the following commitment relation holds.

Before: $Data_0$ Update: $SELECT\ ?variable\ Update$ After: $Data_1$

The Select URI Rule has the same shape. In the case of URIs, a correct URI to input is substituted for the temporary URI which is bound in the Select expression. Thus two URIs replace both the variable and literal in the Select Literal Rule.

Example of the Select Literal Rule. The following example demonstrates how Select can be used to delete some RDF Data which involves a literal not known in advance. The update deletes a triple in which the variable $?title$ appears. The variable can be instantiated with the literal "SPARQL Tutorial". Thus the delete matches the committed triple. Therefore the following commitment is valid.

Before: { $eg:book4\ dc:title\ "SPARQL\ Tutorial"$ }
 Update: $SELECT\ ?title\ \{$
 $DELETE\ \{eg:book4\ dc:title\ ?title\ \}$
 $\}$
 After: { }

5.5 The Choose Left Rule and Choose Right Rule

The Choose Rules allow one of two updates to be committed. The choose rule has a left and right form, where respectively the left or right update is applied. The result of a choice is the result of the update chosen. Consider the Choose Left Rule and suppose that the following commitment relation holds.

Before: $Data_0$ Update: $Update_0$ After: $Data_1$

Given the above commitment relation, the following commitment relation holds.

Before: $Data_0$ Update: $Update_0\ CHOOSE\ Update_1$ After: $Data_1$

The rule above chooses the left update. The Choose Right Rule is the symmetric rule which chooses the right branch instead.

Example of a Choice of Updates. The following demonstrates an update where either the first delete or second delete may be triggered. The two branches use different versions of the Dublin Core metadata vocabulary. In this case, the committed RDF Data matches the right branch. The result is that the committed triple is deleted.

Before: { $eg:book5\ dc11:title\ "SPARQL\ Protocol\ Tutorial"$ }
 Update: $SELECT\ ?title\ \{$
 $DELETE\ \{eg:book5\ dc:title\ ?title\ \}$
 $CHOOSE$
 $DELETE\ \{eg:book5\ dc11:title\ ?title\ \}$
 $\}$
 After: { }

Note that if both branches are satisfied then one branch is chosen non-deterministically. If the update is iterated then two copies of the update can be posed where each copy chooses a different branch. This is different from forcing both branches to be performed simultaneously, which would be expressed as a join of updates rather than as a choice between updates.

5.6 The Filter Axiom

The Filter Axiom imposes a constraint on an update. The constraint is disposed only if the constraint evaluates to true. If the constraint does not evaluate to true then the update is blocked. The procedure for deciding whether a constraint holds is specified in the SPARQL Query Recommendation [17]. Given that the constraint evaluates to true the following commitment relation holds.

Before: {} Update: FILTER *Constraint* After: {}

An Example of a Filtered Update. The following commitment relation holds. The update deletes the title of a book, where the title and the book are discovered using Select. The filter imposes the constraint that the title must also satisfy a regular expression. The literal in the committed triple does match the regular expression. The triple is deleted.

Before: { *eg:book4 dc:title "SPARQL Tutorial"* }
 Update: SELECT :*a*, ?*title* {
 DELETE { :*a* dc:title ?*title* }
 FILTER regex(?*title*, "^SPARQL")
 }
 After: {}

5.7 The Rules for Iterated Updates

All updates above are applied exactly once. Often the update should be applied whenever possible in an RDF store. This is achieved by iteration. The rules for iteration are similar to those for a Kleene star in a regular expression. Regular expressions are commonly used to update text files. This work is a generalisation of this common technique to RDF stores. Generalisations of regular expression date back to the commutative regular algebras of J. H. Conway [8], and remain a prominent area of research [14,10].

Updates can be applied any number of times. Iteration is used when the number of times to apply an update is not known. The Weakening Axiom allows an iterated update to be applied zero times, if there is no term which matches the update. The Weakening Axiom terminates an iterated update with no effect.

Before: {} Update: DO *Update* After: {}

The Dereliction Rule allows an iterated update to be applied once. Assume that an update can be committed in the presence of some term resulting in some process. Dereliction

allows the same update but iterated to be committed in the presence of the same term with the same resulting process. Suppose that the following commitment relation holds.

Before: $Data_0$ Update: $Update$ After: $Data_1$

Given the above commitment relation, the following commitment relation holds.

Before: $Data_0$ Update: $DO\ Update$ After: $Data_1$

The Contraction Rule allows two copies of an iterated update to be simultaneously committed. Contraction can be applied repeatedly, along with the Join Rule and Dereliction Rule, to simultaneously commit any number of copies of an iterated update. Suppose that the following commitment relation holds.

Before: $Data_0$ Update: $DO\ Update\ JOIN\ DO\ Update$ After: $Data_1$

Given the commitment relation above, the following commitment relation holds.

Before: $Data_0$ Update: $DO\ Update$ After: $Data_1$

The combination of the Weakening, Dereliction and Contraction rules allow zero, one, two, or more copies of an iterated update to be atomically committed. The use of Join in the Contraction Rule ensures that disjoint RDF Data is used for each copy of the update.

An Example of an Iterated Update. The following demonstrates an iterated update. The update replaces occurrence of the predicate *dc11:title* with the predicate *dc:title*. The iteration of this update means that the update can be applied twice. The result is that two triples are committed and replaced by two new triples.

Before: { *eg:book5 dc11:title "Query Tutorial"* },
 { *eg:book6 dc11:title "Update Tutorial"* }
 Update: DO SELECT *:a ?x* {
 DELETE { *:a dc11:title ?x* }
 INSERT { *:a dc:title ?x* }
 }
 After: { *eg:book5 dc:title "Query Tutorial"* },
 { *eg:book6 dc:title "Update Tutorial"* }

5.8 The Context Rule for Unused Data

The Context Rule allows some data to not be used in an update. This is important when considering updates in the context of an RDF store. The rule composes the same unchanged data before and after the update. Assume that the following commitment holds.

Before: $Process_0$ Update: $Update$ After: $Process_1$

Given the above commitment, the following commitment holds.

Before: $Process_2$, $Process_0$ Update: $Update$ After: $Process_2$, $Process_1$

The context rule is demonstrated in the example in the next section, where one of the two triples is not updated.

5.9 The Blank Node Rule for Updating Local Names

The Blank Node Rule is used for updates which involve blank nodes. The trick is to treat the blank node as a temporary URI in the premise of the rule. The temporary URI must not appear free in the conclusion of the rule, thus an extra side-condition is required. Suppose that the following commitment holds.

Before: $Data_0$, $Data_1$ Update: $Update$ After: $Data_2$, $Data_3$

Given that the above commitment holds, such that $:a$ is not free in $Data_0$ or $Data_2$, then the following commitment holds.

Before: $Data_0$, $BNODE :a Data_1$ Update: $Update$ After: $Data_2$, $BNODE :b Data_3$

An Example of the Blank Node Rule. The following example demonstrates a blank node updated. A temporary URI can represent $:a$ in the premise of the Blank Node Rule. This allows the update to be considered as if $:a$ is not bound. One triple is deleted by a commitment relation, which discovers the temporary URI. However, the conclusion of the Blank Node Rule still binds $:a$. This has the effect of discovering the blank node and using it in an update.

Before:

```
Before: BNODE :a {
  { :a foaf:name "Alice" } ,
  { :a foaf:mbox mailto:alice@example.org }
}
```

```
Update: SELECT :b {
  DELETE { :b foaf:mbox mailto:alice@example.org }
  INSERT { :b foaf:mbox mailto:alice@new.org }
}
```

```
After: BNODE :a {
  { :a foaf:name "Alice" } ,
  { :a foaf:mbox mailto:alice@new.org }
}
```

5.10 An Example of a Nested Update

This example, firstly, demonstrates most of the constructs combined to answer a larger update. Secondly, it demonstrates a common scenario which is enabled by nested selects and nested explicit iteration, which is impossible to express as an atomic update

in initial proposals for SPARQL Update [18,9]. Consider the following commitment, which removes all *foaf:knows* links to people younger than 18.

```

Before: { eg:youth0 eg:dob '01-01-2010' } ,
        { eg:youth1 eg:dob '01-02-2010' } ,
        { eg:person foaf:knows eg:youth0 } ,
        { eg:person foaf:knows eg:youth1 } ,
        { eg:youth0 foaf:knows eg:youth1 }
Update: DO SELECT :a ?dob {
        WHERE { :a eg:dob ?dob }
        FILTER (current-year - year(?dob) < 18)
        DO SELECT :b {
            DELETE { :b foaf:knows :a }
        }
    }
After: { eg:youth0 eg:dob '01-01-2010' } ,
        { eg:youth1 eg:dob '01-02-2010' }

```

Without the nested iteration and selects, the effect of the above update could only be achieved using two updates. This means that the update would not be atomic. The above update is correct and atomic. This example highlights a common problem which also appears in the first SPARQL Query recommendation [17]. This illustrates an improvement made by this work to the expressiveness of the language.

6 Related Work

Updates for RDF have been considered before using the RUL language [7]. The work on RUL focusses on the effect of updates in the presence of RDFS entailment. The effect of RDFS entailments and the operational semantics of the core of SPARQL Update are perpendicular issues. Thus, this update language can be extended to accommodate RDFS entailment as considered in RUL.

This language only claims to model a core of SPARQL Update. A feature missing is the handling of named graphs [6]. Named graphs can be achieved naïvely by allowing quads as well as triples, in the data format. However, further subtleties may arise depending on design decisions of the working group.

An operational semantics for updates enables further formal investigations. This operational semantics can be used to derive equivalences over updates, where two equivalent updates are operationally indistinguishable. This equivalence can be used to verify an algebra for rewriting updates without changing their operational behaviour. An algebra over updates is useful for the optimisation of updates. This line of work has been investigated for queries by the authors [12]. The same results also hold for updates; for instance iteration, join, choice, true and false form a commutative Kleene algebra with tests and quantifiers [14]. Kleene algebras are a common and useful algebra in computer science, a prominent example being the equations of regular expressions.

An operational semantics also allows a type system for updates to be specified. A simple static type system checks that literals are of the correct type to evaluate any

constraints in which they appear. The operational semantics are then essential to verify that the static properties guaranteed by the type system are preserved by the behaviour of updates. A type system can be realised through type reconstruction, meaning that a programmer does not need to program using types for the type system to be useful. More powerful type systems inspired by RDFS can also ensure that URIs are used consistently, as investigated by the first author [11].

7 Conclusion

This work introduces a language with an operational semantics for fine grained updates over RDF. A fine grained update language is important for enabling a Read–Write Web of Data [3], where contributing content is as important as consuming content. SPARQL Update is currently being developed by the W3C as a standardised fine grained update language for RDF, for which an operational semantics is beneficial [9]. This work provides the first such operational semantics in the established tradition of Plotkin [16].

A specification with an operational semantics has many benefits. This operational semantics can be used as the basis of tools for compiler verification, type checking and optimisation [12,11]. It can also be used to evaluate the proposed language to tackle issues including: redundant operations, ambiguous definitions, rigid syntax, inadequate expressiveness, and incomplete specifications, as described below.

The operational semantics exposes redundant operations, which could instead be expressed using a combination of operators. Redundancies reduce the number of operators which must be directly implemented. For instance, it is shown in Sec. 3.5 that it is a myth that operator OPTIONAL is primitive.

Ambiguous definitions are clarified by the operational semantics. For instance, it is not clear in the original proposal [18] whether queries occur sequentially before or in parallel to an update. In this operational semantics queries occur in parallel, which allows more concise updates and avoids concurrency issues.

The abstract syntax (Sec. 3.4) is more explicit than the concrete syntax. For instance, the quantifiers of the abstract syntax clearly delimit the scope of a variable; whereas in the original proposal it is not obvious which variables share the same scope. Furthermore the abstract syntax is compositional, meaning that updates can be written in a modular fashion then combined, enabling interesting programming techniques and optimisations.

Some modest improvements to the expressive power of the language can be suggested. For instance, the CHOOSE operator (Sec. 5.5) can be extended to the entire language, instead of only queries. Similarly, Example 5.10 cannot be expressed without explicit iteration. Such improvements enable some common scenarios to be expressed.

The operational semantics presented here, goes considerably further than the formal model sketched in the current working draft. For instance, this operational semantics precisely specifies how blank nodes are updated (Sec. 5.9). Importantly, the model is purely operational, unlike the sketched model which combines operational and denotational approaches.

This work has been presented without using any meta-syntax to express the operational semantics. This can be achieved since an operational semantics is defined directly over the syntax of a language. The intention is that this style of presentation of operational semantics is accessible to a diverse audience.

References

1. Abramsky, S.: What are the fundamental structures of concurrency? We still don't know! Electronic Notes in Theoretical Computer Science 162, 37–41 (2006); Proceedings of the Workshop Essays on Algebraic Process Calculi
2. Beckett, D., Berners-Lee, T.: Turtle - Terse RDF Triple Language. Team submission, W3C (2008)
3. Berners-Lee, T.: Read-Write Linked Data, personal view (December 2010), <http://www.w3.org/DesignIssues/ReadWriteLinkedData.html>
4. Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., Hendler, J.: N3logic: A logical framework for the World Wide Web. Theory and Practice of Logic Programming 8(3), 249–269 (2008)
5. Brickley, D., Guha, R.: RDF vocabulary description language 1.0: RDF Schema. Recommendation REC-rdf-schema-20040210, W3C (2004)
6. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs. Journal of Web Semantics 3(4), 247–267 (2005)
7. Magiridou, M., Sahtouris, S., Christophides, V., Koubarakis, M.: RUL: A Declarative Update Language for RDF. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 506–521. Springer, Heidelberg (2005)
8. Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall (1971)
9. Gearon, P., Passant, A., Polleres, A.: SPARQL 1.1 Update. Working draft WD-sparql11-update-20110512, W3C (May 2011)
10. Hoare, C., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene Algebra. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 399–414. Springer, Heidelberg (2009)
11. Horne, R.: Programming Languages and Principles for Read–Write Linked Data. Ph.D. thesis, University of Southampton (2011)
12. Horne, R., Sassone, V.: A verified algebra for Linked Data. In: FOCLASA 2011, Aachen, August 10. Electronic Proceedings in Theoretical Computer Science, vol. 58, pp. 20–33 (2011)
13. Klyne, G., Carroll, J.: Resource Description Framework: Concepts and abstract syntax. Recommendation REC-rdf-concepts-20040210, W3C (2004)
14. Kozen, D.: Kleene algebra with tests. ACM Transactions on Programming Languages and Systems 19, 427–443 (1997)
15. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Transactions on Database Systems 34(3), 1–45 (2009)
16. Plotkin, G.D.: A structural approach to operational semantics. internal notes. Aarhus University (1981)
17. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Recommendation REC-rdf-sparql-query-20080115, W3C (2008)
18. Seaborne, A., Manjunath, G.: SPARQL/Update: A language for updating RDF graphs. External HPL-2007-102, HP Labs Bristol (2007)

Appendix: Summary of the Operation Semantics

For a concise summary of the operational semantics, some meta-syntax is introduced. Commitments are written in the body of the paper as follows.

Before: $Data_0$ Update: $Update$ After: $Data_1$

In this summary commitments are instead written as follows.

$$Data_0, Update \longrightarrow Data_1$$

Rules which are explained in English in the body of this work are presented in the style of natural deduction, as conventional for operational semantics [16]. The horizontal line separate the premises on the top from the conclusion on the bottom. Some rules have side conditions. The axioms and rules of the core update language are therefore summarised as follows.

$$\text{Delete: } Data, DELETE Data \longrightarrow \{\} \quad \text{Insert: } \{\}, INSERT Data \longrightarrow Data$$

$$\text{Filter: } \{\}, FILTER Constraint \longrightarrow \{\} \text{ only if } Constraint = \text{true}$$

$$\text{Join: } \frac{Data_0, Update_0 \longrightarrow Data_2 \quad Data_1, Update_1 \longrightarrow Data_3}{Data_0, Data_1, Update_0 JOIN Update_1 \longrightarrow Data_2, Data_3}$$

$$\text{Select literal: } \frac{Data_0, Update["literal"/?variable] \longrightarrow Data_1}{Data_0, SELECT ?variable Update \longrightarrow Data_1}$$

$$\text{Select URI: } \frac{Data_0, Update[:b/:a] \longrightarrow Data_1}{Data_0, SELECT :a Update \longrightarrow Data_1}$$

$$\text{Choose left: } \frac{Data_0, Update_0 \longrightarrow Data_1}{Data_0, Update_0 CHOOSE Update_1 \longrightarrow Data_1}$$

$$\text{Choose right: } \frac{Data_0, Update_1 \longrightarrow Data_1}{Data_0, Update_0 CHOOSE Update_1 \longrightarrow Data_1}$$

$$\text{Weakening: } \{\}, DO Update \longrightarrow \{\} \quad \text{Dereliction: } \frac{Data_0, Update \longrightarrow Data_1}{Data_0, DO Update \longrightarrow Data_1}$$

$$\text{Contraction: } \frac{Data_0, DO Update JOIN DO Update \longrightarrow Data_1}{Data_0, DO Update \longrightarrow Data_1}$$

$$\text{Structure: } \frac{Data_0 = Data_2 \quad Data_0, Update \longrightarrow Data_1 \quad Data_1 = Data_3}{Data_2, Update \longrightarrow Data_3}$$

$$\text{Context: } \frac{Data_0, Update \longrightarrow Data_1}{Data_0, Data_2, Update \longrightarrow Data_1, Data_2}$$

$$\text{Blank node: } \frac{Data_0, Data_1, Update \longrightarrow Data_2, Data_3}{Data_0, BNODE :a Data_1, Update \longrightarrow Data_2, BNODE :a Data_3}$$

only if :a does not appear free in $Data_0$ or $Data_2$

Any commitment which can be derived using the above axioms and rules is a valid commitment. Valid commitments include all examples in the body of this work.

Knowledge-Driven Diagnostic System for Traditional Chinese Medicine

Peiqin Gu and Huajun Chen

Zhejiang University, Hangzhou 310027, P.R. China
{gupeiqin, huajunsir}@zju.edu.cn

Abstract. Recognizing diseases from theoretical perspective can help ordinary people have a general understanding of medicine. The usual process of identifying syndromes or diseases in Traditional Chinese Medicine (TCM) is by confirming the frequently symptom patterns. Semantic Web and ontologies introduce well-structured controlled vocabularies for biomedical science. The direct correspondence between symptoms and syndromes can be formatted to semantic inference rules as a additional knowledge upon a medical ontology.

In this paper, we present a simplified rule-based diagnostic system for febrile disease theory in TCM, which make use of the capability of semantic inference based on medical ontology. Actually the method is rather general for logic-based medical diagnosis, and we show that without interpreting clinical data, the medical knowledge itself can be applied to do basic clinical diagnosis.

Keywords: Semantic Web, Domain ontology, Bioinformation.

1 Introduction

In the Semantic Web, resources and relations between them are annotated with semantic markups which are defined in Web Ontology Language OWL [5] and Resource Description Framework (RDF) [4]. Knowledge defined as semantic ontology can be processed to obtain novel information through inference technologies.

Biological knowledge [1] is inherently complex and so cannot readily be integrated into existing databases of sequential data. Building domain ontology is a formal way of representing knowledge in which concepts and relationships are both based on logic. Unique identifiers that are associated with each concept in biological ontologies can be used for linking to and querying other knowledge sources, such as other ontologies and medical databases.

Traditional Chinese Medicine (TCM) is a complete medical system that covers basic theories, diagnostics, diseases, therapeutics, and medical treatments as well as Western Medicine. However, unlike Western Medicine, the basic theory of TCM is based on the ancient ideas of Chinese natural philosophy, such as the essential *Qi* theory, the *Yin-Yang* theory, and the *Five Phase* theory. Ancient Chinese philosophers were greatly interested in the relationship and patterns that occurred between human body and the nature. They viewed the world as a harmonious and integrated entity, instead of a set of disparate and isolated things. Influenced by these ideas, TCM is characterized by holism, regarding the human body as an organic whole constantly interacted with the

external environment, rather than an anatomical organism. In TCM, a disease or syndrome is viewed as a holistic morbid pattern resulting from the interactions between local pathological and maladjustments of the body.

The existence of ontology knowledge modeling inherited from the Semantic Web is crucial to bring structural and logical elements to medicine knowledge management. In this paper, we define a shared domain ontology *WenBing*, which is supported by formal knowledge representation languages known as World Wide Web Consortium standards. Among them, the RDF [4] is a standard model for representing information on the Web. As an extension of RDF model, OWL [5] has been designed to present the explicit meaning of various domains as an ontology model, which can be used to define classes, properties, equalities, property characteristics, and etc.

Once the medicine ontology has been built or partially built by domain experts, it can be used to conduct the medical diagnostic process, because now computers are able to recognize the semantics of the texts. As mentioned above, TCM clinical diagnosis embodies the philosophical concepts or rule patterns in nature. Thus, the relationships between concepts and fact assertions between individuals are taken as variables, atom assertions separately, making up the body and head of OWL rules. Our diagnostic system take these rules as patterns to compare with user inputs of symptoms, then output the most likely syndrome as the diagnostic suggestion.

2 Medical Ontology Modeling

Medical diagnostic decision making have long been great interest among AI researchers for dozens of years. Especially TCM is recognized to have unique characteristics such as vagueness, linguistic uncertainty, hesitation, measurements imprecision, natural diversity, subjectivity. The entirely expertise-dependent characteristics of diagnosis and vagueness of medical terms in TCM are great challenges to knowledge acquisition, which has been known as the bottleneck of diagnosis systems in TCM. Many classical methods for representing and matching ontological knowledge such as description logics, frame-based representations and semantic nets have gained their significance in artificial intelligence. Semantic ontologies provide a useful way to make a description model for real world application and research domains. The core component of the Semantic Web is web ontology which represents the relationships between web resources. Semantic Web ontologies are described in a RDF-based standard Web ontology language OWL which has its logical foundation on DL (Description Logic).

Due to the accuracy limitation of data analysis especially for medicine, the demand for building trustable theoretical model and knowledge-driven application is growing. Among these technologies, semantic rules upon ontologies hold various facts outside the knowledge model itself rather than a large directed labeled graph.

The medical community has long been recognized the need of modeling its knowledge and of making its terminologies explicit, shared and linkable. Therefore, there exists several terminological or ontological resources in medical domain: GALEN [6], MENELAS [10], ON9 Library [2], SNOMED RT [7], UMLS [3], which are all ontologies or thesauri that model a sub topic of the medical domain.

As a pioneer of engineering TCM knowledge, the team from China spent several years building the unified traditional Chinese medical language system [9] to study the terminology standardization, knowledge acquisition and integration in TCM. Based on these previous work, several small ontologies for part of TCM domain are built as well as the *WenBing*.

Semantic Ontology is a technology for knowledge representation which describes some concepts and relationships between them. By inferring Web ontologies and semantic markups in ontologies and Web documents, meaningful information and resources can be retrieved.

3 WenBing Domain Ontology

In this paper, we introduce the *WenBing*, a domain ontology that models the TCM clinical reasoning by representing the basic concepts, theories, rules and principles in formal languages. In computer science, a domain ontology is defined as a formal, explicit specification of a shared conceptualization.

We implement the ontology *WenBing* using the Semantic Web, a Web-based knowledge representation and integration technology. The Semantic Web provides formal languages, such as Resource Description Framework (RDF), RDF Schema (RDFS), and Web Ontology Language (OWL), to express domain ontologies that any intelligent agents can understand.

We engineer the ontology *WenBing* with the following process:

- Conduct a systematic analysis of TCM literature and extract a set of concise statements about the TCM clinical logic.
- Codify the meaning of TCM concepts, and define the semantic relationships between these concepts.
- Represent TCM domain knowledge as formal statements and rules.

The resulting domain ontology covers such categories as basic theories, diagnostics, diseases, therapeutics, and medical treatments. The *WenBing* formalizes the basic concepts, theories, rules, and principles in clinical reasoning, such as syndrome classification and differentiation, therapeutic principles, and herbal formula combination. The experimental *WenBing* contains 167 concepts, 14 object properties, 151 class assertions and 68 object property assertions. *WenBing* concepts are not only hierarchically defined, but also inter-related by binary relationships. We develop the domain ontology to store the facts of the febrile disease theory in TCM. Automated reasoning methods can then be used to infer consequences of the ontology. The *WenBing* can be utilized to represent such resources as medical rules, and facilitate such applications as clinical intelligence, and knowledge management and sharing.

The *WenBing* formalizes the basic concepts, theories, rules, and principles in clinical reasoning, such as syndrome classification and differentiation, therapeutic principles, and herbal formula combination, thus the *WenBing* can facilitate such applications as information integration and exchange, clinical intelligence, and knowledge management and sharing.

3.1 Basic Theory of Febrile Disease

As a important branch of TCM, febrile diseases are often related to abnormally high body temperature. Fever is a common symptom of many medical conditions such as infectious disease, various skin inflammations, tissue destruction, and etc. In all febrile diseases with a hot, dry skin, hot breath, and thirst, the sedative influence of cold is strongly indicated. It may be applied by means of cool fresh air inhaled into the lungs, which in general proves grateful and refreshing.

Febrile disease is usually very common, its occurrence has obvious seasonal, most rapid onset, rapid mass change, and mostly have varying degrees of contagious and epidemic.

To distinguish between infectious and epidemic, febrile diseases infectious and pandemic with a strong characteristic are called plague disease, febrile diseases which cause small or non-epidemic disease are characterized as normal febrile disease. In recent years, H1N1 and SARS both take fever as clinical manifestation, and they can be classified into febrile disease. To prevent from wide range of major disease outbreak, it is important to study the incidence and treatment of febrile disease.

3.2 Terminology and Assertions

As a Semantic Web ontology, the *WenBing* mainly contains two parts:

- The *TBox* is a box of terminology. The *TBox* defines a system of controlled vocabularies consisting of (1) *classes* (such as *Drug*, *Herb*, and *Disease*) that defines types of things, (2) *properties* (such as *treats*, *consists_of*, and *interacts_with*) that defines binary relationships between things, and (3) individuals (such as *qi*, *yin*, and *wood*) that define things that exist in the domain, which are associated with each other through properties, and which are grouped into classes.
- The *ABox* is a box of assertions. *ABox* are *TBox*-compliant statements that capture the knowledge content of TCM domain.

The Semantic Web is based on the idea of using global identifiers (Uniform Resource Identifiers, or URIs) to identify any things, including physical objects (such as drug or an organ), conceptual entities (such as febrile disease), events (such as therapy and diagnosis), and generalized concepts (such as semantic relations and classes).

For example, a treatment method *Clearing qi, cooling nutrient and detoxify* can be identified as:

```
<http://www.semanticweb.org/wenbing/cqnd>.
```

A resource is defined as anything that is identified by a URI. Since the syntax of URIs is rather verbose and difficult to read, we use abbreviated URIs: once a prefix such as @prefix *wb*:<http://www.semanticweb.org/wenbing> is defined, the qualified name *wb:cqnd* becomes a shorthand for the URI. In addition, we use <http://www.semanticweb.org/wenbing> as the default namespace, therefore, *wb:cqnd* can be further abbreviated as *cqnd*. In RDF and OWL, a set of such triple (*s*, *p*, *o*) intuitively means that a relation denoted by *p* exists between *s* and *o*, where *s*, *p*,

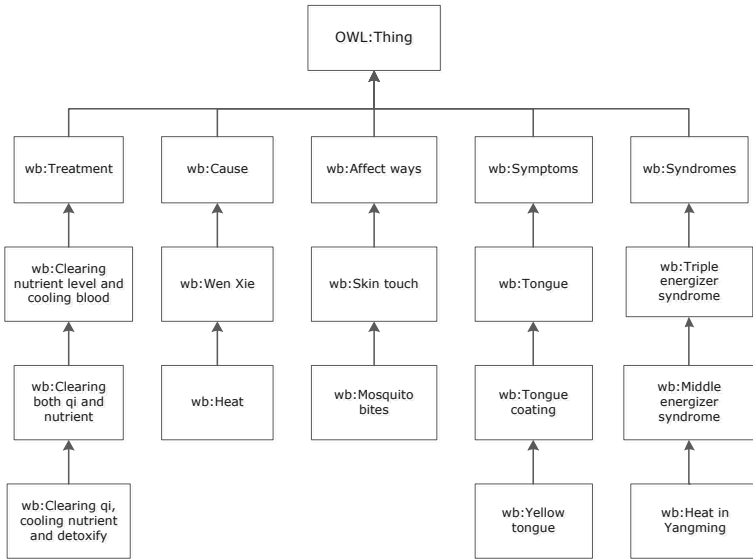


Fig. 1. The class subsumption graph of WenBing ontology

and *o* are all resources identified by URIs. The object relations we concern in building *WenBing* ontology is given in Table 1. The following triple asserts that the treatment method *cqnd* is made according to the specific syndrome *Qi and nutrient burnt*:

```
<http://www.semanticweb.org/wenbing/cqnd>
<http://www.semanticweb.org/wenbing/according_to>
<http://www.semanticweb.org/wenbing/qi_nutrient_burnt>.
```

and the medication for the treatment method *cqnd* is usually the prescription *Qingwen Baidu Yin*:

```
<http://www.semanticweb.org/wenbing/cqnd>
<http://www.semanticweb.org/wenbing/apply>
<http://www.semanticweb.org/wenbing/qingwen_baidu_yin>.
```

RDF provides a standard way of representing terminological vocabularies in the knowledge domain, thus can be used to capture the semantic relations between resources or vocabularies. This *WenBing* stands for a sound knowledge base for TCM available for further reasoning and querying.

3.3 Modeling Syndrome Classification

TCM practitioners typically characterize patients with *TCM syndromes (zheng)*, which are integrated patterns of human morbidity indicating the hidden cause of observable symptoms. A *syndrome type* is a common syndrome mode with a standard name. Each

Table 1. Semantic relations in *WenBing*

Relation	Definition	Domain	Range
<i>according to</i>	The decision of treatment measures is based on syndrome.	<i>treatment</i>	<i>syndrome</i>
<i>invasion</i>	The ways of clinical invasion.	<i>pathogeny</i>	<i>invasion ways</i>
<i>cause</i>	Lead to.	<i>pathogeny</i>	<i>febrile disease</i>
<i>invalid</i>	Not directly associated.	<i>prescription</i>	<i>symptoms</i>
<i>valid</i>	Directly associated.	<i>prescription</i>	<i>health care</i>
<i>has symptom</i>	The patient gets some symptom.	<i>patient</i>	<i>symptom</i>
<i>has syndrome</i>	The patient gets some syndrome.	<i>patient</i>	<i>syndrome</i>
<i>compose</i>	Constitutes something as a part.	<i>drug</i>	<i>prescription</i>
<i>display</i>	Show as an appearance.	<i>febrile disease</i>	<i>symptom</i>
<i>differentiate</i>	To see or decide a difference between things.	<i>symptom</i>	<i>syndrome</i>
<i>apply</i>	To use a particular prescription for treatment.	<i>treatment</i>	<i>prescription</i>
<i>prevent</i>	Prevent.	<i>prevention</i>	<i>febrile disease</i>

syndrome type is defined as a composite concept in terms of yin-yang, five phases, viscera and bowels, etc. The physician determines the specific syndrome for patient by the identification of symptoms, and constructs a logical justification for a decision of intervention based on therapeutical principles. For example, a physician identifies that a patient has the syndrome *Heat_In_YangMing*, which usually happen during summer when the heat turns into invading power by recognizing a set of symptoms such as thirsty, rapid pulse, etc. The physician reasons that an intervention is needed to purge the heat and clear the qi inside the body.

Terminology hierarchy in the medical ontology is not enough for knowledge inference. An OWL ontology contains a sequence of axioms and facts as we defined using RDF and OWL description languages. Due to the limited capability of representing knowledge, the usage of rules is imported as an extension for the ontology. The rules are abstract deduction relations extracted from clinical principles. A rule axiom consists of an antecedent and a consequent, each of which consists of a set of atoms:

$$\begin{aligned}
 \text{rule} &:= \{ \text{antecedent}, \text{consequent} \} \\
 \text{antecedent} &:= \text{atom} \\
 \text{consequent} &:= \text{atom} \\
 \text{atom} &:= \{ i\text{-object} \} \\
 i\text{-object} &:= i\text{-variable} | \text{individual}
 \end{aligned}$$

Informally, a rule means that if the antecedent holds, then the consequent must also hold. We express clinical facts such as the correspondence between symptoms and syndromes as a set of rules between them.

Take the syndrome *Heat_In_YangMing* in the domain ontology *WenBing* for example, we formalize the usual diagnostic pattern of symptoms in to a rule stating that:

$$\begin{aligned}
 \text{rule} &:= \{ \{ ?X \text{ has_symptom } \text{high_fever}, \\
 &\quad ?X \text{ has_symptom } \text{much_sweating}, \\
 &\quad ?X \text{ has_symptom } \text{thirsty}, \\
 &\quad ?X \text{ has_symptom } \text{upset}, \\
 &\quad ?X \text{ has_symptom } \text{headache},
 \end{aligned}$$

```

?X has_symptom dizzy,
?X has_symptom breathless,
?X has_symptom slight_chill,
?X has_symptom red_tongue,
?X has_symptom yellow_tongue_surface,
?X has_symptom dry_tongue,
?X has_symptom flood_pulse,
?X has_symptom rapid_pulse, },
?X has_syndrome Heat_In_YangMing, }

```

In particular, the syndrome *Heat_In_YangMing* indicates that the summer heat invades the human body, one of the syndrome as a subclass of *Middle_Energizer_Syndrome* (*Heat_In_YangMing* **subclass** *Middle_Energizer_Syndrome*). However, its meaning becomes more clear when it is associated with causing sweating, dry tongue and flood pulse, etc. The syndromes in the syndrome hierarchy as shown in Figure 1 appear to be disjoint, but the set of symptoms connected to each syndromes may have intersections.

4 TCM Diagnostics

In TCM diagnostic theory, a pathologic condition of a patient is diagnosed according to abnormal data collected through two ways [8]: classical diagnosis based on four conventional examinations, and patient' own report. According to observed symptoms and patient's own report, TCM practitioners will perform diagnosis about patient's pathological conditions in terms of syndromes. Functional imbalance and specific manifestations of disease are differentiated as syndrome and corresponding intervenes for treatment will be taken subsequently. Hence, the concept of *syndrome differentiation* is of great importance in the diagnostics of TCM.

In medicine, a syndrome is the association of several clinically recognizable features, signs (observed by a physician), symptoms (reported by the patient), phenomena or characteristics that often occur together. It may includes a long list of possible symptoms and combinations of symptoms, and most of these symptoms are subjective.

In TCM diagnostics, we have to solve the problem of symptom matching in order to behave efficient syndrome differentiation. We assume the diagnostic system accepts a list of symptoms both observed by doctors and the user himself from the specific user interface, and finally it outputs a proper diagnosis based on domain knowledge stored as an ontology.

Definition 1. *The input for our TCM diagnosis system $S = \{s_1, s_2, \dots, s_n\}$ stands for a user specified collection of abnormal symptoms.*

Definition 2. *Let $C = \{C_1, C_2, \dots, C_M\}$ be a set of M diagnoses possible in the context of the febrile disease theory. C would be: a single disease name plus a combination of syndromes, etc.*

The problem of medical diagnosis can be formalized as follows: given a set of symptoms $S = \{s_1, s_2, \dots, s_n\}$ and a group of rules $R = \{r_1, r_2, \dots, r_p\}$ where each r is

Algorithm 1. Symptom Matching Algorithm**Input:** $S = \{s_1, s_2, \dots, s_n\}$, $R = \{r_1, r_2, \dots, r_p\}$ **Output:** $C = \{C_1, C_2, \dots, C_M\}$ $index = 0;$ $count = 0;$

```

for  $i = 0; i < R.size; i++$  do
   $hash = R.get(i);$ 
  for  $j = 0; j < S.size; j++$  do
    if ( $hash.contains(S.get(j))$ ) then
       $temp++;$ 
    end
  end
  if ( $temp > count$ ) then
     $count = temp;$ 
     $index = i;$ 
  end
  return  $R.get(index);$ 
end

```

composed of symptoms and a syndrome, determine a set of diagnoses $C = \{C_1, C_2, \dots, C_M\}$. The *symptom matching* algorithm is a process of calculating the text similarity for each input symptoms, in order to obtain a most likely answer related to defined rules. It keeps the theoretical accuracy assuming that the symptoms are disjoint, and independent. This method returns the text-closest answer, yet may get rid of some semantic similarities between syndromes or symptoms, which is for our next step to promote the accuracy of actual medical diagnosis.

5 Medical Scenario

We have built a Web-based diagnostic platform to facilitate the knowledge-driven clinical diagnosis for febrile disease in TCM, which allows both the common people and TCM practitioners to explore the performance of diagnostics. This platform contains the following components: (1) As shown in Figure 2, a graphical user interface (GUI) is provided to view the description of knowledge model in *WenBing*. (2) As shown in Figure 3, a diagnostic panel is created online to enable Web users submit any combination of symptoms, then the diagnostic system give processed suggestion of syndromes, treatment method and prescription. The *WenBing* ontology is stored and managed in a back-end knowledge base, and is also accessible to client programs via a Web-based programmable interface.

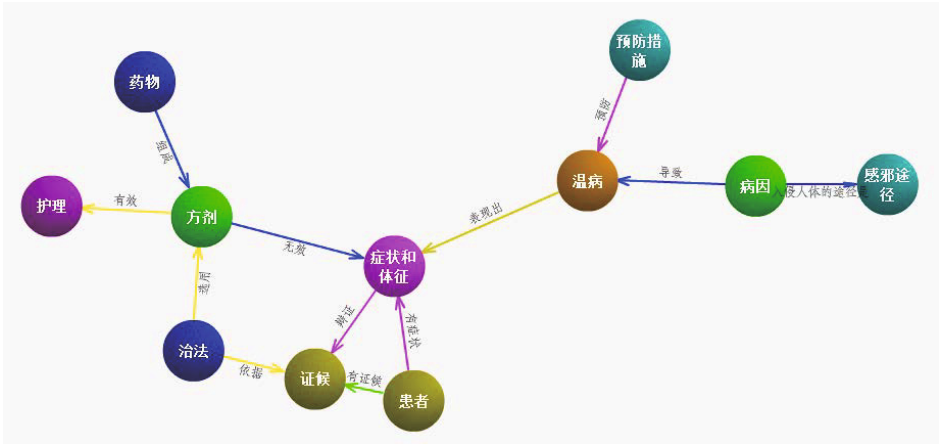


Fig. 2. The model view of WenBing ontology, which indicates the relationships between concepts



Fig. 3. The user interface of the diagnostic panel

6 Conclusions

A shared conceptualization of the clinical logic in Traditional Chinese Medicine (TCM) is fundamental to the preservation, investigation, utilization, and further development of this cultural heritage. The characteristics of TCM clinical logic, such as ancientness, fuzziness, diversity, and complexity, require advanced knowledge modeling techniques. To deal with these issues, we model the TCM clinical logic as a domain ontology, and use the Semantic Web technologies to implement the ontology. The modeling process includes: (1) constructing a formal terminology that codifies the meaning of complex

TCM concepts; and (2) constructing semantic graphs that represent semantic relationships between these concepts. The resulting domain ontology, namely *WenBing*, is a standard codification of the febrile disease theory in TCM. *WenBing* provides a knowledge model to foster the understanding of TCM clinical reasoning, and to facilitate the decision making in clinical studies of integrated medicine.

In TCM, syndrome differentiation is the most important part in diagnostics. In order to behave knowledge-driven inference, diagnostic principles are formalized as assertions and rules, which are also conceptual reflection of TCM clinical logic. The correspondent relationships are represented as logic rules, and the diagnostic system take the collection of such rules and user inputs of symptoms as an input, check text similarities one by one for each user-specified symptom to get the most likely syndrome according to rules.

References

1. Bard, J.B.L., Rhee, S.Y.: Ontologies in biology: Design, applications and future challenges. *Nature Reviews Genetics* 5, 213–222 (2004)
2. Gangemi, A., Pisanelli, D.M., Steve, G.: Ontology integration: Experiences with medical terminologies. In: *International Conference on Formal Ontology in Information Systems* (1998)
3. Humphreys, B.L., Lindberg, D.A.B.: The unified medical language system. *Methods of Information in Medicine* (1993)
4. Manola, F., Miller, E.: Rdf primer: W3c recommendation. *Decision Support Systems* (2004)
5. Mcguinness, D.L., Harmelen, F.V.: Owl web ontology language overview. In: *World Wide Web* (2004)
6. Rector, A., Gangemi, A., Galeazzi, E., Glowinski, A., Rossi-Mori, A.: The galen core model schemata for anatomy: Towards a re-usable application-independent model of medical concepts. In: *Medical Informatics Europe* (1994)
7. Spackman, K.A., Campbell, K.E., Cote, R.A.: Snomed rt: A reference terminology for health care. In: *Proceedings of the 1997 AMIA Annual Symposium*, pp. 640–644 (1997)
8. Wang, X., Qu, H., Liu, P., Cheng, Y.: A self-learning expert system for diagnosis in traditional chinese medicine. *Expert Systems with Applications* 26, 557–566 (2004)
9. Zhou, X., Wu, Z., Yin, A., Wu, L., Fan, W., Zhang, R.: Ontology development for unified traditional chinese medicine language system. *Artificial Intelligence in Medicine* 32, 15–27 (2004)
10. Zweigenbaum, P.: Menelas: coding and information retrieval from natural language patient discharge summaries. *Advances in Health Telematics* (1995)

LODDO: Using Linked Open Data Description Overlap to Measure Semantic Relatedness between Named Entities

Wenlei Zhou, Haofen Wang, Jiansong Chao,
Weinan Zhang, and Yong Yu

APEX Data & Knowledge Management Lab
Department of Computer Science Engineering
Shanghai Jiao Tong University, Shanghai, China
{wenlei.zhouwl,whfcarter,jiansong.chao,wnzhang,yyu}@apex.sjtu.edu.cn

Abstract. Measuring semantic relatedness plays an important role in information retrieval and Natural Language Processing. However, little attention has been paid to measuring semantic relatedness between named entities, which is also very significant. As the existing knowledge based approaches have the entity coverage issue and the statistical based approaches have unreliable result to low frequent entities, we propose a more comprehensive approach by leveraging Linked Open Data (LOD) to solve these problems. LOD consists of lots of data sources from different domains and provides rich a priori knowledge about the entities in the world. By exploiting the semantic associations in LOD, we propose a novel algorithm, called LODDO, to measure the semantic relatedness between named entities. The experimental results show the high performance and robustness of our approach.

Keywords: Named Entity, Semantic Relatedness, Linked Open Data.

1 Introduction

Semantic relatedness measuring plays an important role in the area of natural language processing (e.g., word sense disambiguation [14]) and information retrieval. With the advance of Semantic Web, more and more documents are annotated with real world entities. Hence, measuring semantic relatedness between these named entities can be regarded as an effective mean to capture semantic associations between documents, which can be further used for semantic search.

In recent years, there are abundant research studies on measuring semantic relatedness between words. They tried to solve the following two challenges:

- Word Ambiguity. A word might refer to different meanings or can represent different entities.
- Different Representations of a Single Entity. Even for a unique entity, it may have different representations, which requires us to collect all synonyms of a given word.

The existing work can be divided into two types: knowledge based approaches and statistical based approaches. The former ones basically leverage a high-quality knowledge source like WordNet [12] or Wikipedia¹. The main limitation of this kind of work is the coverage issue. While Wikipedia is the world largest domain independent knowledge base, it misses a number of entities in some specific domain. On the other hand, statistical based approaches mainly exploit the Web for this task. However, they fail to provide reliable semantic relatedness between words of low frequencies.

In this paper, we propose a novel approach to overcome the previous problems by leveraging Linked Open Data [1] (LOD). LOD is an abundant Web of data which contains a vast number of named entities. It is constructed by linking diverse data sources. While the openness of the Web might involve data noise, we assume LOD as high-quality data sources since they are published from existing structural or qualified databases. As the data sources cover many domains, given a named entity, it is highly possible that there is some description about it in LOD. Thus entity coverage problem can be eased by using LOD. On the other hand, while the statistical based approaches regard named entities which have the same name in all documents as the same entity, LOD represents them as different entities. As a result, each entity in LOD has its own description and it is distinguished from other entities of the same name.

The contributions of this paper are threefold. First, we build an efficient LOD index mechanism to solve the two challenges: word ambiguity and different representations of a single entity. Second, we propose a novel approach LODDO to accurately measure the semantic relatedness between named entities by exploiting the semantic associations in LOD. Third, the experiments result shows that our approach outperforms the existing semantic relatedness measuring approaches by at least 39.6%.

The remainder of the paper is organized as follows. In section 2 we discuss previous work related to named entities semantic relatedness measuring. The methodology is presented in section 3. The conducted experiments and the benchmark dataset with the evaluation results are presented in section 4. In section 5 we conclude the paper and discuss the future work.

2 Related Work

The existing semantic relatedness measuring approaches can be grouped into two types according to the sources they use: knowledge based approaches and statistical based approaches. The knowledge based approaches take advantage of a high-quality knowledge source such as WordNet, Roget or Wikipedia. The statistical based approaches calculate the statistical information of words by using Web corpus as their source.

Regarding the knowledge source as a graph of concepts connected with others, a straightforward approach to calculate semantic relatedness between two words (concepts) is to find the length of the path connecting the two words in

¹ <http://www.wikipedia.org/>

the graph [17,10,9,22]. Based on the intuition that the relatedness of two words (concepts) can be measured by the amount of information they share, Strube and Ponzetto [20,16] applied intrinsic information content to Wikipedia category graph. Resnik [18] used information content based on WordNet to measure semantic similarity. Hypothesizing that the higher word overlap in two concepts' glosses, the stronger semantic relatedness of these two concepts, Lesk [11] and Banerjee [3] introduced a measure based on the amount of word overlap in the glosses of two concepts. Strube [20] regarded the first paragraph of the concept's Wikipedia article as the concept's glosses. Patwardhan [15] calculated the cosine of the second-order gloss vectors which represented the corresponding words by using WordNet glosses. Gabrilovich [7] introduced ESA which constructed concept vectors from Wikipedia articles where each vector element represented an article. Milne [13] constructed the vectors by using the interlink articles.

For abstract concepts semantic relatedness measuring, single domain independent knowledge source may be enough to cover all the concepts. However, when dealing with hundreds of millions named entities in our real life, the coverage problem arises. Research [23] has also shown that the accuracy differs depending on the choice of the knowledge sources, and there is no conclusion which knowledge source is superior to others. It seems that different knowledge source may have its own preference in describing data, and thus it is unreliable to just use single knowledge source when measuring semantic relatedness.

The statistical based approaches calculate the statistical information of words by using Web corpus as their source. Bollegala [5] used four popular co-occurrence measures to calculate page-count-based similarity metrics for the pairs of single words and automatically extracts lexico-syntactic patterns about the pairs of single words based on the title, snippet and URL of the Web search results. Spanakis [19] modified Bollegala's method by adding consideration of the "Bag of Words" representation to the Web search results text for each single word. Since a named entity usually contains more than one word, the lexico-syntactic patterns extraction cannot be used directly. Gracia [8] proposed a transformation of the Normalized Google Distance [6] into a word relatedness measure based on Web search engine.

Some shortcomings of statistical based approaches are as follows. Without the help of human knowledge, the statistical based approaches actually regard the words in all documents as the same meaning when calculating one word's statistical information. This will lead to the ineffectiveness when measuring two low frequent words' semantic relatedness. In addition, these approaches also depend on the effectiveness and efficiency of the Web search engine.

3 Methodology

In recent years, the amount of structured data available on the Web has been increasing rapidly, making it possible to propose new ways to address complex information needs by combining data from different sources. LOD is aimed to link the existing data sources using RDF, and by September 2010, 203 data sources in different domains consisting of over 25 billion RDF triples have been

added into LOD cloud. This gives us an inspiration to measure named entities semantic relatedness based on LOD. As LOD consists of lots of data sources from different domains, by leveraging LOD, the named entity coverage problem can be overcome. And it gives us a possible solution to synthesize multi-sources. While the statistical based approaches regard named entities which have the same name in all documents as the same entity, LOD represents them as different entities. So even the low frequent entity can have its own description, which can be distinguished from other entities of the same name.

Figure 1 shows the architecture of our approach LODDO, which measures named entities semantic relatedness based on LOD. There are two components in the architecture: offline and online components. The offline component is aimed to build an index from the various LOD sources which can be used to find the entities corresponding to a specific entity name fleetly. For the online component, the Description Retrieval can retrieve all the description information of a given entity name from data sources by leveraging LOD Index. The Description Overlap Measuring uses the description information of two named entities to calculate the semantic relatedness between them.

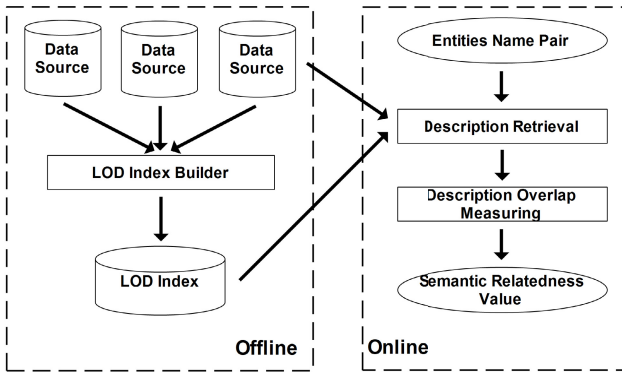


Fig. 1. The Architecture of LODDO

3.1 LOD Index Builder

LOD uses RDF, which is a generic, graph based data framework that represents information based on triples of the form $(subject, predicate, object)$, to organize the data. An entity can be either a subject or an object of any RDF triple. LOD identifies an entity via a HTTP scheme based Uniform Resource Identifier (URI). The URI does not only serve as a unique global identifier but it also provides access to a structured data representation of the identified entity.

It is not trivial to find the entities which have a specific name directly. For example, two data sources, even the same data source, may represent one target entity by different uris. So it becomes very important to find all the uris which mean the same entity. Moreover, some name variants may correspond to one entity, which

is ineffectively solved just by leveraging the string similarity. To solve these problems, we leverage the name properties, uri format and certain relationships in LOD to enumerate all possible name variants and uris to an entity, which can be represented as an entity triple ($entity_id, uri_set, name_set$). Here, $entity_id$ is an automatic generated University Unique Identifier (UUID) of an entity.

3.1.1 Name Extraction for URI

Thanks to the broad coverage of LOD, most name variants of an entity can be discovered by mining the diverse data sources. In this subsection, we focus on the name extraction for each uri. Unfortunately, different data sources may have different representation for names of an entity. A predicate may be used in different ways in different sources. For example, in RDF schema, the predicate `rdfs:label` is defined to provide a human-readable version of a resource's name. However, DBpedia uses it in a different way. Here is an example of `rdfs:label` in DBpedia:

`(dbpedia:3:The_World_Health_Organization, rdfs:label, "The")`.

Obviously it is not right to regard "The" as the name. Therefore, we need to analyze the LOD data sources respectively and identify the ways that may describe the name information. In such a way, we can get all the name variants of a uri and by automatically generating unique entity id corresponding to the uri, we get the initial entity triple space Γ .

$$\Gamma = \{(entity_id, uri, name_set) \mid uri \in LOD\} \quad (1)$$

Here we present the name schema of several data sources: DBpedia, Musicbrainz [21] (DBtune), and Freebase [4].

- For DBpedia, we find that there's no exact *predicate* which can show the name of a DBpedia uri. As a solution, we extract the name by deleting "-" and "(" components from the tail of the uri. For example:
`dbpedia:James_Sikes` has the name "James Sikes".
`dbpedia:Think_Again_(band)` has the name "Think Again".
- Musicbrainz (DBtune) represents a uri's name by *predicate*: `foaf:4:name`, `mo:5:title` and `skos:6:altLabel`.
- Freebase uses `fb:7:type.object.name` as the *predicate* of a uri's name.

3.1.2 Integrate Entity Triples

We have mentioned that different data sources, even the same data source, may represent one target entity by different uris in LOD. However, there exist some

² <http://www.w3.org/2000/01/rdf-schema#>

³ The `dbpedia:` stands for the prefix for URI from DBpedia.

⁴ <http://xmlns.com/foaf/0.1/>

⁵ <http://purl.org/ontology/mo/>

⁶ <http://www.w3.org/2004/02/skos/core#>

⁷ <http://rdf.freebase.com/ns/>

relationships connecting uris which are actually telling the same entity. We have identified three such relationships and make use of them to integrate the entity triples.

DBpedia:disambiguates Relationship. Disambiguation in DBpedia is the process of resolving the conflicts that arise when a name is ambiguous—when it refers to more than one topic covered by DBpedia. A disambiguation uri is linked with other different uris which have the same name. For example, there are two disambiguation triples in DBpedia:

$$\begin{aligned} &(dbpedia: Bell, dbpedia: disambiguates, dbpedia: Bell_Island) \\ &(dbpedia: Bell, dbpedia: disambiguates, dbpedia: Bell_Labs) \end{aligned}$$

which means *dbpedia: Bell_Island* has “Bell” and “Bell Island” as its name variants. And *dbpedia: Bell_Labs* has “Bell” and “Bell Labs” as its name variants.

Algorithm 1. Entity Triples Integration

Input: Initial entity triple $(entity_id, uri_set, name_set)$ space Γ got from subsection “Name Extraction for URI”; LOD triple $(subject, predicate, object)$ space Σ .

Output: Entity triple space Γ .

```

1: for all  $x$  in  $\Sigma$  do
2:   if  $x$  is a dbpedia:disambiguates or dbpedia:redirect triple then
3:      $et1 \leftarrow$  entity triple whose  $uri\_set$  contains  $x.subject$ 
4:      $et2 \leftarrow$  entity triple whose  $uri\_set$  contains  $x.object$ 
5:      $et2.name\_set = et1.name\_set \cup et2.name\_set$ 
6:   end if
7: end for
8: for all  $x$  in  $\Sigma$  do
9:   if  $x$  is a dbpedia:disambiguates or dbpedia:redirect triple then
10:     $et \leftarrow$  entity triple whose  $uri\_set$  contains  $x.subject$ 
11:     $\Gamma = \Gamma - et$ 
12:  else if  $x$  is a owl:sameAs triple then
13:     $et1 \leftarrow$  entity triple whose  $uri\_set$  contains  $x.subject$ 
14:     $et2 \leftarrow$  entity triple whose  $uri\_set$  contains  $x.object$ 
15:     $entity\_id \leftarrow$   $UUID\_Generation()$ 
16:     $\Gamma = \Gamma \cup \{(entity\_id, et1.uri\_set \cup et2.uri\_set, et1.name\_set \cup et2.name\_set)\}$ 
17:     $\Gamma = \Gamma - et1$ 
18:     $\Gamma = \Gamma - et2$ 
19:  end if
20: end for

```

DBpedia:redirect Relationship. DBpedia may use a redirect relationship to link one uri, which has no description, to another uri which has a description. The reasons for creating and maintaining such a schema include: alternative names, alternative spellings or punctuation, abbreviations, etc [2]. If uri_1 redirects to uri_2 , the uri_2 should also have the name of uri_1 as its name variant. For example, we have such a triple in DBpedia:

$$(dbpedia: UK, dbpedia: redirect, dbpedia: United_Kingdom)$$

which means *dbpedia:United_Kingdom* has “UK” and “United Kingdom” as its name variants.

Owl:sameAs Relationship. By common agreement, Linked Data publishers use the link type owl:sameAs to state that two URI aliases refer to the same resource. Therefore, if uri_1 owl:sameAs uri_2 , their entity triples should be integrated.

If two uris have an owl:sameAs relationship, their uris and names will be integrated to the same *entity_id*. For *dbpedia:disambiguates* and *dbpedia:redirects* relationships, we just integrate their names excluding uris. The detail algorithm of Entity Triples Integration is shown in Algorithm 1. And the time complexity is $O(|\Sigma|)$.

3.1.3 Index Storage

After getting all the entity triples, we need a mechanism to store and index them in order to guarantee the efficient retrieval for online semantic relatedness measuring. Considering the existence of one word’s different formats, such as *apple* and *Apples* which may indicate to the same entity, we need to normalize the names at first. The rules are as follows: (1) convert the names to lowercase; (2) perform word stemming on the names; (3) remove any articles from names.

Then, the inverted list is utilized to store such information. The storage mechanism is shown in Figure 2 and corresponding notation description is in Table 1.

After Entity Triple Integration, all the name variants and uris of an entity are extracted which means that the challenge, different representations of a single entity, has been solved. By using the LOD Index, we can find all the entities of a given name, which means that the word ambiguity challenge also has been solved.

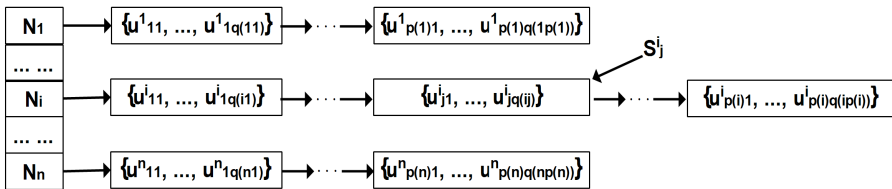


Fig. 2. LOD Index mechanism

3.2 Semantic Relatedness Measuring

Given an entity name, normalization of the name should be processed at first. Then we can retrieve all the entities with such a normalized name variant by leveraging the LOD Index. As there is a large variety of description about an entity in LOD, the heuristics arises that the more common description two entities have, the stronger semantic relatedness they have. In the following section, we will describe Description Retrieval and Description Overlap Measuring in detail.

⁸ <http://www.w3.org/2002/07/owl#>

Table 1. Notations for (Figure 2)

Notation	Description
N_i	name string
S_j^i	j th entity with a name variant of N_i
u_{jk}^i	k th uri which indicates to the S_j^i entity
n	the whole number of name strings
$p(i)$	number of entities with N_i as its name variant
$q(ij)$	number of uris corresponding to entity S_j^i

3.2.1 Description Retrieval

Since an entity is represented as a set of uris, the description of the entity can be constructed by accumulating the description of the uris in the *uri_set*. The description of a uri is defined as a vector of subjects and objects which forms RDF triples with the uri. In a LOD triple, if uri_i is the subject, then the object should be inserted into the description of uri_i . Otherwise, if uri_i is the object, the subject should be inserted into uri_i 's description. In LOD, an entity uri may have types in all probability. However, there exist some type assertions which are too loose. For example, almost every entity uri in DBpedia has a type of *owl:Thing*. So for avoiding such noise in LOD, we ignore the type assertion when generating the description.

3.2.2 Description Overlap Measuring

Having the heuristics that two related named entities may have many common related things, we leverage the LOD Description Overlap, named as LODDO, to calculate the semantic relatedness between two named entities.

In the real world, there exists such a situation: entity p has many related entities including entity q which leads to a weak semantic relatedness between p and q , however q only has few related entities including p which leads to a stronger semantic relatedness. So, it becomes an issue about how to determine the final semantic relatedness between p and q . Having such a puzzle, we use the following two strategies to determine the final semantic relatedness.

(1) LODJaccard: Consider equally to both named entities when measuring the semantic relatedness. It is defined as follows:

$$\begin{aligned}
 CommonDescription(p, q) &= |Description(p) \cap Description(q)| \\
 Denominator(p, q) &= |Description(p)| + |Description(q)| \\
 &\quad - |Description(p) \cap Description(q)| \quad (2) \\
 LODJaccard(p, q) &= \frac{CommonDescription(p, q)}{Denominator(p, q)}
 \end{aligned}$$

(2) LODOverlap: Have a bias towards the less description named entity when measuring the semantic relatedness. It is defined as follows:

$$\begin{aligned}
CommonDescription(p, q) &= |Description(p) \cap Description(q)| \\
Denominator(p, q) &= \min(|Description(p)|, |Description(q)|) \\
LODOverlap(p, q) &= \frac{CommonDescription(p, q)}{Denominator(p, q)}
\end{aligned} \tag{3}$$

where $Description(p)$ means the description of entity p .

Table 2. Four strategies to measure LOD Description Overlap

label	strategy name	description
1	LODJaccard_L	Choose LODJaccard to determine semantic relatedness. And choose largest LODJaccard to deal with multi-pairs problem.
2	LODOverlap_L	Choose LODOverlap to determine semantic relatedness. And choose largest LODOverlap to deal with multi-pairs problem.
3	LODJaccard_LC	Choose LODJaccard to determine semantic relatedness. And choose largest CommonDescription to deal with multi-pairs problem.
4	LODOverlap_LC	Choose LODOverlap to determine semantic relatedness. And choose largest CommonDescription to deal with multi-pairs problem.

As there may be several entities which have the same name variant, given two entity names, multi-pairs may be generated. So we should determine which two entities should be chosen to calculate the semantic relatedness. Because of the lack of context around the given entity names, we should choose the entities pair which is mostly in agreement with usual human sense. There are two strategies: (1) Largest LODJaccard or LODOverlap: Choose the pair which has the strongest semantic relatedness. This strategy has been adopted by many semantic relatedness measuring approaches, such as [10,18]. (2) Largest Common-Description: Choose the pair which has the most abundant related things in common. If several pairs have the same largest CommonDescription, the smallest Denominator will be chosen.

Assume m means the entity number of p , n means the entity number of q , ap means the average size of p 's description, aq means the average size of q 's description. Then the time complexity of Description Overlap Measuring is $O(m \times n \times (ap + aq))$. $(ap + aq)$ means the time complexity of $CommonDescription(p, q)$.

All in all, four strategies can be used to deal with the semantic relatedness between two named entities. They are described detailedly in Table 2. An experimental study is provided in Section 4 to compare the four strategies.

4 Experiments

In this section, we conducted some experiments to demonstrate the effectiveness of our proposed approach in named entities semantic relatedness measuring. The

experiments results showed that our approach greatly outperformed the previous semantic relatedness measuring approaches. Extensive experiments were also carried out to prove the robustness of our approach.

4.1 Experimental Setup

4.1.1 LOD Data Sources

In our work, we randomly select two cross-domain data sources: DBpedia, Freebase, and a specific-domain data source: Musicbrainz (DBtune). In our future work, we will consider other domains and do more comprehensive experiments. we have generated a LOD Index which includes DBpedia, Musicbrianz (DBtune) and Freebase. The scale statistics are shown in Table 3.

Table 3. LOD scale statistics

Data Source	DBpedia	Musicbrainz (DBtune)	Freebase
Entity Number (million)	3.9	23.2	29

From Table 3 we find that the entity number of Musicbrainz (DBtune) and Freebase exceeds DBpedia greatly. As DBpedia is extracted from Wikipedia, and Wikipedia has a larger coverage than WordNet, we can conclude that LOD does enlarge entity coverage tremendously than Wikipedia and WordNet. So by leveraging LOD, the entity coverage problem, which appears in traditional knowledge based approaches, can be solved.

4.1.2 Evaluation Measure

There are two different correlation measures which have been used for evaluating semantic relatedness measuring. The Pearson product-moment correlation coefficient γ is to correlate the scores computed by a semantic relatedness measuring approach with the numeric judgements of semantic relatedness provided by humans. The Spearman rank order correlation coefficient ρ is to correlate named entities pair rankings. Zesch [23] compared these two measures and recommended to use Spearman rank correlation to evaluate semantic relatedness measuring. So in our experiments we just leverage Spearman rank correlation ρ as the evaluation measure.

4.1.3 Dataset

Unfortunately, there is no benchmark data set for named entities semantic relatedness measuring. In our experiment, we make our own data set and offer it as a standard for testing named entities semantic relatedness. In our work, we have generated a LOD index which includes DBpedia, Musicbrianz (DBtune) and Freebase. Musicbrainz (DBtune) mainly focuses on the music domain while DBpedia and Freebase are cross-domain data sources. we randomly select 60 music related entities pairs from last.fm⁹ and 60 other domains entities pairs from Wikipedia, giving a total of 120 pairs of named entities.

⁹ <http://www.last.fm/>

In the evaluation work, the semantic relatedness of each pair is rated by six subjects with the following instructions:

Indicate how strongly these named entities are related using integers from 0 to 4. The description and an example corresponding to each number are given as follows, and if you think some pairs fall in between two of these categories you must push it up or down (no halves or decimals).

- 0: not at all related; “Linux” and “Beijing”
- 1: vaguely related; “China” and “Tokyo”
- 2: indirectly related; “Backstreet Boys” and “Britney Spears”
- 3: strongly related; “Backstreet Boys” and “As Long as You Love Me”
- 4: inseparably related; “Gate of Heavenly Peace” and “Tiananmen”

The named entities pairs were sorted in descending order by average score, and 100 pairs were selected in order to balance the rate distribution from 0 to 4. The average Spearman rank correlation ρ among these six subjects is 0.9617, which means the rate result is objective. Moreover, 0.9617 can also be used as the upper bound of the performance.

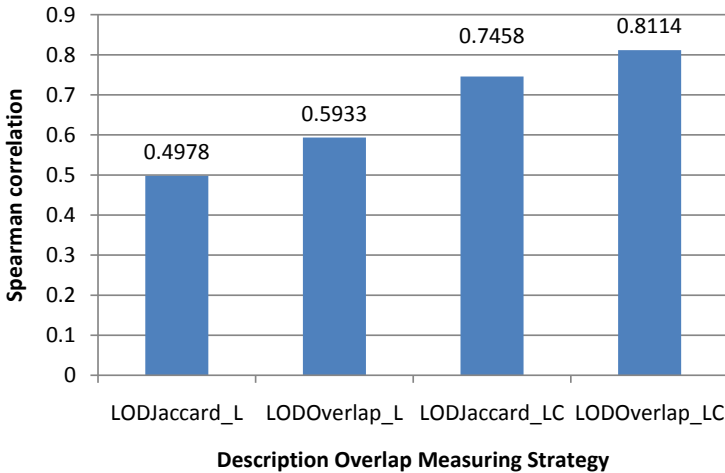


Fig. 3. Four Description Overlap Measuring strategies’ performance; Spearman rank correlation ρ with humans

4.2 Description Overlap Strategy Comparison

In this section, we compare the performance of the four strategies in Description Overlap Measuring. The results are shown in Figure 3.

From Figure 3, we can see that LODOverlap_L outperforms LODJaccard_L, LODOverlap_LC outperforms LODJaccard_LC. This tells us that when dealing with the semantic relatedness between named entities, it is more reasonable to

focus on the less description named entity. From the results, we also find that `LODOverlap_LC` is better than `LODOverlap_L`, `LODJaccard_LC` is better than `LODJaccard_L`. It is mainly caused by the noise in LOD when handling multi-pairs problem. In LOD, there exist some obsolete and incomplete uris. They have little and even wrong description which will lead to high overlap between two unrelated named entities and thus reduce the performance. Leveraging the largest common description pair has two advantages: (1) The largest common description pair is probably well described in LOD, which can reduce the influence of noise in LOD; (2) It is more likely to have an objective semantic relatedness which conforms to the human sense.

In the following experiments, we choose `LODOverlap_LC` as the strategy of our approach LODDO. Table 4 shows some result examples of LODDO.

Table 4. Result examples of LODDO

Named Entities pair	LOD Description Overlap
" <i>Gate of Heavenly Peace</i> " and " <i>Tiananmen</i> "	1
" <i>Backstreet Boys</i> " and " <i>As Long as You Love Me</i> "	0.3758
" <i>Backstreet Boys</i> " and " <i>Britney Spears</i> "	0.1538
" <i>China</i> " and " <i>Tokyo</i> "	0.0556
" <i>Linux</i> " and " <i>Beijing</i> "	0.0047

4.3 Semantic Relatedness Measuring Performance

Six previous semantic relatedness approaches are used to compare with our proposed approach.

- Rad [17] regards WordNet as a graph: concepts as vertexes and all types of relationships as edges. Given two concepts, the semantic relatedness is represented by the shortest path length between them, the larger path length, the weaker semantic relatedness between them.
- GlossOverlap [20] calculates the text overlap of two concepts' glosses, which are the first paragraph of their Wikipedia articles, to measure the semantic relatedness. GlossOverlap is defined as follows:

$$GlossOverlap(p, q) = \tanh \left(\frac{overlap(Gloss(p), Gloss(q))}{length(Gloss(p)) + length(Gloss(q))} \right) \quad (4)$$

- Intrinsic Information Content (IIC) [16] applies an intrinsic information content measure relying on the hierarchical structure of the Wikipedia category tree. It's defined as follows:

$$IIC(p, q) = 1 - \frac{\log(hypo(lcs(p, q)) + 1)}{\log(C)} \quad (5)$$

where $lcs(p, q)$ means the least common subsumer of p and q in Wikipedia category tree. $hypo(lcs(p, q))$ is the number of hyponyms of node $lcs(p, q)$ and C equals the total number of conceptual nodes in the hierarchy.

- ESA [7] firstly constructs weighted vector of Wikipedia concepts to each input text. Then to compute semantic relatedness of this pair of text, it compares their vectors using the cosine metric.
- WebJaccard and WebOverlap [5] are two popular co-occurrence measures to compute semantic similarity using page counts. They are defined as follows:

$$WebJaccard(p, q) = \frac{H(p \cap q)}{H(p) + H(q) - H(p \cap q)} \quad (6)$$

$$WebOverlap(p, q) = \frac{H(p \cap q)}{\min(H(p), H(q))} \quad (7)$$

Here $H(p)$ denotes the page counts for the query p in a search engine. In our experiment, we choose Google¹⁰ to get page counts.

Figure 4 shows the results of these approaches on the test dataset.

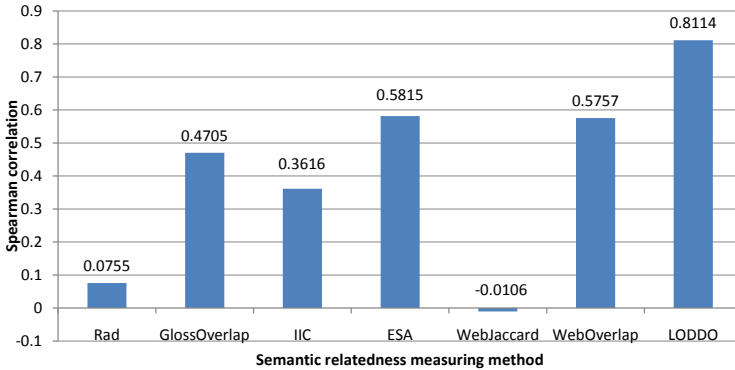


Fig. 4. Different approaches' performance; Spearman rank correlation ρ with humans

From Figure 4, we can find that our proposed approach significantly improves the performance of named entities semantic relatedness measuring. Even compared with ESA, the second best performance, we get an improvement of 39.6%. As WordNet has limited entity coverage and 75 pairs in the test dataset cannot be measured because of the miss-hit in WordNet, Rad achieves a low Spearman rank correlation. ESA, GlossOverlap and IIC obtain a better performance than Rad, because Wikipedia has a larger coverage and richer description than WordNet. In Wikipedia, only 6 pairs in the test dataset is miss-hit. However, ESA considers the words in a name independently, thus may misunderstand the meaning of the name. GlossOverlap regards the uncritical words equally to the critical words in the gloss, thus the effectiveness may be reduced by the uncritical words. Since IIC only takes into account the category hierarchy relation without considering other meaningful relations, the performance is limited. WebJaccard

¹⁰ <http://www.google.com>

and WebOverlap use the Google search statistical information to measure the semantic relatedness between named entities. As they regard a name in all documents as the same meaning, the effectiveness can be reduced. Since WebJaccard considers the two named entities equally, the larger hit entity brings more noise which influences the accuracy greatly. Furthermore, WebOverlap provides a better performance than WebJaccard, which proves the heuristics that the semantic relatedness should bias the less description entity.

4.4 LOD Data Source Selection

In this section, the influence of selecting different LOD data source is figured out. What will the performance change if we merge the data sources rather than use them singly. Table 5 gives the results of using different data sources.

Table 5. Performance of selecting different data sources

Data Source	average description number	missed pairs number	Spearman rank correlation ρ with humans
Musicbrainz (DBtune)	35.79	26	0.0128
Freebase	10468.4	16	0.4217
DBpedia	11658.5	6	0.7668
Musicbrainz (DBtune) & Freebase & Dbpedia	26076.1	0	0.8114

It is noted that the Spearman rank correlation is calculated without the consideration of the pairs which can't be found in corresponding data sources. There are two reasons why Musicbrainz (DBtune) gets such a low performance: (1) The description of an entity is insufficient (only 35.79 descriptions on average), compared with other data sources (more than 10k descriptions on average); (2) The entity corresponding to a name in Musicbrainz (DBtune) sometimes is not the sense in our daily experience, for example "Ferrari" is a song in Musicbrainz (DBtune) rather than automotive in common sense. From the column "*missed pairs number*" we can know that the use of single data source also leads to entity coverage problem, however, by merging the data sources together, the coverage problem can be relieved. Although the average description number of Freebase and DBpedia are similar, their performances are different. So we can conclude that different data sources may have different constructions and qualities, which contributes to the different semantic relatedness measuring performances. In addition, having more description is likely to lead to better performance. It verifies that with more data sources, the performance can be improved steadily, which proves the robustness of our approach.

5 Conclusion

In this paper, we target on the task of named entities semantic relatedness measuring. As the existing knowledge based approaches have the entity coverage issue and the statistical based approaches have unreliable result to low frequent entities, we propose a more comprehensive approach by leveraging LOD to solve these problems. By exploiting the semantic associations in LOD, we propose a novel algorithm, called LODDO, to measure the semantic relatedness between named entities. Specifically, we first propose a mechanism to index the various LOD sources which can be used to find the entities corresponding to a specific entity name fleetly. Then, we bring forward LOD Description Overlap to measure the named entities semantic relatedness. The experimental results show that our approach greatly outperforms the previous semantic relatedness measuring approaches. And it is robust to leverage more data sources in LOD and provide better performance.

In the future, we plan to investigate more data sources from LOD in order to extend the coverage and promote the performance. We will investigate the quality of LOD and see how it will influence the performance of semantic relatedness measuring between named entities. We will also try to find a uniform approach to measure the semantic relatedness between abstract concepts and named entities.

References

1. Linked open data, <http://linkeddata.org>
2. Wikipedia:redirect, <http://en.wikipedia.org/wiki/Wikipedia:Redirect>
3. Banerjee, S., Pedersen, T.: An Adapted Lesk Algorithm for Word Sense Disambiguation using Wordnet. In: Gelbukh, A. (ed.) CILCling 2002. LNCS, vol. 2276, pp. 136–145. Springer, Heidelberg (2002)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: The 2008 ACM SIGMOD International Conference on Management of Data, New York, USA, pp. 1247–1250 (2008)
5. Bollegala, D., Yutaka, M., Ishizuka, M.: Measuring semantic similarity between words using web search engines. In: The 16th International Conference on World Wide Web, New York, NY, USA, pp. 757–766 (2007)
6. Cilibrasi, R., Vitanyi, P.M.B.: The google similarity distance. *IEEE Trans. Knowledge and Data Engineering* (3), 370–383 (2007)
7. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: The 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India, pp. 1606–1611 (2007)
8. Gracia, J., Mena, E.: Web-Based Measure of Semantic Relatedness. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 136–150. Springer, Heidelberg (2008)
9. Hirst, G., St-Onge, D.: Lexical chains as representation of context for the detection and correction malapropisms. In: *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, pp. 305–332. MIT Press (1998)

10. Jarmasz, M., Szpakowicz, S.: Roget's thesaurus and semantic similarity. In: *Recent Advances in Natural Language Processing*, pp. 212–219 (2003)
11. Lesk, M.: Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: *The 5th Annual International Conference on Systems Documentation*, Toronto, Canada, pp. 24–26 (1986)
12. Miller, G.A.: Wordnet: A lexical database for english. *Communications of the ACM* (11), 39–41 (1995)
13. Milne, D., Witten, I.H.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: *The AAAI 2008 Workshop on Wikipedia and Artificial Intelligence (WIKIAI 2008)*, Chicago, IL (2008)
14. Patwardhan, S., Banerjee, S., Pedersen, T.: Using Measures of Semantic Relatedness for Word Sense Disambiguation. In: Gelbukh, A. (ed.) *CICLing 2003*. LNCS, vol. 2588, pp. 241–257. Springer, Heidelberg (2003)
15. Patwardhan, S., Pedersen, T.: Using wordnet based context vectors to estimate the semantic relatedness of concepts. In: *The EACL 2006 Workshop Making Sense of Sense - Bringing Computational Linguistics and Psycholinguistics Together*, Trento, Italy, pp. 1–8 (2006)
16. Ponzetto, S.P., Strube, M.: Knowledge derived from wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 181–212 (2007)
17. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics* (1), 17–30 (1989)
18. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: *The 14th International Joint Conference on Artificial Intelligence*, San Francisco, CA, USA, pp. 448–453 (1995)
19. Spanakis, G., Siolas, G., Stafylopatis, A.: A hybrid web-based measure for computing semantic relatedness between words. In: *The 2009 21st IEEE International Conference on Tools with Artificial Intelligence*, Washington, DC, pp. 441–448 (2009)
20. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: *The 21st National Conference on Artificial Intelligence*, Boston, MA, pp. 1419–1424 (2006)
21. Swartz, A.: Musicbrainz: A semantic web service. *IEEE Intelligent Systems* (1), 76–77 (2002)
22. Wubben, S., van den, B.: semantic relatedness metric based on free link structure. In: *The Eighth International Conference on Computational Semantics*, Tilburg, The Netherlands, pp. 355–358 (2009)
23. Zesch, T., Gurevych, I.: Wisdom of crowds versus wisdom of linguists measuring the semantic relatedness of words. *Natural Language Engineering* (1), 25–59 (2010)

What Should I Link to? Identifying Relevant Sources and Classes for Data Linking

Andriy Nikolov, Mathieu d’Aquin, and Enrico Motta

Knowledge Media Institute, The Open University, Milton Keynes, UK
{a.nikolov,m.daquin,e.motta}@open.ac.uk

Abstract. With more data repositories constantly being published on the Web, choosing appropriate data sources to interlink with newly published datasets becomes a non-trivial problem. It is necessary to choose both the repositories to link to and the relevant subsets of these repositories, which contain potentially matching individuals. In order to do this, detailed information about the content and structure of semantic repositories is often required. However, retrieving and processing such information for a potentially large number of datasets is practically unfeasible. In this paper, we propose an approach which utilises an existing semantic web index in order to identify potentially relevant datasets for interlinking and rank them. Furthermore, we adapt instance-based ontology schema matching to extract relevant subsets of selected data source and, in this way, pre-configure data linking tools.

1 Introduction

The principles of Linked Data¹ recommend data publishers to reuse exiting URIs for their entities, where possible, or to provide links to them. In this way, more information can be obtained by following the links. In order to achieve that, data publishers face two non-trivial problems. First, they must be able to find existing data sources which can be reused or linked to. For this, they must be aware of the content of existing repositories describing relevant domains and be able to assess their suitability. Second, they must configure and run data linking tools which would discover mappings between individuals in their dataset and the chosen external ones.

With the growing number of repositories published based on the Linked Data principles, identifying relevant datasets and resources can become problematic. As a result, data publishers usually only link their datasets to the popular repositories (such as DBpedia² and Geonames³). This may not always be the optimal solution in some cases, for example:

- If the data domain is highly specialised and not covered by popular repositories in sufficient details.

¹ <http://www.w3.org/DesignIssues/LinkedData>

² <http://dbpedia.org>

³ <http://www.geonames.org/>

- If different parts of the dataset are covered by several external repositories: e.g., when a repository contains references to scientific publications both on computer science (described by DBLP⁴) and medicine (described by PubMed⁵).

To support assessment of different sources, catalogs of Linked Data repositories are maintained (e.g., in CKAN⁶), and meta-level descriptors of repositories are provided using the VoiD vocabulary⁷. However, these sources can still be insufficient as they do not take into account the distribution of instances in repositories. For example, several repositories contain information about academic researchers, however, they use different criteria to include individuals: e.g., DBpedia only mentions the most famous ones, DBLP only includes Computer Science researchers, and RAE⁸ deals with researchers working in UK institutions. In order to be able to choose the most appropriate repositories to link to, one must have access to complete instance-level data stored in them. Obtaining these data directly from the data sources and analysing them is often not feasible due to the size of datasets which need to be downloaded.

Once the dataset is chosen, the second challenge is to configure the data linking tool which would discover actual links between individuals in two repositories. The configuration typically includes choosing the selection criterion for determining potential matching candidates and the matching function, which would determine whether a pair of matching candidates actually represent the same entity. Both choices heavily depend on the structure of data in the external data repository.

In this paper we describe an approach which helps to solve these tasks without the need to process complete external datasets. The approach involves two methods, which we consider our contribution:

- Identifying and ranking relevant candidate data repositories for linking. To achieve this, the method utilises keyword-based search over existing semantic web index, integrates search results, and analyses them.
- Identifying relevant classes containing potentially matching individuals in chosen external sources. To this end, the method adapts and extends instance-based ontology matching techniques. We define the task of finding the best matching class in an external dataset and evaluate the suitability of different instance-based similarity metrics to the task.

The rest of the paper is organized as follows. Section 2 outlines the use case which provided the main motivation for this work. Section 3 describes the method for selecting and ranking the data sources. Section 4 focuses on application of instance-based ontology matching techniques in order to determine a relevant

⁴ <http://dblp.l3s.de/>

⁵ <http://www.ncbi.nlm.nih.gov/pubmed/>

⁶ <http://ckan.net/> see <http://ckan.net/group/lodcloud>

⁷ <http://semanticweb.org/wiki/VoiD>

⁸ <http://rae2001.rkbexplorer.com/>

subset of the chosen data source. Section 5 discusses the results of the experiments we performed to test our algorithms. Finally, section 7 concludes the paper.

2 Motivation

The problem of determining a set of relevant repositories is a generic one and can occur in different contexts. Our work was primarily motivated by two use cases: the SmartProducts project and development of the *data.open.ac.uk* repository.

2.1 Scenarios and Requirements

One of the tasks within the SmartProducts project⁹ involves reusing the data from external semantic repositories to build knowledge bases for smart consumer devices: e.g., to extend the core domain knowledge base of food recipes for a smart kitchen with nutritional data, alternative recipes, health profiles of food products, etc. In order to extend the core domain knowledge base, the developer has to be able to find relevant repositories on the Web of Data and interlink them with this core knowledge base².

In another scenario, the *data.open.ac.uk* repository¹⁰ aims at publishing various data related to the activities of The Open University (OU)¹¹ according to Linked Data principles. These datasets include, among others, the publications originated by OU researchers, courses provided by the university, etc. Many entities referenced in these datasets are also mentioned in other public repositories. Thus, in order to facilitate data integration, it makes sense to create links from instances used in the *data.open.ac.uk* datasets to external semantic data stores. Given the range of categories to which data instances belong, it is difficult to select a single external source to link to: e.g., publication venues can be linked to different subsets of RKBExplorer, DBLP, PubMed, DBPedia, or Freebase¹². Moreover, the repository is constantly extended with more instance data for existing topics (e.g., as more research output is published with time) as well as with more topics (as more internal datasets are released online). Selecting relevant sources for linking and selecting specific individuals to link to within these sources becomes a time-consuming procedure, which needs to be automated as much as possible.

There are several factors which can guide the selection of the repository for linking, in particular:

- *Degree of coverage.* In order to maximise the possibility to reuse external descriptions, the sources which contains more references to the entities stored in the newly published repository are preferable.

⁹ <http://www.smartproducts-project.eu>

¹⁰ <http://data.open.ac.uk>

¹¹ <http://www.open.ac.uk>

¹² <http://www.freebase.com/>

- *Additional information provided by the source.* When selecting a source to link to, it is important to take into account how much additional information about entities is provided by each external source: i.e., what properties and relations are used to describe these entities.
- *Popularity of the source.* Linking to URIs defined in a popular data source or reusing them makes it easier for external developers to find the published data and use them.

Among these factors, only the degree of coverage heavily relies on instance-level data stored in external repositories. The level of detail of instance descriptions can be obtained from the domain ontology used by the external dataset and, possibly, a few example instances, while the popularity of the source can be estimated based on VoiD linkset descriptors. Therefore, when designing our algorithm, we primarily focused on estimating the degree of coverage between the internal dataset prepared for publishing and potentially relevant external datasets.

2.2 Overview of the Approach

The task of finding relevant repositories assumes that there is a dataset to be published $D_s = \{O_s, I_s\}$ containing a set of individuals I_s structured using the ontology O_s . We will refer to this dataset as the *source dataset*. Each individual belongs to at least one class c_λ defined in O_s : $I = \{i_j | c_\lambda(i_j), c_\lambda \in O_s\}$. On the Web there is a set of Linked Data repositories $\{D_1, \dots, D_n\}$ such that $D_j = \{O_j, I_j\}$. There is a subset of these repositories $\{D_1, \dots, D_m\}$ which overlap with D_s , i.e., $\forall (j \leq m) \exists (I_j^O \subseteq I_j) : I_j^O = \{i_k | equiv(i_k, i_s), i_j \in I_j, i_s \in I_s\}$, where *equiv* denotes the relation of equivalence between individuals. The meaning of the equivalence relation here depends on the identity criterion chosen by the data publisher: e.g., *owl:sameAs* links or direct reuse of URIs assume that URIs must be strictly interchangeable (see [4] for the analysis of different types of identity). The goal is to identify the subset of relevant repositories $\{D_1, \dots, D_m\}$ and to rank them according to the degree of coverage $|I_j^O|/|I_s|$. Given that the publisher may want to select different repositories to link for different categories of instances in D_s , then for each class $c_\lambda \in O_s$ a separate ranking should be produced based on the degree of coverage for instances of this class $|I_{j\lambda}^O|$, where $I_{j\lambda}^O = \{i_k | equiv(i_k, i_s), i_s \in I_s, c_\lambda(i_s)\} \subseteq I_j^O$.

Since the actual discovery of links is usually performed by an automated tool (such as Silk [14] or KnoFuss [10]), another important task is to restrict the search space for this tool by identifying in each dataset D_j a set of relevant classes c_{jk} which contain potentially overlapping individuals with c_λ . Then the tool can be configured to select only individuals of these classes as candidates for linking. The main obstacle with these tasks is the need to identify the overlapping subset of instances $|I_j^O|$ from each external dataset. Downloading whole datasets or applying data linking tools to their complete sets of instances is often unfeasible due to their size and required computational time, network load, and local disk space.

In order to minimize the amount of data from external repositories which must be processed locally, we adopted a two-stage approach. At the first stage, a semantic web index which supports keyword-based search for data instances is utilised to propose and rank relevant sources as well as potentially relevant classes. This solution, which extends the earlier version of the algorithm presented in [8], is described in section 3. Once a source is selected, the second stage involves finding the relevant classes which would facilitate the data linking process. At this stage, partial information is retrieved from the selected source, in particular, labels of instances of the candidate classes (see section 4).

3 Selecting Relevant Data Sources Using Keyword Search Services

We assume that a semantic keyword search service takes as its input a set of keywords $K = \{k_1, \dots, k_i\}$. As output, it returns a set of potentially relevant individuals which may belong to different repositories: $I^{res} = I_1^{res} \cup I_2^{res} \cup \dots \cup I_m^{res}$, where $I_j^{res} \subseteq I_j$. For returned individuals $i_{jk} \in I_j^{res}$, their types $\{c_{jk\lambda} | c_{jk\lambda}(i_{jk})\}$ are also available in the search results. An example of the search service which satisfies this assumption is Sig.ma [12], which uses Sindice as its search index.

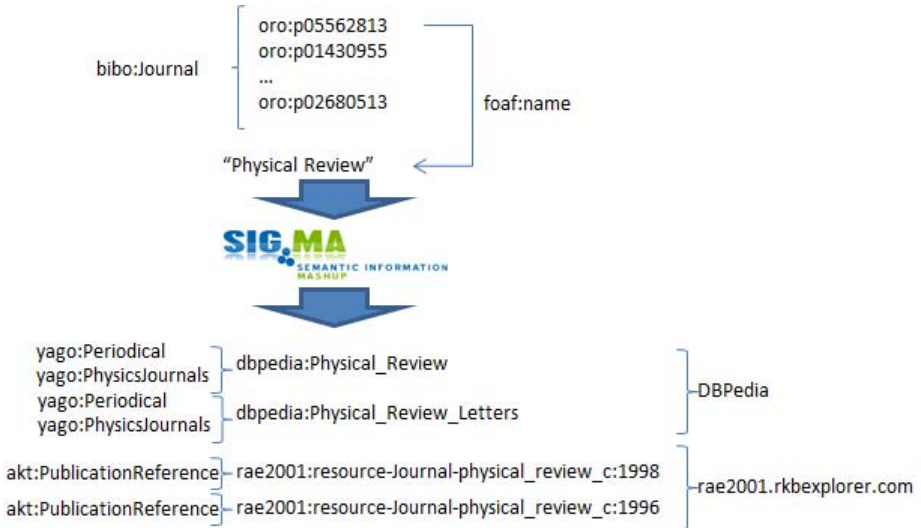


Fig. 1. Keyword-based search for relevant individuals

3.1 Finding Potentially Relevant Sources

In order to find potentially relevant individuals from the source dataset D_s , we query the search service using the labels of individuals (values of *rdfs:label*,

foaf:name, *dc:title*, etc.) as keywords. Then, these query results are aggregated to estimate the degree of coverage of different data sources (Fig. 11). The procedure consists of the following steps:

1. Randomly selecting a subset of individuals I_s^* from D_s belonging to a class c_s . This is done in order to reduce the number of queries to the search service in case where the complete extension set of individuals is too large. On the other hand, the subset must be large enough to produce reliable ranking of sources.
2. Querying the search service (Sig.ma) for labels of each individual in the selected subset. The results of each search are returned as an RDF document, which includes the references to individuals, their sources, and the classes they belong to.
3. Aggregation of the search results. RDF documents returned by Sig.ma are loaded into a common repository, and the individuals i_{jk} are grouped according to their sources D_j .
4. Data sources are ranked according to the number of their individuals returned by the search service $|\{i_{jk} | i_{jk} \in D_j\}|$.

In our approach we assume that the relevance function used by the search service to select query answers serves as an approximation of the identity function *equiv*(*.*). In the general case, this is not true due to ambiguity of labels and the fact that search services may not always achieve 100% precision. Taking a sufficiently large subset of individuals to search makes it possible to reduce the impact of ‘false positives’ returned by the search engine.

After applying these steps to our test scenarios (see section 5), we found that the rankings obtained using this procedure are still likely to be imprecise for two main reasons:

- Inclusion of irrelevant sources. For individuals belonging to classes with highly ambiguous labels, many ‘false positives’ in the set of answers can result in irrelevant repositories achieving high ranking positions. For instance, when searching for specific subcategories of people, any source mentioning sufficiently large number of people would be considered relevant: e.g., Twitter and DBLP were highly ranked when searching for music contributors.
- Inclusion of irrelevant classes. Resulting sets often contained classes which would not allow selecting appropriate candidate individuals by a matching tool. Sometimes a generic superclass was ranked higher than the correct class: e.g., *dbpedia:Person* was ranked higher than a more relevant *dbpedia:MusicalArtist*. In other cases, completely irrelevant classes were included: e.g., for scientific journals the class *akt:Publication-Reference* describing specific volumes of journals was ranked higher than *akt:Journal*.

In order to overcome these issues, our approach includes the second stage: filtering of search results using ontology matching techniques.

3.2 Using Ontology Matching Techniques to Filter Out Irrelevant Results

In order to filter out irrelevant search results, our approach can utilise mappings between classes provided by existing schema matching tools (Fig. 2). In our experiments we utilised ontology mappings produced by two algorithms:

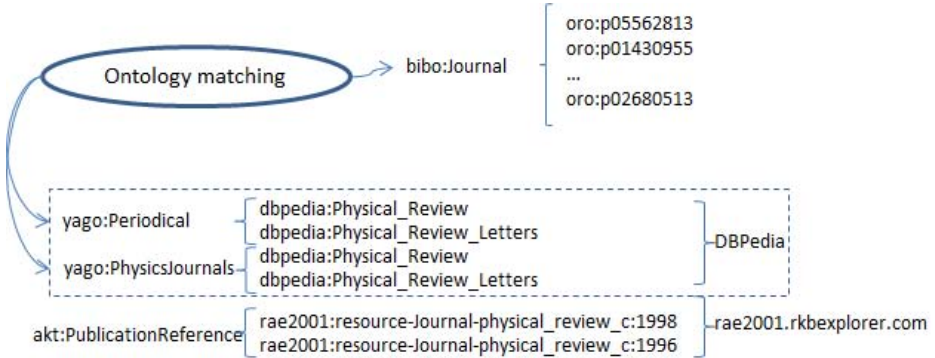


Fig. 2. Using ontology matching to refine search results

- CIDER [3] which takes as input two ontologies in RDF format and two URIs defining ontological terms from these ontologies and produces as output the similarity score between these terms. CIDER utilises evidence defined at the level of ontological schema: string similarity between class labels, semantic relations defined in WordNet and positions of classes in class hierarchies.
- Instance-based matching algorithm described in [9], which generated schema mappings between classes on the Web of Data based on their overlapping sets of instances. Overlapping sets of instances were inferred based on existing *owl:sameAs* relations between them published in the Billion Triple Challenge 2009 (BTC) dataset¹³. Resulting mappings represent subsumption relations of the form $c_A \sqsubseteq c_B$, where c_A and c_B belong to different ontologies.

As the first step of the filtering procedure, CIDER is applied to measure similarity between the class c_s in D_s , for which overlapping sources have to be found, and each of the classes c_{jkl} appearing in the aggregated search results. Then, a threshold is applied to filter out classes with low similarity scores. Remaining classes from the search results constitute the set of ‘confirmed’ classes $C_{confirmed}$. At the next stage, this set of ‘confirmed’ classes is enriched using the mappings obtained using instance-based matching. For each class $c_i \in C_{confirmed}$, all mappings from the BTC-based set where $c_A \sqsubseteq c_i$ are selected, and all c_A are added into $C_{confirmed}$. After the filtering stage, the datasets for which there is at least one ‘confirmed’ class are moved in the ranking above those for which no classes were confirmed.

¹³ <http://vmlion25.deri.ie/>

In our tests described in section 5, the filtering stage led to improved precision in the resulting ranking of data sources. However, the approach was found insufficient to deal with the actual task of finding relevant classes in target repositories. While the main problem with unfiltered results was the choice of too generic classes, the filtering procedure left out many relevant classes or chose too specific classes in the hierarchy. Because of this, the special method was implemented to identify the best-matching classes in the ontology of a given dataset.

4 Identifying Relevant Classes in the Dataset

Identifying a relevant repository for interlinking represents only the first stage of the process. In order to configure the data linking method and minimize the possible errors, it is important to select the relevant subset of instance containing potentially coreferent individuals. Selecting too broad subset can substantially increase the computational time required to compare irrelevant individuals and can also lead to many spurious mappings, thus reducing precision. Selecting too small subset, on the other hand, can lead to missing mappings and reduced recall. Given that the instances in semantic repositories are organised using ontological class hierarchies, selecting a relevant subset of data for interlinking requires selecting the best fitting class in the target ontology.

Definition 1: Let I_s represent a set of individuals from the source repository D_s . A subset of these individuals I_s^O has matching individuals in the target repository D_t : $\forall i_s \in I_s^O \exists i_t \in I_t^O : i_s \equiv i_t$. Then the best fitting class for I_s is such a class $c_x^{fit} \in D_t$ that it contains all individuals from I_t^O and there is no subclass $c_x \sqsubseteq c_x^{fit}$ that $\forall i_t \in I_t^O : i_t \in c_x$.

Assuming that all instances $i_s \in I_s$ belong to the same class c_s , the task of choosing a best-fitting class represents a special case of the ontology matching problem. However, it has several specific features [11]:

- It is possible that not all instances of the source class have matching counterparts in the target repository. Thus, the goal is to find a fuzzy ‘overlap’ relation between classes rather than strict logical equivalence or subsumption.
- Class definitions in the ontology can be insufficient to capture the intended meaning: e.g., the class *Actor* in *LinkedMDB* refers to any person participating in a movie, while *Actor* in *DBPedia* refers to professional actors (both film and stage).

Because of this, instance-based ontology matching techniques, which determine relations between classes based on the overlap between their instance sets, appear especially suitable for the task. However, these techniques cannot be directly reused: in the absence of mappings between individuals in two repositories, it is impossible to determine the overlap between their instance sets. Thus, applying instance-based ontology matching to the task of determining the most relevant class in the hierarchy requires dealing with two challenges:

- Approximating the power of the overlapping set of instances for two classes in the absence of actual instance mappings.
- Selecting a suitable set-based similarity measure, which can determine the degree of relevance of a particular class c_j for a set of instances I_s .

In order to estimate the power of the overlap relation between two classes, we use the same evidence as for the source ranking: keywords extracted from instance labels. The algorithm takes as its input the selected subset of individuals from the source dataset as well as the output of the source selection algorithm: the target repository D_t and the initial class c_t^{top} . As our tests have shown, the class c_t^{top} returned by the source selection procedure is usually too generic. The goal of the algorithm is to find the ‘best-fitting’ subclass $c_t^{fit} \sqsubseteq c_t^{top}$. The procedure consists of the following steps:

1. Create profiles of all instances $i_{tj} \in c_t^{top}$. A profile $P(i_{tj})$ of the instance i_{tj} includes all keywords extracted from the label of i_{tj} .
2. Randomly select a subset of individuals I_s^* from I_s .
3. For each individual $i_s \in I_s^*$, find the ‘best matching’ individual i_{tj}^+ using the cosine similarity between individual profiles: $i_{tj}^+ = \operatorname{argmax}(\operatorname{cosim}(P(i_s), P(i_{tj})))$. For each class c_{ti} such that $i_{tj}^+ \in c_{ti}$, increase the score $s(c_i)$.
4. For each subclass $c_{ti} \sqsubseteq c_t^{top}$, its score $s(c_{ti})$ serves as an estimation of the overlap $|c_s \cap c_{ti}|$. Based on this overlap estimation, the similarity between classes is calculated $\operatorname{sim}_i(|c_s \cap c_{ti}|) \approx \operatorname{sim}_i(s(c_{ti}))$. The class c_{ti} with the highest similarity degree is assigned as c_t^{fit} .

Obtaining the best matching target individual is implemented using a standard keyword-based search mechanism using an in-memory Lucene index. This procedure cannot be used as a replacement for the actual instance matching tools due to its low accuracy, but, given a sufficient sample size, it can approximate instance equivalence in order to produce schema-level links. At this stage, the sample from the first step of the algorithm (search for potentially relevant sources) can be reused.

In order to measure the actual similarity between two classes with overlapping sets of instances, several metrics have been used, in particular:

- The *Jaccard index* is defined as $JC(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}$. In [5] a modified version was proposed to give advantage to classes with large number of instances. This *corrected Jaccard index* is defined as $JC_{corr} = \frac{\sqrt{|I_1 \cap I_2| \times (|I_1 \cap I_2| - 0.8)}}{|I_1 \cup I_2|}$.
- *Overlap coefficient* is another set similarity metrics defined as $Overlap(I_1, I_2) = \frac{|I_1 \cap I_2|}{\min(|I_1|, |I_2|)}$. It can reduce the impact of situations in which classes of one dataset contain substantially less individuals than in the other one. However, it is often incapable of ranking several alternative mappings with the same size of the overlap.
- *Pointwise Mutual Information* determines the reduction of uncertainty provided by the assignment of an instance to one class to the assignment to the other: $PMI(I_1, I_2) = \log_2 \frac{|I_1 \cap I_2| \times N}{|I_1| \times |I_2|}$.

- Log likelihood ratio represents a statistical test used to compare the fit of two hypotheses. The null hypothesis states that the probability $p(i \in I_1)$ that an instance belongs to I_1 does not depend on whether it already belongs to I_2 , i.e., $p_0 = p(i \in I_1 | i \in I_2) = p(i \in I_1 | \neg i \in I_2) = \frac{|I_1|}{N}$, where N is a total number of instances in both datasets. The alternative hypothesis states that $p_1 = p(i \in I_1 | i \in I_2) = \frac{|I_1 \cap I_2|}{|I_2|}$ and $p_2 = p(i \in I_1 | \neg i \in I_2) = \frac{|I_1| - |I_1 \cap I_2|}{N - |I_2|}$. The log likelihood ratio is defined as $-2(\log L(p_0, k_1, n_1) + \log L(p_0, k_2, n_2) - \log L(p_1, k_1, n_1) - \log L(p_2, k_2, n_2))$, where $\log L(p, k, n) = k \ln p + (n - k) \ln(1 - p)$, $k_1 = |I_1 \cap I_2|$, $k_2 = |I_1| - |I_1 \cap I_2|$, $n_1 = |I_2|$, and $n_2 = N - |I_2|$.
- *Information Gain* measures the reduction of entropy of assigning an instance to one set, if it has already been assigned to another set. $IG = e_1 - e_2$, where $e_1 = -\frac{|I_2|}{N} \log_2 \frac{|I_2|}{N}$ and $e_2 = -\frac{|I_1 \cap I_2|}{|I_1|} \log_2 \frac{|I_1 \cap I_2|}{|I_1|}$.

An empirical study [5] found the Jaccard index to be the most suitable similarity measure for instance-based ontology matching. However, this study was primarily aimed at identifying equivalence mappings, and the experiments were performed with the ontologies which actually had overlapping sets of instances. Because of this, we decided to perform experiments to evaluate the suitability of different similarity metrics for determining the best-fitting classes.

5 Experiments

We performed two sets of experiments. First, we tested the dataset selection algorithm in three different scenarios (section 5.1). Second, we performed experiments with the algorithm identifying best matching classes in order to choose the instance-based similarity measure best suited to the task.

5.1 Dataset Search

In our tests, we have applied the approach described in section 3 to the following datasets:

- ORO journals. A set of 3110 journals mentioned in the ORO repository constituting a part of *data.open.ac.uk*. Each individual belongs to the class *bibo:Journal*¹⁴.
- LinkedMDB films. A subset of 400 randomly selected instances of the class *movie:film*¹⁵ representing movies in the LinkedMDB repository.
- LinkedMDB music contributors. A subset of 400 randomly selected instances of the class *movie:music_contributor* representing music contributors for films in the LinkedMDB repository.

For each individual in these sets, we queried Sig.ma using their labels as keywords. The search results containing potentially relevant instances were aggregated, and individuals were grouped by data source and ontological class. These

¹⁴ <http://purl.org/ontology/bibo/Journal>

¹⁵ <http://data.linkedmdb.org/movie/film>

grouped results were used to produce the ranking of sources as described in section 3.1. Among the top-10 ranked data sources, we counted the number of actually relevant ones. Then, we applied the filtering mechanism using ontology schema matching results and checked the relevance of remaining sources. The results we obtained are presented in Table 1 for each dataset it shows the list of top ranked sources as well as our judgement whether these sources were actually relevant (column “+/-”). In the table, “(RKB)” denotes the datasets from RKBExplorer and “open EAN” corresponds to *openean.kaufkauf.net*. For both LinkedMDB datasets, we did not consider the LinkedMDB repository itself when it was returned in the search results. As we can see from the results, the initial search-based ranking managed to discover relevant datasets for the sets of individuals in question. Top-ranked sources in the *Journals* and *Films* categories contained relevant individuals which could be linked to the individuals in D_s , and their sets of individuals are to a large degree overlapping. For music contributors, the proportion of irrelevant sources was substantially larger due to higher ambiguity of human names. The filtering stage in all cases resulted in improving the ranking precision: only relevant sources were confirmed. However, if we look at the ranking of ontological classes (Table 2), we can see that correctly identifying classes presents a number of issues. The table shows the highest ranking classes returned after each stage of the algorithm (only one highest-ranking class from each ontology is shown). Top-ranked classes produced

Table 1. Test results: ranking of data sources

Dataset	Before filtering		After filtering	
	Top-ranked	+/-	Top-ranked	+/-
Journals	rae2001(RKB)	+	rae2001(RKB)	+
	dotac(RKB)	+	DBPedia	+
	DBPedia	+	dblp.l3s.de	+
	oai(RKB)	+	Freebase	+
	dblp.l3s.de	+	DBLP(RKB)	+
	wordnet(RKB)	-	eprints(RKB)	+
	www.bibsonomy.org	-		
	eprints(RKB)	+		
	Freebase	+		
	www.examiner.com	-		
Films	DBPedia	+	DBPedia	+
	open EAN	+	Freebase	+
	bestbuy.com	+		
	Freebase	+		
	www.answers.com	-		
	bitmunk.com	-		
	wordnet	-		
	www.examiner.com	-		
	it.bestshopping.com	+		
www.songkick.com	-			
Musicians	DBPedia	+	Freebase	+
	www.realpageslive.com	-	DBPedia	+
	twitter.com	-		
	BBC	+		
	www.songkick.com	+		
	Freebase	-		
	Open EAN	+		
	LinkedIn	-		
	dblp.l3s.de	-		
Yahoo!Movies	+			

Table 2. Test results: ranking of ontological classes

Dataset	Before filtering	After filtering	Best-fitting classes
Journals	<i>akt:Publication-Reference</i>	<i>akt:Journal</i>	<i>dc:BibliographicResource</i>
	<i>dc:BibliographicResource</i>	<i>yago:Periodical</i>	<i>akt:Publication-Reference</i>
	<i>foaf:Document</i>	<i>surc:Journal</i>	<i>akt:Journal</i>
	<i>swrc:Publication</i>	<i>dbpedia:Work</i>	<i>yago:Periodical</i>
	<i>vcard:VCard</i>	<i>freebase:book.periodical</i>	<i>dbpedia:Work</i>
	<i>yago:Periodical</i>		<i>freebase:book.periodical</i>
	<i>geo:SpatialThing</i>		
	<i>wn:Word</i>		
	<i>rss:item</i>		
<i>swap:SocialEntity</i>			
Films	<i>dbpedia:Work</i>	<i>dbpedia:Film</i>	<i>dbpedia:Film</i>
	<i>goodrelations:ProductOrServiceModel</i>	<i>yago:Movie</i>	<i>goodrelations:ProductOrServiceModel</i>
	<i>yago:Movie</i>	<i>freebase:film.film</i>	<i>yago:Movie</i>
	<i>icalendar:Vevent</i>		<i>freebase:film.film</i>
	<i>foaf:Person</i>		<i>searchmonkey:Product</i>
	<i>vcard:VCard</i>		
	<i>searchmonkey:Product</i>		
	<i>skos:Concept</i>		
	<i>geo:SpatialThing</i>		
<i>freebase:common.topic</i>			
Musicians	<i>vcard:VCard</i>	<i>freebase:film.music_contributor</i>	<i>freebase:film.music_contributor</i>
	<i>geo:SpatialThing</i>	<i>yago:AmericanTelevisionComposers</i>	<i>mo:MusicArtist</i>
	<i>swap:Person</i>		<i>dbpedia:Artist</i>
	<i>foaf:Person</i>		<i>yago:Composer</i>
	<i>dc:Agent</i>		
	<i>mo:MusicArtist</i>		
	<i>icalendar:vcalendar</i>		
	<i>dbpedia:Person</i>		
	<i>goodrelations:ProductOrService</i>		
<i>frbr:ResponsibleEntity</i>			

from the search results usually represent high-level concepts and correspond to superclasses of the original class: e.g., *foaf:Document* or *dc:BibliographicResource* for journals, *dbpedia:Work* for movies, and *foaf:Person* for musicians. The filtering stage largely removed these problems so that only classes with a stronger degree of semantic similarity were confirmed. However, it also reduced the recall in cases where a directly corresponding class was not present in the external ontology: e.g., individuals from *dotac.rkbexplorer.com* and *oai.rkbexplorer.com*, which only used the generic class *dc:BibliographicResource* were not considered as relevant sources for linking journals. Similarly, many relevant classes were filtered out because they were not considered as exact matches or subclasses of the class *movie:music_contributor* (e.g., *mo:MusicArtist* and *dbpedia:MusicalArtist*). In other cases, the algorithm selected too specific class, such as *yago:AmericanTelevisionComposers*. Applying the best-fitting class selection procedure in these cases (column 4) provided more adequate results.

5.2 Finding the Best-Fitting Class

In order to evaluate different set similarity metrics for the best-fitting class, we needed a set of multiple test cases. Each test case required the availability of gold standard mappings between instances as well as ontologies with detailed class hierarchies. To generate sufficient number of such test cases, we have chosen two large-scale datasets which has already been linked: DBPedia and Freebase. Pairs of classes for tests were selected from the YAGO ontology and the Freebase schema. We selected such pairs of classes $(c_y; c_f)$ from YAGO and Freebase respectively that:

- There is a set of *owl:sameAs* mappings $M_i = \{(i_y, i_f)\}$ such that $\forall i_y, i_f : i_y \in c_y, i_f \in c_f$.
- There is a pair of classes $(c_y^{top}; c_f^{top})$ such that $c_y \sqsubseteq c_y^{top}, c_f \sqsubseteq c_f^{top}$, and $c_y^{top} \equiv c_f^{top}$.
- There is no such class c_x such that $|c_x| < |c_y|$ and, given $M_i = \{(i_y, i_f)\}$, all i_y would belong to c_x . The same holds for c_f and i_f , respectively.

We selected medium-size classes from Freebase and DBPedia (having between 400 and 20000 individuals) with at least 400 mappings between them, coming from two different domains: people and organisations. After eliminating classes which did not satisfy the criteria or semantically irrelevant ones, the test set contained 111 pairs of classes. For each test, we randomly selected n individuals for which *owl:sameAs* mappings existed, and used them as I_s^{ast} . For these individuals, we ran the procedure described in section 4 using different sim_i and the sample size n . If the procedure returned the actual target class as the best fitting one, the result was considered correct. The test results are summarised in Table 3 (numbers show the proportion of correctly identified target classes). As can be seen, the log likelihood ratio clearly outperforms other metrics both in terms of absolute performance and robustness. The *PMI*, *IG*, and *Over* measures were found to be unsuitable for the task. While they usually return semantically correct class mappings, they tend to select too specific classes in the hierarchy.

Table 3. Test results: finding the best fitting classes

N	sim_i	$n = 50$	$n = 100$	$n = 200$	$n=400$
1	Jaccard index, JC	0.25	0.46	0.61	0.74
2	Corrected Jaccard index, JC_{corr}	0.41	0.51	0.65	0.74
3	Log likelihood ratio, $LogL$	0.93	0.96	0.97	0.98
4	Pointwise mutual information, PMI	0.12	0.07	0.06	0.05
5	Information gain, IG	0.0	0.0	0.0	0.0
6	Overlap coefficient, $Over$	0.0	0.0	0.0	0.0

6 Related Work

Although both the problem of search in semantic datasets and the task of data interlinking are actively studied in the Semantic Web community, there has been relatively little research dedicated to the task of search for relevant datasets. One recent approach [7] also discusses the problem of integrating a dataset with external semantic resources. As a use case, the authors consider the Google Refine application¹⁶ scenario: enriching data from tabular sources. The authors describe an extension to this application capable of linking these tabular data to external semantic repositories and discuss applicable linking techniques (e.g., SPARQL extension and reuse of Sindice and Silk services). However, their experiments only compare these techniques on the task of linking to pre-defined data sources, and do not focus on the actual search for relevant sources. The OKKAM project¹⁷ took a radical centralised approach, in which a global repository of entities exists and provides lookup services for other datasets to retrieve canonical URIs for their data instances.

To deal with the task of identifying matching classes, instance-based matching techniques are actively researched in the ontology matching community [1] and incorporated in several schema matching tools (e.g., ILIADS [13] and RIMOM [6]). In particular, in [15] the authors use the ‘bag of words’ approach adapted from the natural language processing: classes are annotated with the sets of string tokens extracted from properties of their instances, and similarity between classes is measured using the cosine similarity. However, this technique loses the information about distribution of words in different instances and is not suitable for estimation of the overlap between instance sets. As mentioned in section 4, the comparative study reported in [5] evaluated the suitability of different similarity metrics, although the focus of their task and their conclusions differ from ours.

7 Conclusion

The Linked Data cloud is constantly growing, and in order to make its use widespread, data owners must be able to publish their datasets without extensive knowledge about the state of the Web of Data or assistance from the research community. Interlinking is an important part of the publishing process and the one which can require substantial exploratory work with external data. Thus, this process has to become straightforward for data publishers and, preferably, require minimal human involvement. A specific feature of this problem is the fact that the amount of necessary information about the Web of Data which is immediately available on the client (data publisher) side is limited, and gathering this information is a time-consuming process for the user. The proposed solution provides the data publisher with a ranked set of potentially relevant data sources and, in addition, a partial configuration of the data linking tool

¹⁶ <http://code.google.com/p/google-refine/>

¹⁷ <http://www.okkam.org>

(classes containing relevant sets of instances). In this way, it can substantially reduce the need to perform exploratory search. One direction of the continuation work, which we are currently pursuing, involves developing algorithms which are able to suggest to the user suitable instance matching algorithms for the data linking tool depending on the task at hand.

Another potentially interesting research direction is related to the development of semantic indexes. Search for relevant data repositories can become a novel interesting use case in addition to the more common search for entities and documents. In order to support it, new types of search services can be valuable: for example, batch search for a large array of resource labels instead of multiple queries for small sets of keywords, which increase number of server requests and overall processing time.

Acknowledgements. This research has been partially funded under the EC 7th Framework Programme, in the context of the SmartProducts project (231204).

References

1. Euzenat, J., Shvaiko, P.: *Ontology matching*. Springer, Heidelberg (2007)
2. Fernandez, M., Zhang, Z., Lopez, V., Uren, V., Motta, E.: *Ontology augmentation: combining semantic web and text resources*. In: 6th International Conference on Knowledge Capture, K-CAP 2011 (2011)
3. Gracia, J., Mena, E.: *Matching with CIDER: Evaluation report for the OAEI 2008*. In: 3rd Ontology Matching Workshop (OM 2008) at the 7th International Semantic Web Conference (ISWC 2008), Karlsruhe, Germany (2008)
4. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: *When owl:sameAs isn't the same: An analysis of identity in linked data*. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I*. LNCS, vol. 6496, pp. 305–320. Springer, Heidelberg (2010)
5. Isaac, A., van der Meij, L., Schlobach, S., Wang, S.: *An Empirical Study of Instance-Based Ontology Matching*. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 253–266. Springer, Heidelberg (2007)
6. Li, J., Tang, J., Li, Y., Luo, Q.: *RiMOM: A dynamic multistrategy ontology alignment framework*. *IEEE Transactions on Knowledge and Data Engineering* 21(8), 1218–1232 (2009)
7. Maali, F., Cyganiak, R., Peristeras, V.: *Re-using cool URIs: Entity reconciliation against LOD hubs*. In: *Workshop on Linked Data on the Web (LDOW 2011), WWW 2011, Hyderabad, India* (2011)
8. Nikolov, A., d'Aquin, M.: *Identifying relevant sources for data linking using a semantic web index*. In: *Workshop on Linked Data on the Web (LDOW 2011), WWW 2011, Hyderabad, India* (2011)
9. Nikolov, A., Motta, E.: *Capturing emerging relations between schema ontologies on the web of data*. In: *Workshop on Consuming Linked Data (COLD 2010), ISWC 2010, Shanghai, China* (2010)

10. Nikolov, A., Uren, V.S., Motta, E., De Roeck, A.: Integration of Semantically Annotated Data by the KnoFuss Architecture. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 265–274. Springer, Heidelberg (2008)
11. Nikolov, A., Uren, V., Motta, E., de Roeck, A.: Overcoming Schema Heterogeneity between Linked Semantic Repositories to Improve Coreference Resolution. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 332–346. Springer, Heidelberg (2009)
12. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: Live views on the Web of Data. *Journal of Web Semantics* 8(4), 355–364 (2010)
13. Udea, O., Getoor, L., Miller, R.J.: Leveraging data and structure in ontology integration. In: SIGMOD 2007, Beijing, China, pp. 449–460 (2007)
14. 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)
15. Wang, S., Englebienne, G., Schlobach, S.: Learning Concept Mappings from Instance Similarity. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 339–355. Springer, Heidelberg (2008)

Interacting with Linked Data via Semantically Annotated Widgets

Armin Haller¹, Tudor Groza², and Florian Rosenberg³

¹ CSIRO ICT Centre

`armin.haller@csiro.au`

² University of Queensland

`tudor.groza@uq.edu.au`

³ IBM T.J. Watson Research Center

`rosenberg@us.ibm.com`

Abstract. The continuous growth of the Linked Data Web brings us closer to the original vision of the Web as an interconnected network of machine-readable resources. There is, however, an essential aspect in principle still missing from this vision, i.e., the ability for the Web user to interact directly with the Linked Data in a read/write manner. In this paper we introduce a lifecycle and associated mechanism to enable a domain-agnostic read/write interaction with Linked Data in the context of a single data provider. Our solution uses an ontology to build a binding front-end for a given RDF model, in addition to RDFa to maintain the semantics of the resulting form/widget components. On the processing side, a RESTful Web service is provided to seamlessly manage semantic widgets and their associated data, and hence enable the read/write data interaction mechanism. The evaluation shows that the generation process presents no performance issues, while the content overhead required for the actual form-data binding is kept to a minimum.

1 Introduction

Over the course of the last five years, the progressive use of Semantic Web technologies in conjunction with the Linked Data principles [3] has led to an explosion of datasets being openly published on the Web. The emergence of this Linked Data Web [6] was foreseen in the very early conception of the World Wide Web itself. The original vision had, however, a second part that regarded the Web as a bi-directional communication channel between content producers and consumers [5]. In other terms, as opposed to today's read-only Web, which allows us only to act as simple viewers with respect to the published content or data, the read/write Web enables a direct interaction, as part of a common creative process.

From the pure textual created/consumed content perspective, social media environments such as Facebook, Twitter, blogs or wikis are close to fulfilling the vision. However, interacting with the large diversity of existing Linked Data in a read/write fashion over a typical Web application is still an open research

topic. Some work has already been done in the area (e.g., [16,4,13]), in general by creating (HTML) forms as a medium between RDF data and humans. While this bridging concept represents probably the best solution, the underlying technical aspects of current approaches share a common drawback: they either require manual mappings between domain concepts and form components, or are specifically tailored for a particular domain (i.e., forms generated for a specific schema/ontology describing a dataset).

This paper presents a novel solution aimed to bring us a step closer to the original vision, by providing a lifecycle and associated mechanism to enable read/write interactions with Linked Data exposed by a single producer, via Web forms embedded in a widget. We use widgets as a manifestation paradigm because of their versatility and seamless integration possibility within diverse Web environments. The interaction takes place in a *local* dataset context (i.e., the context of a dataset exposed by the producer). However, subject to the user's knowledge, it also takes advantage of the *global* Linked Data Web context. The proposed lifecycle is applicable to arbitrary RDF graphs (hence being domain-agnostic) and comprises three phases, described in the following via the direct contributions of this paper:

Widget generation. We propose a markup ontology that describes the structure of a Web form, the RDFa User Interface Language [12] (RaUL), for semantically defining Web widgets. We use RDFa to maintain the semantics of a widget in XHTML and for binding input data in Web forms/widgets to an RDF graph. A RaUL widget provides a binding to RDF data similar to what popular traditional web application frameworks (e.g. Ruby on Rails [19], GWT [10], Apache Struts [21] etc.) provide for a relational model. The data submitted through a semantically annotated widget is processed by our proposed ActiveRaUL client-side JavaScript (JS) API and send to the ActiveRaUL Web service where it is stored as RDF triples in the underlying database. If the same data is used by multiple Web widgets, it can be bound to the underlying widget elements by querying for the same uniquely defined resource. These relations (triples) can exist locally, or in the Linked Data Web.

Widget deployment. We developed ActiveRaUL, a RESTful Web service to seamlessly deploy, manage and retrieve semantically annotated widgets. ActiveRaUL maps the four common HTTP methods, POST, GET, PUT, DELETE to corresponding CRUD operations (i.e., CREATE, READ, UPDATE, DELETE) on an RDF model in the backend. This approach relieves the Web developer from defining SPARQL queries to be executed when data is submitted in a Web widget. Depending on the desired operation on the model, only the appropriate method in our Web service has to be called.

Widget usage. The Web user is able to transparently manipulate the underlying RDF data via traditional Web forms in a browser, while being unaware of the underlying data binding mechanism. Using a non-ambiguous RDF model

Figure 1 displays three examples of simplified forms (A, B, and C) used in a trading platform. Each form is enclosed in a dashed border and contains various input fields and buttons.

- Form A:** Includes fields for FOAF URI, First Name (Maria), Last Name (Example), Your Email (maria@example.com), Password (*****), Gender (Male/Female), and Birthday (10 October 1978). A "Register" button is at the bottom.
- Form B:** Includes fields for Product URI, Product Name (Sony Bravia LX900), Product Description (LCD Television), Product Depiction (http://example.org/prod...), Offer Type (Sell), and Price (5500 USD). An "Add" button is at the bottom.
- Form C:** Includes fields for Product Name (Sony Bravia LX900), Product Description (LCD Television), Amount (2), Transaction Type (Buy), and Price (5500 USD). An "Order" button is at the bottom.

Fig. 1. Examples of simplified forms used in a typical trading platform

as binding mechanism in RaUL forms allows the ActiveRaUL client-side JS API to retrieve data that already exists in the Linked Data Web (e.g., personal information about a user which is stored and published elsewhere) and pre-populate the form elements.

Following this lifecycle, a developer only needs to deploy the backend service¹ and to deal with the usual form styling elements, unless s/he explicitly opts for manually creating the widget model. In this case, knowledge about the RaUL ontology is required, however, even so, the widget model once created can be shared with and immediately adopted by others.

The remainder of the paper is structured as follows: Sect. 2 introduces a motivating/running example used in Sect. 3 to showcase the phases of the proposed lifecycle. Sect. 4 presents the ActiveRaUL service, its architecture and the client-side JS API. In Sect. 5 we benchmark the overhead introduced by RaUL annotations and test the performance of the ActiveRaUL service, and before concluding in Sect. 7 we discuss existing related work in Sect. 6.

2 Running Example

To illustrate our approach, we introduce a running example inspired by a typical e-Commerce interaction. Trading platforms are ideal candidates for exposing their product datasets on the Linked Data Web. Pioneering examples already exist, such as <http://www.bestbuy.com> or <http://www.sears.com>, which use RDFa and the Good Relations ontology [14] to publish its product information as Linked Data. In addition, due to the very nature of the domain, one can truly exploit the benefits of the Linked Data Web, as product instances in one site may have slightly different or extra information on other sites. Hence, being able to link the different instances brings an added value for the end-user.

Our example simulates a second-hand goods trading platform at which registered users can sell and buy second-hand goods². For demonstration purposes,

¹ The service is packaged as a Web archive and available for download at: <http://w3c.org.au/raul/>

² The example can be tested online at <http://w3c.org.au/raul/demo.html>

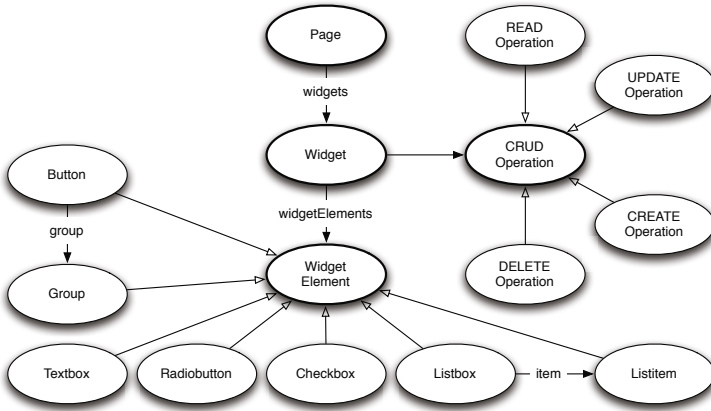


Fig. 2. The RaUL form model

and later for understanding the value provided by our approach, we consider a scenario in which a typical user performs three basic (yet common) operations: registration, selling a product and buying a product. Fig. 1 depicts these three operations by means of simplified widgets generated by our solution from the underlying RDF data supposedly used by the trading platform. Some additional details on the three operations are presented as follows:

User Registration. A new seller/buyer has to register a new user account on the e-Commerce website. The user requests the registration form, which offers him the standard fields to fill in his data (Fig. 1, part A). One option would be to directly use the FOAF [7] profile.

Product Advertisement. A user, once registered, may add products to be sold (Fig. 1, part B). The associated form/widget consists of examples of common fields used to describe a product. Similar to the first operation, the user could re-use an existing description of the product on the Linked Data Web by entering the product's URI.

Bid on a Product. Finally, the user may want to purchase an advertised product. The product can be easily found by filling in the form/widget depicted in Fig. 1, part C.

In the remainder of the paper we show how by starting from the underlying RDF representation of the product and using RaUL, we are able to generate semantically annotated front-end widgets to uniformly identify the data input, and to link it to the Linked Data Web. At the same time, we show how to manipulate the data stored in the trading platform's triple store over the ActiveRaUL RESTful service interface.

3 A Framework for Data Publishing in RDFa

In this section we detail the three phases of our proposed lifecycle, i.e., widget generation, deployment and usage, all using as back-end the ActiveRaUL service presented in Sect. 4.

3.1 Widget Generation

The first step of the lifecycle creates a widget model in RDF, using the RaUL ontology. The widget generation task is performed by the developer, who, subject to the chosen option (i.e., manual or semi-automatic), may need to be familiar with the RaUL ontology. However, this is done once for the lifetime of the site and can be compared with the creation of a form (widget) in HTML or a template in Web application frameworks like Ruby on Rails [19], GWT [10], Apache Struts [21] etc. However, in contrast to HTML forms that are usually custom-build for every website, RaUL-based widget models are reusable RDF graphs which are assigned a URI by ActiveRaUL. As such, they can be reused and can become standardised widget models for certain tasks themselves. Using the ActiveRaUL framework, there are two options to create a widget: (i) manually, by posting a handcrafted RaUL-based widget model (in RDF/XML, RDF/JSON or RDF/N3) to the ActiveRaUL service; or (ii) (semi)-automatically, by posting an arbitrary RDF graph to the ActiveRaUL service from which a RaUL-based widget model will be created.

The RaUL ontology³ (defining the widget model) consists of two parts: (i) **Form controls** describing the structure of the widget, and their associated operations (i.e., READ, UPDATE, CREATE or DELETE), and (ii) a **Data model** defining the structure of the exchanged data as RDF statements which are referenced from the *form model* via a data binding mechanism. The model gives meaning to the data used in the *form controls* by uniquely referencing standard ontologies on the Semantic Web.

Form Controls. A *form control* in RaUL is an element that acts as a direct point of user interaction and provides access to the triples describing the *data model*. Fig. 2 depicts a high-level overview of the RaUL form model defining a set of *form controls*. As most of the concepts have a self-explanatory name we refer the interested reader to our earlier publications [12,11] for more detail on the ontology itself. It is worth mentioning, however, that the controls have corresponding XHTML elements which are used when generating the widget for rendering and interaction purposes. Fig. 3 shows the RaUL representation of a *Listbox* and its corresponding XHTML created by the ActiveRaUL service, part of the *Product Advertisement* widget introduced in our running example in Sect. 2.

Data Model. In contrast to untyped key/value pairs used in traditional XHTML forms, data in RaUL widgets is submitted in a structured way, as RDF data according to a certain schema. Hence, the data model is de-coupled from the form

³ RaUL can be found at <http://purl.org/NET/raul#>

<pre> <!DOCTYPE rdf:RDF [<!ENTITY product "http://w3c.org.au/raul/service/public/forms/addproduct" >]> <rdf:Description rdf:about="@product#currency" > <rdf:type rdf:resource="http://purl.org/NET/raul#Listbox" /> <id xmlns="http://purl.org/NET/raul#" >transaction.type</id> <value xmlns="http://purl.org/NET/raul#" >@product#value.currency </value> <list xmlns="http://purl.org/NET/raul#" >@product#currency_list </list> </rdf:Description> <rdf:Description rdf:about="@product#currency_list" > <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns #Seq" /> <rdf:1 xmlns="http://purl.org/NET/raul#" rdf:resource="@product#currency.1" /> <rdf:2 xmlns="http://purl.org/NET/raul#" rdf:resource="@product#currency.2" /> </rdf:Description> <rdf:Description rdf:about="@product#currency.1" > <rdf:type rdf:resource="http://purl.org/NET/raul#Listitem" /> <label xmlns="http://purl.org/NET/raul#" >USD</label> <value xmlns="http://purl.org/NET/raul#" >USD</value> </rdf:Description> <rdf:Description rdf:about="@product#currency.2" > <rdf:type rdf:resource="http://purl.org/NET/raul#Listitem" /> <label xmlns="http://purl.org/NET/raul#" >EUR</label> <value xmlns="http://purl.org/NET/raul#" >EUR</value> </rdf:Description> </pre>	<pre> <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" [<!ENTITY product "http://w3c.org.au/raul/service/public/forms/addproduct" >]> <div about="@product#currency" typeof="raul:Listbox" > </div> <select id="currency" name="currency" > <option value="USD">USD</option> <option value="EUR">EUR</option> </select> <ol style="display:none;" about="@product#currency.options" > <li rel="rdf:1" resource="@product#currency.1" > <li rel="rdf:2" resource="@product#currency.2" > <div about="@product#currency.1" typeof="raul:Listitem" > </div> <div about="@product#currency.2" typeof="raul:Listitem" > </div> </pre>
--	---

Fig. 3. A RaUL Listbox in RDF/XML (left) and the generated XHTML+RDFa (right)

controls. Handcrafting the form–data mapping provides the dataset developers with full flexibility in defining the structure of the model. Every form control in the widget maps to (a set of) triples that describe the input. For example, the form controls composing the *User Registration* widget in our running example (Fig. 1 part A in Sect. 2) could be mapped using the FOAF ontology and the W3C Time ontology [15] as follows: (i) *FOAF URI* – foaf:Person (ii) *First name* – foaf:givenName (iii) *Last name* – foaf:familyName (iv) *Email* – foaf:mbox (v) *Birth day* – time:day (vi) *Birth month* – time:month (vii) *Birth year* – time:year. Similarly, the mappings behind the other two widgets could be handcrafted, using for example the GoodRelations ontology [14].

The actual binding of the *form controls* to the underlying data structure is realised via reification within the RDFa embedded in the resulting XHTML representation of the form controls. Fig. 4 shows, for example, how the *firstname Textbox* in the RaUL *User Registration* widget references the RDF triple representing the corresponding underlying data element (i.e., $\langle \text{http://...} \rangle \langle \text{foaf:firstName} \rangle \langle ' \rangle$). The `rdf:subject` triple references the URI assigned by the ActiveRaUL service for the instance graph, the `rdf:predicate` triple is a reference to the URI of a standard Web ontology property, and the `rdf:object` triple is a reference to the value that can be edited by the respective form control. Empty `rdf:object` fields serve as place-holders and are filled at runtime by the ActiveRaUL client-side JS API with the user input.

Semi-automatic Widget Generation. To assist the developer/ontology engineer in the creation of a RaUL widget model, we propose a mapping framework to semi-automatically derive semiotics according to our RaUL user interface model from arbitrary RDF graphs. Although it can be foreseen that such user interface (widget) models become part of the Linked Data Web, there are no standard

```

/• Defined Reification Mapping •/
<span about="#valuefirstname">
  <span property="rdf:subject" content="http://..." />
  <span property="rdf:predicate" content="foaf:givenName" />
  <span property="rdf:object" content="" />
</span>

```

Fig. 4. RDFa reified triple for a foaf:givenName object

models available yet. Thus, the ActiveRaUL service includes a generation algorithm that creates a best-effort widget model generation on any deployed RDF graph. Figure 5 briefly outlines our algorithm. The *RaUL-generation* function takes as input the URI (U) created by the ActiveRaUL controller for the new widget model (see Sect. 4.1) and a ground RDF graph G with no blank nodes (if blank nodes exist, they are discarded). The algorithm iterates through all unique subject URIs and creates a corresponding RaUL *Widget* (line 6-8). The *trim-fragment()* function creates a fragment from any input URI and ensures that there are no duplicates. Then, for every predicate in G , where the subject is a URI reference, a RaUL *WidgetElement* is created. The actual type of the *WidgetElement* is determined as follows: if the predicate p_x exists only once in G and if the object is a *Literal* and the XSD datatype is not *boolean* then a RaUL *Textbox* is created (line 11-14). Also the associated reified triple referenced through the value property (line 12) is created with the *create-reified-triple()* function (line 13). The value of the object o_x in G is inserted as the value for the object of the reified triple. If the datatype of the object o_x is *boolean* then a RaUL *Group* (17-19) is created and two RaUL *Radiobuttons* are created (line 20-24). If a predicate p_x occurs more than once in G and if the object o_x is a *Literal*, a RaUL *Listbox* is created (line 26-31), a RDF sequence with the number of predicates p_x (line 32) is inserted and for each predicate a RaUL *Listitem* with the value of the object literal in G is created. If the object o_x is a *Literal* then a *raul:label* with the value of o_x is created (line 38-39), if the object is a URI reference and if the URI reference is in the local graph, we follow it and check for the existence of a triple with a predicate *rdfs:label*. If it exists we use the object o_y of this triple for the *raul:label* property of the *Listitem* (line 42-43), otherwise, we use the URI reference (line 44-45). For every *WidgetContainer* we create a RaUL submit *Button* (line 51).

The resulting RaUL widget model is currently meant for assisting the developer, but in most cases he will need to refine the model which can be retrieved via its URI assigned at generation time by ActiveRaUL. However, additionally to the best-effort generation of *form controls*, a correct data binding between the *form controls* and *data model* is ensured, significantly easing the effort of the developer in defining/refining the widget model.

3.2 Widget Deployment

The ActiveRaUL service offers an endpoint to deploy a widget with a POST request to its `/public/forms` resource. The developer has two options for the payload when deploying a widget depending on the chosen generation path:

```

0: RaUL-generation( $U, G$ )
  Where  $U$  is the URI assigned to the new widget model by the ActiveRaUL controller
  Where  $G$  is a ground RDF graph (no blank nodes) containing a set of triples  $(s, p, o)$  with
   $s \in \text{URIs}$ ,  $p \in \text{URIs}$ ,  $o \in \text{URIs} \cup \text{Literals}$ , where
   $s$  is the subject,  $p$  the property and  $o$  the object of the triple and, where
   $G(s_u)$  is the set of unique subjects in  $G$ .
1: Create empty widget model  $W$ 
    $W(G) = I$ 
2:  $OPEN.enqueue((s_1, p_1, o_1), \dots, (s_m, p_m, o_m))$ 
3: WHILE  $OPEN \neq \emptyset$ 
4:    $(s_x, p_x, o_x) = OPEN.dequeue()$ 
5:   IF  $(s_x, p_x, o_x) == END$ , report success and return  $W$ 
6:   FOR each  $s_x$  in  $G(s_u)$ 
7:      $U_{wc} = U . trim-fragment(s_x)$ 
8:      $W(G) = W(G) \cup (U_{wc}, rdf:type, raul:WidgetContainer)$ 
9:     IF  $|p_x|$  in  $G == 1$  AND if  $o_x \cup \text{Literals}$  AND literal-type( $o_x$ ) is not boolean
10:       $U_{tb} = U . trim-fragment(p_x)$ 
11:       $W(G) = W(G) \cup (U_{tb}, rdf:type, raul:Textbox)$ 
12:       $W(G) = W(G) \cup (U_{tb}, raul:value, value-reference(o_x))$ 
13:       $W(G) = W(G) \cup (create-reified-triple(value-reference(o_x)), p_x, o_x)$ 
14:       $W(G) = W(G) \cup (U_{tb}, raul:label, trim-fragment(p_x))$ 
15:     ELSEIF  $|p_x|$  in  $G == 1$  AND if  $o_x \cup \text{Literals}$  AND literal-type( $o_x$ ) is boolean
16:       $U_{gr} = U . trim-fragment(p_x)$ 
17:       $W(G) = W(G) \cup (U_{gr}, rdf:type, raul:Group)$ 
18:       $W(G) = W(G) \cup (U_{gr}, raul:value, value-reference(p_x))$ 
19:       $W(G) = W(G) \cup (create-reified-triple(value-reference(p_x)), p_x, o_x)$ 
20:       $U_{rbt1} = U . trim-fragment(U_{gr})."1"$ 
21:       $U_{rbt2} = U . trim-fragment(U_{gr})."2"$ 
22:       $W(G) = W(G) \cup (U_{rbt1}, rdf:type, raul:Radiobutton) \dots$  same for  $U_{rbt2}$ 
23:       $W(G) = W(G) \cup (U_{rbt1}, raul:group, U_{gr}) \dots$  same for  $U_{rbt2}$ 
24:       $W(G) = W(G) \cup (U_{rbt1}, raul:label, trim-fragment(p_x)) \dots$  same for  $U_{rbt2}$ 
25:     ELSEIF  $|p_x|$  in  $G > 1$  AND if  $o_x \cup \text{Literals}$ 
26:       $U_{lb} = U . trim-fragment(p_x)$ 
27:       $W(G) = W(G) \cup (U_{lb}, rdf:type, raul:Listbox)$ 
28:       $W(G) = W(G) \cup (U_{lb}, raul:value, value-reference(p_x))$ 
29:       $U_{ll} = U . create-listuri(U_{lb})$ 
30:       $W(G) = W(G) \cup (U_{lb}, raul:list, U_{ll})$ 
31:       $W(G) = W(G) \cup (create-reified-triple(value-reference(p_x)), p_x, o_x)$ 
32:      FOR each  $p_x$ 
33:         $i++$ 
34:         $U_i = U . trim-fragment(U_{lb}).i$ 
35:         $W(G) = W(G) \cup (U_{ll}, create-rdf-seq-element(p_x), U_i)$ 
36:         $W(G) = W(G) \cup (U_i, rdf:type, raul:Listitem)$ 
37:         $W(G) = W(G) \cup (U_i, raul:value, o_x)$ 
38:        IF  $o_x \in \text{Literals}$ 
39:           $W(G) = W(G) \cup (U_i, raul:label, o_x)$ 
40:        ELSE
41:          IF  $o_x$  in  $G(s_u)$ 
42:            IF exists  $p_y$  in  $G$  where  $s == s_u$  AND  $p_y == "rdfs:label"$ 
43:               $W(G) = W(G) \cup (U_i, raul:label, o_y)$ 
44:            ELSE
45:               $W(G) = W(G) \cup (U_i, raul:label, o_x)$ 
46:          ENDIF
47:        ENDIF
48:      ENDFOR
49:    ENDFOR
50:  ENDIF
51:   $W(G) = W(G) \cup (U, rdf:type, raul:Button)$ 
52:  ENDFOR
53:  ENDWHILE
54:  RETURN Failure

```

Fig. 5. RaUL widget model generation algorithm

(i) a handcrafted **RaUL-based widget model** that can be deployed within the public namespace `/public/forms` of the ActiveRaUL service backend (to support the re-use of generic form models in other Semantic Web applications and enable the emergence of standard widget models – e.g. a user registration form shared between many sites), and (ii) an **Arbitrary RDF graph**, which sent to the same endpoint will trigger the ActiveRaUL service to perform a best-effort generation of a widget according to the process described above.

The ActiveRaUL service supports different data representations, such as RDF/XML, RDF/JSON or RDF/N3. If the `POST` request is successful, the service will return the URL of the newly created widget in the `HTTP Location` header of the response, for example, `/public/forms/addproduct`. The resource name `addproduct` is automatically generated from the URI of the widget class in RDF.

3.3 Widget Usage

Once a widget has been deployed with ActiveRaUL by the developer, a Web user can access and use it through a browser. The browser issues an `HTTP GET` request to the URL that was created in the widget deployment phase to retrieve the widget. It depends on the `HTTP Accept` header in the `GET` request to determine what content type is sent back to the client. In the case of accessing the URL through a browser the response content type is `XHTML`.

Whenever the Web user fills in the form or changes already existing data in a form, the ActiveRaUL client-side JS API processes the form at submission time (details in Sect. 4.2) and sends the appropriate `HTTP` message (derived by interpreting the type of *CRUDOperation* in the RaUL widget model) to the ActiveRaUL service. The client-side JS API distinguishes several cases when updating the `rdf:object` in the reified value triple depending on the type of the form control. For example, the values provided in *Textboxes* are directly written to the `rdf:object` in the value triple. *Listboxes*, on the other hand, are more complex. After the submission of a *Listbox* form control the client-side JS API creates a *checked* relation for all selected *Listitems*. Additionally, it writes the reference to the object describing the *Listitem* into the `rdf:object` of the value triple. If the *Listbox* is a multi select one, defined by its *multiple* property, the referenced reified triple in the value property must be an RDF collection, whereby all selected *Listitem* references are written to the value triple.

In our running example, when a Web user fills in the *Product Advertisement* widget and submits it, the ActiveRaUL client-side JS API issues a `POST` request to the resource `/public/forms/addproduct/`, with a payload consisting of the RDF triples that are parsed from the RDFa annotations and the user input data. The ActiveRaUL service processes the request, inserts the data in the RDF triple store, and sends the URI of the newly created resource for the submitted data, e.g., `/public/forms/addproduct/101` in the `HTTP Location` header back to the client. This uniquely identified data can then be accessed with different widgets, as long as the data binding uses similar URI references for the `predicates` in the value triple. For example, when a user posts a certain product for sale, this data can also be retrieved in the widget that displays a product for sale.

3.4 Data Reuse

An important feature in this lifecycle is the reuse of existing data. Often, the data to be provided in a widget is already present on the Linked Data Web (e.g., a FOAF file describing a person) which usually includes many of the properties required by a registration form. The relation-based data binding implemented by RaUL form controls via standardised ontologies (e.g., *foaf:givenName*), enables a direct re-use of such Linked Data. The user can hence point, for example, to an existing FOAF file and ActiveRaUL will automatically fill in the corresponding widget controls.

All widget controls designed to reference an *owl:sameAs* relation are treated as reference to external data. From the XHTML rendering perspective, these widgets are mapped to *Textboxes* with an associated **update** button. At runtime, the Web user can directly provide a URL to an existing Linked Data resource in the *Textbox* which is used by the client-side JS API to retrieve the RDF graph and pre-fill the form controls in the widget if the data exists in the resource graph. Alternatively, the user can type in a search term which the client-side JS API uses for a call to the Sindice API [18] which in turn returns an RDF graph that is again used to pre-fill the remaining form controls. In either case, if a URI to an external resource is provided, the underlying *owl:sameAs* relations for this form control asserts that the graph representing the user's input data and its URI assigned by ActiveRaUL is the same individual with a different URI in the Linked Data Web.

An illustrative example of the use of form controls that reference external data is shown in the *User Registration* widget (Fig. 1 part A in Sect. 2), where the **FOAF URI** control was mapped to a *foaf:Person*, which enables the form to auto-fill the rest of the controls according to the given schema. Similarly, in the *Product Advertisement* widget (Fig. 1 part B in Sect. 2) the **Product URI** could be mapped to a GoodRelations *gr:Offering* object.

4 The ActiveRaUL System

The lifecycle introduced in Sect. 3 relies, particularly for the deployment and usage phases, on the ActiveRaUL backend⁴ described in this section. The backend consists of two main parts, as depicted in Fig. 6: (i) a RESTful Web service, and (ii) a client-side JS API. In the following, we detail the technical details of both the RESTful Web service, as well as the client-side JS API.

4.1 ActiveRaUL RESTful Web Service

The RESTful Web service provides a uniform way to manage widgets and the data that is processed by a widget. It abstracts from specific low-level details, such as storing and querying RDF data by interacting with an RDF triple store. This enables Web developers to use and integrate the ActiveRaUL framework

⁴ ActiveRaUL is available at <http://w3c.org.au/raul/service>

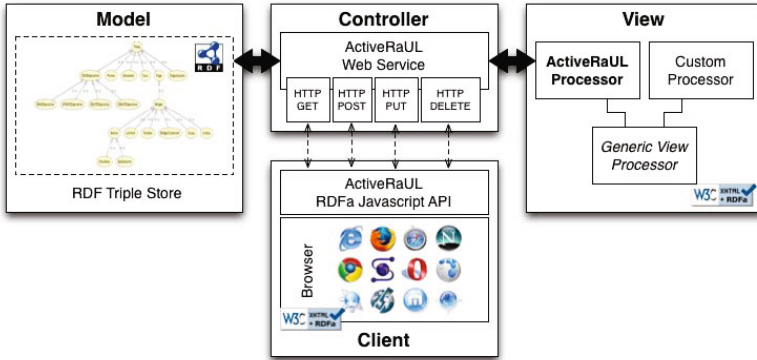


Fig. 6. Architecture of ActiveRaUL

without requiring a deep understanding of the core Semantic Web technologies (e.g., SPARQL, triple stores, etc.). Additionally, the service provides a mechanism to produce and render widgets via RDFa annotated XHTML forms. As shown in Fig. 6, the ActiveRaUL service architecture implements a Model-View-Controller (MVC) pattern [20].

Model. The *form controls* and their associated *data model* constitute the *model* part of the ActiveRaUL system. The service uses the RaUL ontology for defining widgets. However, its implementation is generic and can accommodate diverse models, as only the pluggable rendering part is depending on the RaUL ontology. ActiveRaUL currently uses OpenRDF Sesame⁵ as a triple store to persist all RDF data. However, any other triple store can be plugged into the backend with little modifications.

View. The view in MVC is the part that the user sees. In ActiveRaUL, the model can be queried using different representations, e.g., RDF/XML, RDF/JSON, RDF/N3 and XHTML+RDFa. The first three representations are not meant for human consumption and as such, do not require the generation of a view. For those, ActiveRaUL provides serialisations according to the content type requested in the HTTP header.

Only the XHTML+RDFa representation is meant for human consumption. Its rendering as a widget depends on the underlying widget model, in our case RaUL. However, to keep this view generic, the `GenericViewProcessor` component is provided as a Java interface that defines method signatures for the rendering functionality specific to a particular widget model. For RaUL, we provide an `ActiveRaULProcessor` that implements the view generation based on the RaUL model. The view generation is triggered once a client requests a view via the controller (using an HTTP GET request with the `Accept` header MIME type set

⁵ <http://www.openrdf.org/>

to `application/xhtml+xml`). The view generation then traverses all the form controls and generates the necessary XHTML+RDFa (see Section 3). The widget layout is built via CSS references.

Controller. The *controller* is responsible for creating, updating and deleting widget definitions and associated data required to be processed and stored as part of a form submission. The *controller* also assigns URIs to the submitted resources and returns the URL in the HTTP Location header of the response, for example, `/public/forms/addproduct`. To deal with duplicate names, unique numbers are appended at deployment time, e.g., `addproduct1`. For the instance data submitted for a widget, the ActiveRaUL service dynamically assigns a URI, such as the `/public/forms/addproduct/101` we mentioned above.

Errors are handled by returning the status code 500 (Internal Server Error) if an unexpected error happens. For errors related to wrong or incorrect input (payload), the service returns a status code 400 (Bad Request). Incorrect URL parameters will result in a 404 (Not Found).

User Authentication. In the URL scheme of the ActiveRaUL service the `/public/` part of the resource identifier is essentially a reference to a user id. As the name implies, `public` represents an open access space to upload forms and data. Any other `{userid}` parameter in the `{userid}/forms` resource represents the user id that has access to the forms under this specific `forms` resource. Access to any `{userid}`, but the `/public/` identifier, is only granted if the necessary HTTP authentication credentials are present in the corresponding HTTP requests.

4.2 ActiveRaUL Client-Side JS API Processing

The current implementation of the ActiveRaUL client-side JS API uses the `rdflibrary`⁶ library as an RDFa parser and jQuery for handling and querying the XHTML DOM tree. The actual processing consists of two main steps: (i) *data binding*, and (ii) *RDFa parsing and server communication*.

The data binding is performed immediately prior to parsing the RDFa. If data is retrieved from the Linked Data Web as described in Section 3.4, the data fetched from the external RDF Graph is treated as if it was provided by the user through the form. The update of the form controls is done by querying the received RDF for the object values of the predicates defined in the reified triple of the widget, which are then replaced with each successful query result. The processing of the data binding is done directly over the XHTML DOM tree using the jQuery library. For each form element the reified triple is identified by its URI and the respective object is replaced with the user input value.

After the data binding operations, the document is parsed to extract the RDF triples from the XHTML+RDFa representation and the full RDF graph is sent to the ActiveRaUL service. To determine which operation to invoke in the ActiveRaUL service, the client extracts the invocation URL and the HTTP method (from the type of the *CRUDOperation* defined in the RDFa annotations).

⁶ <http://code.google.com/p/rdflibrary/>

Table 1. Added overhead by RDFa markup for a HTTP GET response for a form request

Form Element	XHTML		XHTML+RDFa		# triples		Overhead in %	
	min	max	min	max	min	max	min	max
Widget	72	115	251	376	3	6	248%	226%
Textbox	41	100	88	377	2	7	114%	277%
Listbox	49	125	228	459	5	9	365%	367%
Button	48	81	98	415	2	8	104%	412%

Table 2. Added overhead by RDFa markup in HTTP POST/PUT requests for a form submission/update

Form Element	XHTML		RDF/XML		# triples		Overhead in %	
	min	max	min	max	min	max	min	max
Widget	278	321	525	617	3	6	88.84%	92.21%
Textbox	239	298	454	613	2	7	89.95%	105.7%
Listbox	274	323	656	758	5	9	139.41%	134.67%
Button	245	278	459	720	2	8	87.34%	158.99%

5 Evaluation

Since Web forms/widgets are the de facto data interaction mechanism on the Web and their superiority over direct RDF editing is indisputable, we chose to perform a quantitative evaluation to analyse possible performance issues of our novel data binding mechanism. The performance of the ActiveRaUL framework is influenced by two factors: (i) the overhead introduced by the RDFa annotations, and (ii) the performance of the widget/model generation.

RDFa Overhead. The first aspect, i.e., the RDFa overhead, can be investigated by looking into two factors: (i) the time required for upload/download of the generated widgets, and (ii) the efficiency of the client-side JS API. For an accurate measuring of the overhead, we need to distinguish between the four different HTTP methods (**GET**, **PUT**, **POST**, **DELETE**) and their associated payloads. From the evaluation perspective, only the **GET** response and the **POST/PUT** requests are interesting because these operations are responsible for the data transfer. The rest have no payload attached. We have measured the additional data transfer for HTTP requests and responses for each individual annotated form control and compared it to the size of plain XHTML form controls.

Table 1 shows the overhead of the HTTP **GET** requests grouped by widget components. The first column (XHTML) shows the size of the equivalent XHTML element rendered without annotations (in bytes), while the second column (XHTML+RDFa) shows the size of the XHTML+RDFa element (in bytes). The minimal (min) and maximal (max) values denote the size of the RaUL model required to generate the simplest or the richest (i.e., using all possible properties) widget. The third column shows the number of triples required in the backend and encoded in the resulting widget as annotations. Finally, the last column shows the overhead in percentages.

The evaluation results show that the *Widget* element adds around 2.5 times the overhead to a pure form container in XHTML. Since only one *Widget* is required for every form, the bytes presented in the table are in most cases only added once per page. An annotated *Textbox* takes about twice the size of the pure XHTML form and minimally requires two triples in the form model. Adding all properties (in total 7 RDF statements) to a *Textbox* causes an overhead of about 277%. The *Listbox* form control rendered in RDFa adds more than 3.5 times the size of the pure XHTML form. This is due to the fact that there is at least one *Listitem* associated with a *Listbox* which includes the reference to the value triple. As such, a *Listbox* needs at least five statements. Similar to the *Textbox* an annotated *Button* takes about twice the size of the pure XHTML control element and minimally requires two triples in the widget model. Again, adding all properties to a *Button* adds a considerable amount of space (more than four times the pure XHTML element) to the page due to the *Group* class which can be used to associate multiple buttons together. However, it also includes 8 triples in the annotation.

Table 2 shows the results of measuring the overhead of a POST/PUT request for each RaUL form control. We compare a standard form submission using the `application/x-www-form-urlencoded` MIME type (first column) to a RDF/XML submission (second column) via an Ajax request from the client-side JS API. Due to its stateless behaviour, in the case of a POST or PUT request, the client-side JS API always sends the entire form, as it is not aware of which elements need to be updated on the server. Since the representation of the payload can vary (as discussed in Sect. 3.1), the time required for the transaction will also vary. Among the three representations (RDF/XML, RDF/JSON and N3), we have evaluated only the RDF/XML representation as it is the most verbose one.

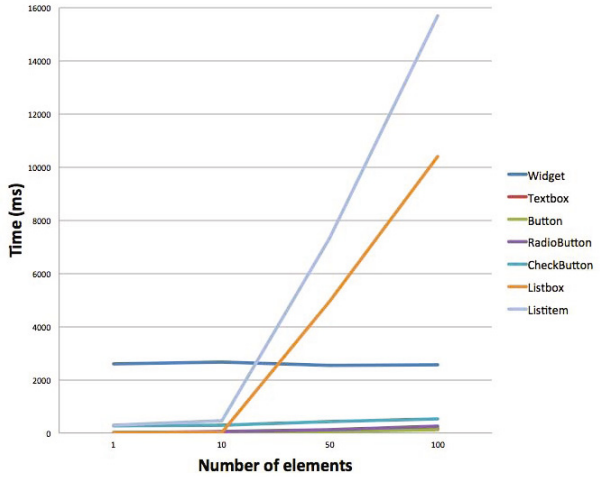
The results show that the serialisation of form elements in RDF/XML causes an overhead of minimal 87% and maximal 159%, depending on the element. The RDF/XML sent to the ActiveRaUL Web service is generated from the RDFa annotations by the client-side JS API. Although the overhead seems to be significant in size when all properties of a form control element are used, for the minimally required annotations the size of the POST or PUT request is less than double (except in the case of the *Listbox*) the size of a pure `application/x-www-form-urlencoded` request.

The second factor we have investigated in the RDFa overhead was the efficiency of the ActiveRaUL client-side JS API. For evaluation purposes we have measured the time required to parse the RDFa out of the XHTML via the Blackbird⁷ JavaScript library for profiling. Table 3 lists the average parse time from 10 runs for the form elements and our demo form pages. We can see that it takes less than 20 ms to parse the RDFa out of the XHTML for each type of form control. Evaluating the parsing time of our motivating example documents, the most expensive parsing is the user registration form due to the long listboxes containing items for the birthdate input.

⁷ <http://www.gscottolson.com/blackbirdjs/>

Table 3. ActiveRaUL client-side JS API performance

RaUL Element	Parsing (ms)	
	min	max
Widget	16	19
Textbox	15	18
Listbox	16	20
Button	16	19
Demo Page	Parsing (ms)	
Add User	465	
Add Product	97	
Buy Product	90	

Table 4. ActiveRaULProcessor performance

Widget/Model Generation Performance. In order to analyse the performance of the server-side widget/model generation we have measured the time required to generate an increasing number of widgets and widget elements. Fig. 4 shows the behaviour (time in ms. as average over 10 runs) of the service for 1, 10 and 100 generated widgets and widget elements (i.e., *Textbox*, *Button*, *Radiobutton*, *Checkbutton*, *Listbox* and *Listitem*). In the case of the *Listbox*, the number of generated *Listitems* per *Listbox* were 1, 10 and 100. The result shows, as expected, a linear scalability of all widget elements except for two cases: (i) the widget itself, which has a constant behaviour (as it does not have any variable elements in its construction), and (ii) the *Listbox* and *Listitems*, which also have a linear scalability, but with a much higher factor, due to the increased number of triples required in the model. In reality, reaching such numbers is highly improbable because they would produce an unusable user interface. Hence, from the usability perspective, the interesting range of values is between 10 and 30 widget elements. Here, the service performs very well (under 0.1 sec. to generate any element), with *Listitems* being the only element that pose some performance issues.

Discussion. We have shown that the content overhead required for the actual form–data binding is kept to a minimum and is, in principle, a result of using RDFa for the annotation of the semantic forms/widgets. As there are potentially many form controls that could be used in a widget, the user has the trade-off between the depth of the annotations and the size and bandwidth they consume. The more triples are used for a form control the richer its annotations. It also has to be noted that while the ActiveRaUL processor automatically adds RDFa annotations to XHTML pages, the practice of manually adding RDFa or Microformat annotations to XHTML pages is already a common and accepted

practice (despite its additional size). The additional size of RDFa annotations, as shown in our benchmarks, is acceptable in most cases. Further, adding the structure of the widget as semantic relations (i.e., RDF statements) yields the following benefits: (a) support for full machine understandable structured form data; (b) structured form data is encoded directly in the Web page and usable to any Semantic Web application; (c) the reuse of existing schemas in the modelling of the form data; (d) the automatic retrieval of form data from the Linked Data Web; and (e) the approach is fully browser agnostic via its rendering in XHTML + RDFa.

6 Related Work

Automatically generating forms from RDF ontologies, with the goal of interacting with Linked Data is a relatively new research topic. We are aware of some earlier attempts concerning form-based editing of RDF data [2], as well as mapping between RDF and forms [16]. None of these approaches propose a generic RESTful Web service to seamlessly combine data binding with the processing and generation of semantic annotations in Web applications.

In [22,4] the authors proposed a read/write-enabled Web of Data through utilising RDFS [13]. It provides a way for a Web browser to communicate structured updates to a SPARQL endpoint. RDFS consists of an XHTML form, annotated with the RDFS vocabulary in RDFa [1], and an RDFS processor that gleans the triples from the form to create a SPARQL Update [23] statement, which is then sent to a SPARQL endpoint. The difference to our approach is that RDFS does not propose an ontology for form controls and it is bound to a domain-agnostic model – that is, it describes the fields as key/value pairs – requiring a mapping from the domain ontology (FOAF, DC, SIOC, etc.). Dietzold [8] propose a JavaScript library, which provides a way for viewing and editing RDFa semantic content independently from the rest of the application. Further, they propose update and synchronisation methods based on automatic client requests. Their model is restricted to a fixed environment (the Wiki), and they only present the client in-memory modification of the model, but the execution of these atomic add/delete actions as performed in our case by ActiveRaUL is not discussed. Furthermore, there are other approaches such as SWEET [17], which deals with semantic annotations of Web APIs. Fresnel [9] provides a vocabulary to customise the rendering of RDF data in specific browser. At time of writing there are implementations for five browsers available.

Finally, the FAST gadget ontology (FGO)⁸ could provide an alternative to, or could be complemented by the RaUL ontology for modelling widgets and their underlying components. Currently it offers a high-level description of the organisation and information flow of gadgets in respect to screens and resources, the finest granularity mentioned being a `Form element`. In practice the `FGO:Form` and `Form element` can be specialised to the classes introduced by the RaUL ontology, thus providing a more comprehensive model of the domain.

⁸ http://kantenwerk.org/ontology/fast_gadget_content/fgo2011-02-11.html

7 Conclusion and Future Work

In this paper we have proposed a novel approach for interacting with Linked Open Data in a read/write manner. The approach uses the RaUL ontology for creating and managing semantic widgets and provides a RESTful Web service, ActiveRaUL, that is used to deploy, generate and retrieve RDFa annotated Web forms. The data – expressed as RDFa triples – is referenced from the RaUL *form model* via a data binding mechanism. The *form model* and *data model* parts make RaUL widgets more tractable and give meaning to the input values in form controls by referencing relations defined in standard Web ontologies. It also eases reuse of forms, since the underlying essential part of a form is no longer irretrievably bound to the page it is used in. We have developed a client-side JS API that parses RaUL annotated XHTML forms and performs a data binding based on the user input or data referenced in the Linked Data Web. The client-side JS API interacts with the ActiveRaUL service, a generic RESTful Web service enabling developers to deploy and manage their Web forms and the data model associated with these forms.

For future work we will focus on extending our approach to allow the modeling of complex page flows including validation and navigation and their automatic rendering in ActiveRaUL. Currently this process is defined in widget specific JavaScript, however, we intend to include it explicitly in the RDF model. In addition, we aim to improve the model generation algorithm, and evaluate it against a gold standard defined by ontology experts. Finally, we plan to develop algorithms to determine the type of form field to be used with object properties. Currently, object properties are handled by manually typing their URI in *Textbox* form controls which is arguably not intuitive enough.

References

1. Adida, B., et al.: RDFa in XHTML: Syntax and Processing. W3C Rec. W3C Semantic Web Deployment WG (October 14, 2008)
2. Baker, M.: RDF Forms (2003), <http://www.markbaker.ca/2003/05/RDF-Forms/>
3. Berners-Lee, T.: Linked Data. Design issues for the World Wide Web, W3C (2006), <http://www.w3.org/DesignIssues/LinkedData.XHTML>
4. Berners-Lee, T., et al.: On Integration Issues of Site-specific APIs into the Web Of Data. Tech. rep., DERI (2009)
5. Berners-Lee, T., Fischetti, M.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. Texere (2000)
6. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) 5(3) (2009)
7. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.91. Namespace document (November 2007), <http://xmlns.com/foaf/spec/>
8. Dietzold, S., Hellmann, S., Peklo, M.: Using JavaScript RDFa Widgets for Model/View Separation inside Read/Write Websites. In: Proceedings of the Scripting and Development for the Semantic Web Workshop (SFSW) (2008)
9. Fresnel, Display Vocabulary for RDF (2005), <http://www.w3.org/2005/04/fresnel-info/>

10. GWT – Google Web Toolkit (2010), <http://code.google.com/webtoolkit/>
11. Haller, A., Rosenberg, F.: A Semantic Web Enabled form model and restful service implementation. In: Proceedings of the Service-Oriented Computing and Applications Conference (SOCA) (2010)
12. Haller, A., Umbrich, J., Hausenblas, M.: RaUL: RDFa User Interface Language – A Data Processing Model for Web Applications. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 400–410. Springer, Heidelberg (2010)
13. Hausenblas, M.: RDFForms Vocabulary (2010), <http://rdfs.org/ns/rdforms/XHTML>
14. Hepp, M.: Web Ontology for e-Commerce (2010), <http://purl.org/goodrelations/>
15. Hobbs, J.R., Pan, F.: Time Ontology in OWL. W3C Working Draft (2006), <http://www.w3.org/2006/timezone>
16. de hOra, B.: Automated mapping between RDF and forms (2005), http://www.dehora.net/journal/2005/08/automated_mapping_between_rdf_and_forms_part_i.XHTML
17. Maleshkova, M., Pedrinaci, C., Domingue, J.: Semantic Annotation of Web APIs with SWEET. In: Proceedings of the Scripting and Development for the Semantic Web Workshop (SFSW) (2010)
18. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Tummarello, G.: Sindice.com: A document-oriented lookup index for open linked data. International Journal of Metadata, Semantics and Ontologies 3(1), 37–52 (2008)
19. Rubys on Rails (2010), <http://rubyonrails.org/>
20. Reenskaug, T.: The original mvc reports. Tech. rep. (February 2007)
21. Apache Struts (2010), <http://struts.apache.org/>
22. Ureche, O., Iqbal, A., Cyganiak, R., Hausenblas, M.: On Integration Issues of Site-Specific APIs into the Web of Data. In: Proceedings of the Semantics for the Rest of Us Workshop (SemRUs), Washington DC, USA (2009)
23. W3C: SPARQL 1.1 Update (2010), <http://www.w3.org/TR/sparql11-update/> Working Draft

RDFa²: Lightweight Semantic Enrichment for Hypertext Content

Xi Bai, Ewan Klein, and Dave Robertson

School of Informatics, University of Edinburgh, UK
xi.bai@ed.ac.uk,
{ewan,dr}@inf.ed.ac.uk

Abstract. RDFa is a syntactic format that allows RDF triples to be integrated into hypertext content of HTML/XHTML documents. Although a growing number of methods or tools have been designed attempting at generating or digesting RDFa, comparatively little work has been carried out on finding a generic solution for publishing existing RDF data sets with the RDFa serialisation format. This paper proposes a generic and lightweight approach to generating semantically-enriched hypertext content by embedding RDF triples derived from diverse provenances in terms of a concept of topic nodes which will be automatically recommended by our discovery algorithm. RDFa² is a proof-of-concept implementation for our approach and works as an online platform assisting Web content publishers in semi-automatically generating, personalising and curating pages with RDFa. RDFa² has been introduced and employed by students in a master level course and the experimental results as well as additional case studies indicate the validity of this approach to generating triple-embedded Web documents such as online profiles and vocabularies with little user intervention.

1 Introduction

The Semantic Web have been proposed as an extension of the Document Web where human-readable hypertext documents currently dominate. As part of the Semantic Web initiative to promote machine-readability of Web documents, RDFa (a W3C recommendation for more than two years) has been designed so that “authors can markup human-readable data with machine-readable indicators for browsers and other programs to interpret” [1]. An increasing number of tools for processing RDFa have been developed which leverage the existing range of techniques for processing standard RDF. Hundreds of thousands of FOAF [2] documents have been created (semi-)automatically or manually but due to the lack of human readability of RDF triples, many of RDF documents are hidden in repositories or behind SPARQL endpoints which are not accessible to users without expertise. On the other hand, a growing number of ready-to-reuse Linked Data sets have been published nowadays and a fully-fledged Linked Data application is likely to make use of data from more than one source. Automatic

¹ <http://xmlns.com/foaf/spec>

information processing and integration hence require Web content to be not only human-readable but also machine-readable. Although content publishers can publish a plain HTML page and point it via `meta` to an RDF document, since they are separate documents the availability of both documents may not be achieved at the same time and it is also difficult to avoid data duplication. While RDFa makes it easy for Web authors to manually add small amounts of semantic markups to XHTML documents, RDFa also offers the potential to transform pre-existing machine-readable data into human-readable format. So far, this has received relatively little attention. We propose a generic and lightweight approach to semi-automatically generating semantically-enriched hypertext content from existing RDF documents². A key ingredient, which we will describe in more detail below, involves the identification of one or more “topic nodes” in the RDF context(s) to guide the injection of RDFa into an XHTML template. A proof-of-concept of this approach has been implemented under the name RDFa² (*RDFa annotator*), and has been available as an online service³. RDFa² runs within standard Web browsers, and allows users to customise its output in two ways: either by modifying the generated data in an edit window (with on-the-fly preview) or by revising the generated XHTML template, which can be saved to local storage for future use.

The remainder of this paper is organised as follows. Section 2 reviews related work on processing RDFa. Section 3 describes the preprocessing of “RDF contexts” required by our approach. Section 4 proposes a hybrid topic-node discovery method based on weighted occurrences of nodes as well as heuristic properties and details how our approach can assist users in creating, customising and reusing Web content with RDFa. How RDFa²-assisted data integration is compliant with the standard RDF data model as well as the Linked Data [8] principles is also discussed in this section. Section 5 evaluates this approach by introducing our prototype to students in a master level course as well as case studies on republishing online vocabularies. Section 6 draws conclusions and indicates future work.

2 Related Work

SPARQLScript⁴ supports output templating which allows users to embed SPARQL query results into the dynamic generated Web pages via place holders. It could be used for dynamically embedding triples and users however need to learn this PHP-like script language and SPARQL. Fresnel [16] is a declarative language for rendering RDF content in specific browsers (as of writing this paper, it supports five browsers) but requires users to learn how to write lenses and formats which are two foundational concepts employed in this language. Tal4Rdf (T4R)⁵ is a template language for presenting RDF data into other formats such

² Here and in the rest of the paper, we take “RDF document” to subsume any documents containing (or embedding) RDF triples.

³ <http://demos.inf.ed.ac.uk:8836/rdfasquare>

⁴ <https://github.com/semsol/arc2/wiki/SPARQLScript>

⁵ <http://liris.cnrs.fr/~pchampin/t4r/>

as HTML, SVG, JSON, Atom and so on. However, it currently does not support the XHTML+RDFa representation of existing RDF data so it is difficult if not impossible for other users or developers to repurpose (e.g. mashup) the reformatted data on the generated Web pages. Therefore, although the above designed languages dedicated to RDF-embedded page generation are self-adaptive to updating of triples, the cost of creating an appropriate intermediate format (e.g., templates or lenses) is not much less than the cost of manually creating an XHTML+RDFa page.

*FOAFy*⁶ allows users to convert their FOAF documents into XHTML pages with RDFa automatically and it is however focused on the FOAF vocabulary⁷ only. Likewise, *FOAF.Viz*⁸ is a visualiser and relation explorer for FOAF documents. It provides RDF documents serialised in RDF/XML and Web pages containing RDFa with visualisations in which there is no embedded meta information. *GoodRelations*¹³ provides a *GoodRelations Annotator*⁹ as well as a *Rich Snippet Generator*¹⁰, both of which assist users in creating RDFa snippets for their businesses or products using the particular *GoodRelations* vocabulary. By filling slots in a provided template, a user will get an RDFa snippet generated using XSLT. Our approach is not domain specific and allows users to generate RDFa snippets using any vocabularies in the RDF data model.

*RDF2RDFa*¹⁴ also allows users to copy and paste RDFa snippets generated from input RDF documents. This copy-and-paste method makes the original RDF content transparent to users so it is difficult if not impossible that users can reuse human-readable content from the original RDF documents. *Drupal 7* allows developers to generate templates for associating RDFa with *Drupal* elements such as content types and fields¹⁰. It has, however, not offered a fine-grained solution for content publishers to easily associate RDFa with more open content lacking a generalised template. Our *RDFa²* provides users with the free-editing functionality and on-the-fly previews after content change so they can get the real WYSIWYG experience when starting the transformation.

3 Topic Nodes and Topic Trees

Our algorithm for transforming RDF documents to XHTML+RDFa pages is based on automatically generated templates. These templates are schematic XHTML documents, and have a tree structure. By contrast, the RDF data model is a graph, and cannot be converted to a single tree without duplicating re-entrant nodes. In order to overcome this problem, the conversion from RDF requires users to select a specific node in the RDF graph which then forms the root of a tree of RDF statements. Which node should the user choose? In practice, this seems to follow straightforwardly from the user's goals, namely to

⁶ <http://sw.joanneum.at:8080/foafr>

⁷ <http://xmlns.com/foaf/spec>

⁸ <http://foaf-visualizer.org>

⁹ <http://www.ebusiness-unibw.org/tools/goodrelations-annotator/en>

¹⁰ <http://www.stalsoft.com/grsnippetgen>

cases where the output of this approach is inserted as a snippet into a larger (X)HTML document. Taking Sir Tim Berners-Lee’s Twitter profile as an example, Figure 2 illustrates RDFa² generated an RDFa snippet from the triples obtained via SemanticTweet APIs¹¹ (it is needless to mention FOAF documents can be fed into this tool directly as snippet-generation seeds [3]). In the copy&paste way, this snippet is ready to be inserted into the <body> section of the homepage or exported as a separate Web page and notably, it can be further customised by publishers through adding more human-readable content.

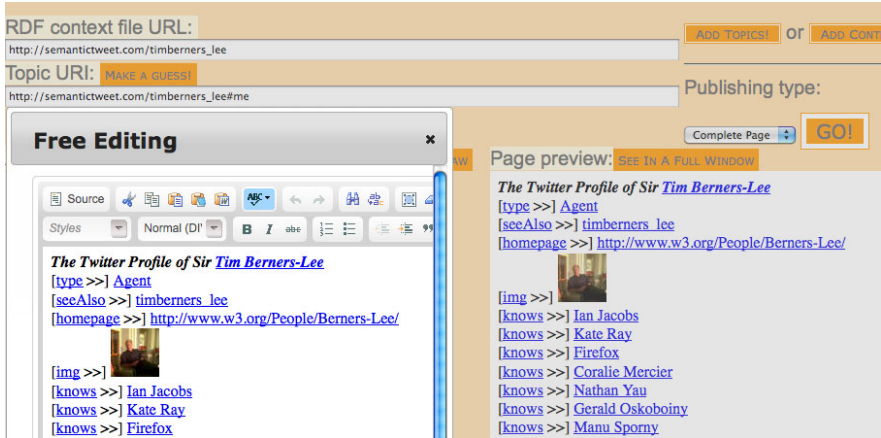


Fig. 2. Personalise the raw page generated by RDFa²

4 Embedded-Annotation Generation

Our approach to assisting users (e.g., Web content publishers) in generating annotations embedded in their hypertext content is detailed in this section. This approach has the ability to automatically discover a candidate set of topic nodes (from existing RDF contexts) which can be offered to the user thereafter and also supports federated integration in the sense that users can embed multiple topic nodes from multiple RDF contexts into a single Web page. Within the publishing process, publishers can revise suggested annotated blocks or raw pages in terms of their individual requirements. Moreover, templates are also provided via our approach and customised by publishers (and also stored, loaded and reused) if needed. Algorithm 1 describes the annotation generation process (generating the partial snippet for the subject topic tree) and will be further discussed in the following subsections. Likewise, the snippet generation corresponding to the object topic tree is not described here due to the space limitation but can be achieved by revising this algorithm and moving the topic node from the subject

¹¹ <http://semantictweet.com/>

Algorithm 1. RDFa Snippet Generation Algorithm (subject (topic) \mathcal{C} -tree)

Input: *topic_uri*, the URI of the topic node and *model*, the model containing triples in the current context.

Output: *rdfa_snippet*, the RDFa snippet representing the information about the inputted topic node.

```

begin
  def rdfa_snippet = getDIVHead(topic_uri);
  def sub_topic_tree = model.getStatementsBySubject(topic_uri);
  def properties = model.getUniquePropertiesBySubject(topic_uri);
  for each property in properties do
    def objects = sub_topic_tree.getObjectsByProperty(property);
    def prop_local_name = property.getLocalName();
    def prop_node_name = (property.getNamespace() + "_" +
      prop_local_name + "rel").replace("_", "dash");
    def prop_curie_name = model.getPrefix(property.getNamespace()) +
      ":" + prop_local_name;
    for each object in objects do
      if object.isLiteral() then
        rdfa_snippet += "<#if topic." + prop_node_name + "??>" +
          "<#list topic." + prop_node_name + "?keys as key>" +
          getLiteralStyle(prop_local_name, property.getURI()) + ... ;
      else
        def snippet = "";
        if object.isURIResource() && object.getURI().indexOf(".") !=
          -1 then
          def obj_uri = object.getURI();
          def expansion =
            obj_uri.subString(obj_uri.lastIndexOf("."));
          snippet += getSnippetByExpansion(prop_curie_name,
            prop_node_name);
        else
          snippet += "<a rel=" + prop_curie_name + ""
            href='${topic.'" + prop_node_name + "[key].uri}'
            onclick='return false;'>${topic.'" + prop_node_name +
            "[key].uri}</a></span><br/>";
          rdfa_snippet += "<#if topic." + prop_node_name + "??>" +
            "<#list topic." + prop_node_name + "?keys as key>" + "<#
            if topic." + prop_node_name + "[key].uri??>" +
            getResourceStyle(prop_local_name, property.getURI(), true) +
            snippet + "<#if><#list><#if>";
        return rdfa_snippet;
  end

```

position to the object position. For the mashup purpose, the embedded RDF triples can be harvested and serialised in several formats such as Notation3 (N3) [7], RDF/XML [6], N-Triples [12] and Turtle [5].

4.1 Topic-Node Discovery

In the preceding section, we assumed that topic nodes will be selected by the user. However, this requires the user to understand the basic syntax of the RDF

context inside which these nodes are represented. One way of automatically identifying topic nodes in a given RDF context is to query the document for URIs with properties that are diagnostic of topic-hood, such as `foaf:primaryTopic` or `foaf:maker` in FOAF files. However, not all RDF documents contain such properties, and even in FOAF files which do employ them, they do not always take semantically appropriate values. Consequently, topic nodes cannot reliably be detected just in terms of the semantics of statements in the RDF context itself. Xiang et al. compared five measurements from three categories (degree centrality, shortest-path-based centrality and eigenvector centrality) for automatically summarising ontologies in a topic-independent manner and their interesting evaluation showed that weighted in-degree centrality measures and several eigenvector centralities all have good performance on ontology summarisation [17]. As analysed in [4], for the case that the target RDF documents mix up ontology-related triples and individual-related triples, the above topic-free measurements may be affected by unforeseen noise nodes. Moreover, each property could have a corresponding inverse property so it is difficult if not impossible to draw a conclusion that an RDF node's in-degree (or out-degree) prioritises its out-degree (or in-degree). In this paper, we propose an improved algorithm for semi-automatically discovering and recommending topic nodes. Since the RDF data model is a directed graph and nodes are connected to one another through directed edges, one solution for discovering the topic node is based on node connectivity. In other words, the more edges (outgoing or incoming) a node has, the more important it is likely to be. In order to maximise the accuracy of this heuristic, our algorithm selects the top n most highly connected URIs and offers them to users for subsequent confirmation¹². Perhaps not surprisingly, this algorithm works especially well for RDF documents such as FOAF files that usually do have a central topic.

When a user inputs the URI of a resource that she wants to integrate into her Web page, together with an RDF context, RDFa² will query this context with the selected URI for all statements in which the URI is either subject or object. From this set, a subject (respectively, object) topic tree will automatically be selected if it exists. Its root will be the topic node and its corresponding properties and values will be stored in other nodes or leaves. Then the user can refer to any information about this topic node using the path structure `root.predicate.values[key].[resource]` or `root.predicate.values[key].[literal]` in the template which will be discussed in Subsection 4.3. Here, `root` denotes the resource currently being integrated; `predicate` denotes a specific property with which this resource is associated; and `values` is a list that stores the values of a property (since some properties may have multiple values). The screenshot in Figure 3 illustrates how topic nodes derived from an RDF context (Sir Tim Berners-Lee's twitter profile in RDF) were discovered and the most important URI in this context was shown at the top of the recommendation list.

¹² The value of n can be any reasonable integer. Although RDFa² takes n to 10, by default it only just shows the top three URIs to users. It is also worth noting that blank nodes are filtered out from the set of candidates.

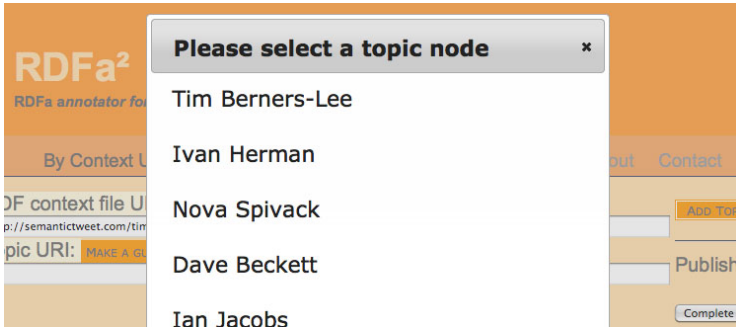


Fig. 3. Screenshot for discovering the topic node from an RDF context

4.2 Federated-Annotation Generation

We do not want to exclude the possibility of the user selecting more than one topic node from a given RDF context. For example, a user may wish to render the FOAF document vocabulary (i.e., encoded as a set of RDF statements) as XHTML, and in this use case, all of the nodes `foaf:Person`, `foaf:Agent` and `foaf:Document`, for example, should be treated as topics. We can use multiple templates to help the user achieve this goal. Once a user selects a temporary topic node, a hash tree, a template and an XHTML+RDFa page will be generated based on node occurrences. Meanwhile, the relevant NSs are also grouped and displayed on the final page. Thereafter, the generated XHTML+RDFa snippets will be automatically combined into a single snippet.

It is not uncommon that users publish an XHTML+RDFa page using triples from different RDF sources (or in our terminology, from different contexts). We can accommodate this in a way similar to our approach to dealing with multiple topic nodes. Our approach supports federated integration by managing the NSs derived from different RDF documents separately and combining them at the final stage. However, it should be noted that different vocabularies do not necessarily employ the same QName prefix for a given NS. *prefix.cc* (*PCC*)¹³ alleviates the issue that RDF documents involve different prefixes indicating the same NS or the same prefix indicating more than one NS by allowing users to look up the collected NSs on *PCC* and vote for their favourite ones. Nevertheless, it is difficult if not impossible to stop people from using ambiguous prefixes. Our approach can automatically detect if a prefix is ambiguous across a set of contexts, and will synthesise new prefixes to ensure disambiguation by generating different prefixes as substitutions. Moreover, many of the NSs in the original RDF context set are unused in the final XHTML+RDFa Web pages. In order to avoid an unnecessary burden on browsers rendering the page, the NSs which are not used in the user's RDF-embedded Web page will be automatically excluded. It is notable that RDFa 1.1 harnesses `@profile` to come over the lengthy declaration

¹³ <http://prefix.cc>

RDF context file URL:	<input type="text" value="http://semantictweet.com/timberners_lee"/>
Topic URI: MAKE A GUESS!	<input type="text" value="http://semantictweet.com/timberners_lee#me"/>
RDF context file URL:	<input type="text" value="http://semantictweet.com/ivan_herman"/>
Topic URI: MAKE A GUESS!	<input type="text" value="http://semantictweet.com/ivan_herman#me"/>
RDF context file URL:	<input type="text" value="http://semantictweet.com/novaspihack"/>
Topic URI: MAKE A GUESS!	<input type="text" value="http://semantictweet.com/novaspihack#me"/>

Fig. 4. Screenshot for generating annotation from multiple contexts

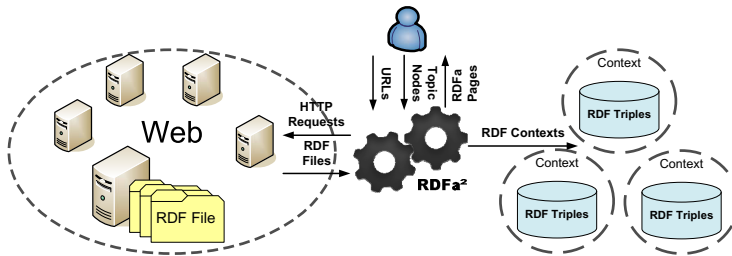


Fig. 5. Context-based federated integration

of NS prefixes recommended in RDFa 1.0 and this can be also used for avoiding possible ambiguous prefixes to some extent. Figure 4 illustrates the generation of a triple-embedded Web page by combining triples derived from three different Twitter profiles (contexts).

Figure 5 illustrates how our approach assists users in creating Web pages annotated with RDF triples derived from different data sources. Users inform RDFa² of the target in one or more RDF contexts by providing one or more URLs. These documents will be retrieved on the fly and each of them forms an RDF context. After the topic nodes are selected, triples related to them will be extracted. Finally, the page with RDFa annotation will be sent back to users.

4.3 Customisation and Template Reuse

One of the primary functions of our approach is to automatically carry out a template-based transformation of RDF to XHTML+RDFa. However, the result of the transformation will almost certainly not be in the precise form required by users, and consequently it is important to allow users to further edit the output. The RDFa² interface provides the user with both a rendered preview and the source code of the generated XHTML+RDFa. Users without expertise in

RDF(a) can modify the output by clicking and editing elements on the preview page or editing the content in the WYSIWYG way as shown in Figure 2. More experienced users can edit the page source and check its preview but it is recommended that revisions are limited to the text nodes of the page since manually edited RDFa needs revalidation.

When users deal with a great number of RDF documents of the same type (e.g., all of them are FOAF documents), they may have to carry similar or even identical manual revisions for each document processed by RDFa². To avoid this unnecessary effort, we provide users with another way of personalising the RDFa-embedded web pages by letting them revise the templates. Each transformation will generate a template and this template will be returned before being applied to the RDF context. A basic template is generated using placeholders of the kind standardly offered by template tools (e.g., FreeMarker¹⁴ applied here). Each placeholder indicates a piece of information which will be extracted during the transformation process (e.g., `personal.firstname` and `personal.lastname` are two placeholders which will be replaced with the first name and the last name of a particular person, respectively). As long as a template is generated, a hash tree that stores the data about the topic nodes is also generated, based on the RDF context: we call this an *intermediate tree*. The structure of the intermediate tree evolved from the structure of the topic tree but is more friendly to templating. Figure 6 shows the excerpt of a generated template that will be used for displaying all the people connected to the selected topic node via `foaf:knows`.

The triples taking the topic node as subjects or objects may take literals or other resources as their objects. Both of these two cases have to be taken into consideration before the template is generated. If the object is a literal, it will be enclosed within an HTML tag with `@property` (`@ATTRIBUTE` is used hereafter for denoting a tag's attribute in terms of the XPath syntax) indicating the predicate attached with this object. If the object is a resource, it will be enclosed within by an HTML tag with `@resource` taking this object as its value and `@rel` indicating the predicate attached to this object. As mentioned in Section 3, the text value



Fig. 6. Excerpt of a generated template for displaying known people

¹⁴ <http://freemarker.org>

of this tag node will be the preferred label (if exists) of the resource rather than its URI. This complies with the modelling pattern introduced in [11] as well.

4.4 Self-adaptability and Reflections on RDF Features

Our approach queries the RDF context using SPARQL with the topic node either given by the user or discovered by the topic recommender semiautomatically, which has been discussed above. The result will be used for replacing the pre-generated placeholders inside templates. In a specific RDF vocabulary, some properties may be defined as functional properties (e.g., `foaf:gender` and `foaf:primaryTopic` in FOAF). Each of them only takes one object or one literal as its value. Other properties (e.g., `foaf:maker` and `foaf:member`) may take more than one object or literal as their values. The SPARQL query results are grouped in terms of properties. With respect to the evolution of an RDF vocabulary, new classes or properties may be involved and some classes or properties may be deprecated. Since templates are created and applied on the fly and always based on the given vocabularies (RDF contexts), the above evolution will be transparent to users. For some of them who want to reuse their templates, their existing templates can be merged with the newly generated ones. A few manual reconciling work on these two kinds of templates might be involved within this process.

According to [1], `@resource` and `@href` can be used for hooking the object of an RDF triple. The value of the former is a URI which is "not intended to be clickable" and normally denotes a non-information resource while the value of the latter is a URI which normally denotes a information resource. The minters of non-clickable URIs need to provide relevant information resources as these URIs' representations [15]. RDFa² currently assumes each non-information resource has an informational representation and by clicking it, users will be redirected to another information resource associated with it. Thus, either information resources or non-information resources will be wrapped in `<a>` tags and attached to `@href` rather than `@resource` here. Since `@href` supports only URIs, the object of each RDF triple will not be expressed in CURIE (a generic, abbreviated syntax for expressing URIs) syntax in the final page. With respect to BNodes, the labelling property (if exists) and corresponding value surrounding a specific BNode in the original RDF context will be used as the representation. Nevertheless, users are recommended not to use BNodes when publishing Linked Data on the Web [9].

4.5 Linking Annotations to the LOD Cloud

There is one step to go before RDF triples are injected into Web pages because these embedded triples may otherwise cause provenance and trust issues. RDF statements are focused on describing who said what but statements themselves may or may not be true. Additionally, the licence is another thing that should not be ignored especially when users attempt to reuse data by other data providers. Therefore, the enriched documents need to be associated with

provenance information and linked to the Linked Open Data (LOD) Cloud¹⁵. Here, we use the Vocabulary of Interlinked Datasets (void)¹⁶ to describe the relationships between the annotations and the RDF contexts from which the harnessed triples are derived. This vocabulary has been used here due to its simplicity and concision but alternative linked dataset vocabularies could be applied here for the same purpose. Suppose the URI of the topic node is denoted by T_{uri} and the URI of the RDF context (provenance) is denoted by C_{uri} . An XHTML+RDFa snippet will be automatically generated to describe the provenance of T_{uri} as follows:

```
<div about=" $T_{uri}$ " xmlns:void="http://rdfs.org/ns/void#"
      xmlns:dcterms="http://purl.org/dc/terms/">
  <span rel="dcterms:isPartOf">
    <span typeof="void:Dataset">
      <span rel="void:dataDump" resource=" $C_{uri}$ " />
    </span>
  </span>
</div>
```

5 Experiment and Use-Case Analysis

We experiment with our approach and show the preliminary performance of RDFa², which has been deployed on the Apache Tomcat server installed on a PC with a Pentium[®]D 3.00GHz × 2 CPU and 1 GB RAM.

Online profiles have been widely used by various Web sites for managing user identification. FOAF is currently one of the most widely used profile vocabulary for RDF on the Web. RDFa² can help users inject their FOAF triples into their online profile documents such as homepages. Our experiment first involved asking students who participated in a masters level course on Semantic Web technologies to use RDFa² to publish their own profiles (FOAF documents) along with information about their favourite actors/actresses denoted by URIs minted and curated on DBPedia¹⁶ and submit URLs of these documents to Sindice¹⁷. Fresnel and SPARQLScript were also introduced during the course as alternatives. In total, 64 students participated in this experiment and 60 of them successfully submitted their reports. On a public server, each student has been allocated personal space to store his or her own documents (e.g., the homepage). By searching documents of type XHTML+RDFa on Sindice with the domain name of the above homepage server as well as students' matriculation numbers, we found that 58 out of 60 students finally published their RDFa profiles and also successfully managed to make Sindice index them. 93.33% of students chose the first topic nodes (at the top of the generated topic-node lists) recommended by our topic-node discovery algorithm as their priority within the

¹⁵ <http://linkeddata.org>

¹⁶ <http://dbpedia.org/>

¹⁷ <http://www.sindice.com/main/submit>

process of RDFa snippet generation while two students chose the second topic nodes as their priority. Based on their feedback, RDFa² made straightforward the process for generating triple-embedded Web pages from existing RDF data sets and Fresnel as well as SPARQLScript are however more flexible for users with expertise on specific languages as well as RDFa itself to customise pages.

We also collected 324 FOAF documents (without considering dead links declared already on the homepage) from *FOAFBulletinBoard (FBB)*¹⁸ and 146 FOAF documents from *W3C RDF Harvester Starting Point (WRDFHSP)*¹⁹ respectively. These two sites are separate Wikis for bootstrapping a community in which any users are allowed to contribute FOAF documents collaboratively. Finally we got 149 and 63 valid FOAF documents in total from *FBB* and *WRDFHSP* respectively and republished them with RDFa² thereafter. Table 1 shows the results of retrievals of FOAF documents collected from the above two sites.

Table 1. FOAF document retrieval on *FBB* and *WRDFHSP*

Dataset		403	404	406	503	invalid	<i>UC</i>	<i>UKH</i>	<i>OOM</i>	valid
<i>FBB</i>	<i>N</i>	6	72	9	1	59	9	18	1	149
	<i>P</i>	2.74%	22.60%	1.37%	.68%	12.33%	6.16%	8.90%	2.05%	43.15%
<i>WRDFHSP</i>	<i>N</i>	4	33	2	1	18	9	13	3	63
	<i>P</i>	1.85%	22.22%	2.78%	.31%	18.21%	2.78%	5.56%	.31%	45.99%

In this table, by “invalid”, we mean these URLs indicate FOAF documents published in an unrecommended way (e.g., FOAF documents have syntax errors or involve deprecated syntax which can not be accepted by the up-to-date RDF parser). Besides, 403, 404, 406 and 503 denotes the numbers of retrievals that caused HTTP 403, 404, 406 and 503 errors respectively. *UC* denotes the numbers of retrievals that caused unconnected errors and *UKH* denotes the ones caused unknown-host errors. A few FOAF documents contain too many triples to be loaded into our parser and the number of these documents is denoted by *OOM*. *N* and *P* denote the number of retrievals and the corresponding percentage respectively. We see in this table that 54.01% of documents on *FBB* and 56.85% of documents on *WRDFHSP* do not contain valid FOAF information. Due to the space limitation, the time costs of these transformations dedicated to the above two sites are not listed here (see in [3]). On average, 98.58% of valid documents on both sites can be transformed via RDFa² within 3 seconds.

Besides online profiles, our approach can be also used for republishing RDF vocabularies on normal Web pages. Since there is no central repository of vocabularies on the Semantic Web²⁰, we collected RDF vocabularies in terms of NSs collected from *Ping The Semantic Web (PTSW)*²¹ and *PCC* respectively. At the time of writing this paper, there were 825 NSs recorded by *PTSW*. Since URLs of corresponding vocabularies can not be inferred with the NS URIs (publishers

¹⁸ <http://wiki.foaf-project.org/w/FOAFBulletinBoard>

¹⁹ <http://esw.w3.org/AnRdfHarvesterStartingPoint>

²⁰ http://vocamp.org/wiki/Where_to_find_vocabularies

²¹ <http://pingthesemanticweb.com/>

may use rewriting rules to manage the URI of the NS and the URL of the vocabulary document separately), we can only use these NS URIs to do the vocabulary retrievals via HTTP requests as well as content negotiations. Finally we got 249 vocabularies in total and republished them with RDFa² afterward. We did the same experiment on *PCC* as well and 349 NS URIs were obtained at the time of writing. We got 165 vocabularies in total and republished them with RDFa². Table 2 shows the results of retrievals of RDF vocabularies in terms of name spaces from *PTSW* and *PCC*.

Table 2. RDF vocabulary retrieval in terms of name spaces from *PTSW* and *PCC*

Dataset		400	401	403	404	406	408	500	503
<i>PTSW</i>	N	1	8	8	180	14	1	2	33
	P	.12%	.97%	.97%	21.82%	1.70%	.12%	.24%	4.00%
<i>PCC</i>	N	-	-	3	35	26	-	-	-
	P	-	-	.86%	10.03%	7.45%	-	-	-
Dataset	invalid	<i>UC</i>	<i>UKH</i>	<i>OOM</i>	valid	-	-	-	
<i>PTSW</i>	N	258	29	39	3	249	-	-	-
	P	31.27%	3.52%	4.73%	.36%	30.18%	-	-	-
<i>PCC</i>	N	108	4	7	1	165	-	-	-
	P	30.95%	1.15%	2.01%	.29%	47.28%	-	-	-

In Table 2, by “invalid”, we mean these NS URIs were not valid HTTP URIs or moved temporarily/permanently or indicate vocabularies which were actually not published in the RDF data model or serialised in syntaxes apart from RDF/XML which we just took into consideration in our experiment or published in an unrecommended way (e.g., RDF codes were attached in the comment section of the HTML document or have the error or deprecated syntax which can not be accepted by the RDF parser). Besides, this table contains more columns than Table 1 because more types of HTTP errors occurred within the process of retrieving vocabularies from these two sites. Within the retrieving process, 69.82% vocabularies on *PTSW* and 52.72% vocabularies on *PCC* can not be retrieved by dereferencing their NS URIs.

Figure 7 depicts the costs of time on republishing RDF vocabularies collected in terms of NS URIs from *PTSW* and *PCC* respectively on XHTML pages with embedded RDFa. From this figure, 20 out of 249 vocabularies on *PTSW* as well as 5 out of 165 vocabularies on *PCC* cost around 16ms and there were no results generated after the running of the program. The reason for this is because these 25 vocabularies in total do not contain any class or property declarations. On average, 93.96% of successfully retrieved vocabularies can be transformed via RDFa² within 3 seconds. RDFa²'s performance on dealing with documents containing a large number of triples is related to the employed third party RDF parser and the memory allocated for running the RDFa snippet generation program. It is however not recommended to use RDFa² to process large RDF documents since this may lead to Web pages with massive content in the end, which will affect the readability and bring the overhead onto browsers when being rendered.

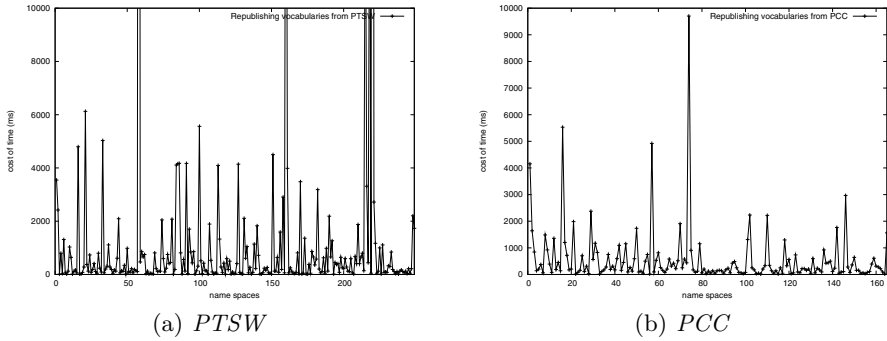


Fig. 7. Republishing RDF vocabularies on *PTSW* and *PCC*

6 Conclusions

A generic and lightweight approach is proposed to assisting content publishers in generating semantically-enriched hypertext content with triples derived from existing distributed RDF (or RDFa) documents (repositories). The experiment and the use-case analysis show that this approach can help publishers republish their triples in the RDFa serialisation with little human intervention. Nowadays, more and more Linked Data applications have come up and began to employ data from more than one source (or contexts in this paper) and RDFa² helps users harness resources from different contexts and potential conflict NSs declarations will be automatically handled. A property of a specific triple could be a topic node as well and a method needs to be carefully designed to synthesise related triples and topic trees. The support in this will be further investigated and integrated in the next step. For other hypertext-friendly formats of embedded metadata such as Microformats²² or Microdata²³, our approach can be employed as well for generating Web pages with those formats from existing RDF data sets. At the time of writing, XHTML+RDFa 1.1²⁴ and HTML+RDFa 1.1²⁵ W3C working drafts were released and have been improved in progress, our goal is to make our approach harness new features compatible with the up-coming W3C recommendations.

References

1. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and processing. W3C recommendation (2008), <http://www.w3.org/TR/rdfa-syntax>

²² <http://microformats.org/about>

²³ <http://www.w3.org/TR/microdata>

²⁴ <http://www.w3.org/TR/2010/WD-xhtml-rdfa-20101109>

²⁵ <http://www.w3.org/TR/rdfa-in-html>

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: Proc. WWW Workshop on LDOW 2009 (2009)
3. Bai, X.: Addressing the RDFa Publishing Bottleneck. In: Proc. WWW (Companion Volume) 2011, pp. 331–336. ACM Press (2011)
4. Bai, X., Delbru, R., Tummarello, G.: RDF Snippets for Semantic Web Search Engines. In: Meersman, R., Tari, Z. (eds.) OTM 2008. LNCS, vol. 5332, pp. 1304–1318. Springer, Heidelberg (2008)
5. Beckett, D., Berners-Lee, T.: Turtle - terse RDF triple language, W3C team submission (2008), <http://www.w3.org/TeamSubmission/turtle>
6. Beckett, D., McBride, B.: RDF/XML syntax specification (revised), W3C recommendation (2004), <http://www.w3.org/TR/REC-rdf-syntax>
7. Berners-Lee, T.: Notation 3 specification, W3C design issues (1998), <http://www.w3.org/DesignIssues/Notation3.html>
8. Berners-Lee, T.: Linked Data (2006), <http://www.w3.org/DesignIssues/LinkedData.html>
9. Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web (2007), <http://www4.wiwi.fu-berlin.de/bizer/pub/LinkedDataTutorial>
10. Corlosquet, S., Delbru, R., Polleres, A., Decker, S.: Produce and Consume Linked Data with Drupal! In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 763–778. Springer, Heidelberg (2009)
11. Dodds, L., Davis, I.: Linked Data patterns - a pattern catalogue for modelling, publishing, and consuming Linked Data (2010), <http://patterns.dataincubator.org/book>
12. Grant, J., Beckett, D., McBride, B.: RDF test cases, W3C recommendation (2004), <http://www.w3.org/TR/rdf-testcases>
13. Hepp, M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 329–346. Springer, Heidelberg (2008)
14. Hepp, M., García, R., Radinger, A.: RDF²RDFa: Turning RDF into snippets for copy-and-paste. In: Proc. Posters and Demonstrations Track on ISWC 2009 (2009)
15. Lewis, R.: Dereferencing HTTP URIs (2007), <http://www.w3.org/2001/tag/doc/httpRange-14/HttpRange-14.html>
16. Pietriga, E., Bizer, C., Karger, D.R., Lee, R.: Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 158–171. Springer, Heidelberg (2006)
17. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on RDF sentence graph. In: Proc. WWW 2007, pp. 707–716. ACM Press (2007)

GoRelations: An Intuitive Query System for DBpedia^{*}

Lushan Han, Tim Finin, and Anupam Joshi

University of Maryland, Baltimore County, USA
{lushan1, finin, joshi}@umbc.edu

Abstract. Although a formal query language, SPARQL, is available for accessing DBpedia, it remains challenging for users to query the knowledge unless they are familiar with the syntax of SPARQL and the underlying ontology. We have developed both an intuitive *semantic graph* notation or interface allowing one to pose a query by annotating a graph with natural language terms denoting entities and relations and a system that automatically translates the query into SPARQL to produce an answer. Our key contributions are the robust techniques, combining statistical association and semantic similarity, that map user terms to the most appropriate classes and properties used in the DBpedia Ontology.

Keywords: Intuitive Query, Ontology Mapping, Statistical Association.

1 Introduction

The growth of Linked Open Data (LOD) has made large amounts of Semantic Web data available. DBpedia [1] is an important example, since it is a key LOD integrating component serving as a microcosm for larger, evolving LOD collections. Most of DBpedia data is extracted from Wikipedia infoboxes, which are designed by different communities and edited by individuals, making infobox names and attributes largely heterogeneous. DBpedia addressed this problem by manually mapping infoboxes describing the same type of thing to the same DBpedia ontology class and synonymous attributes to the same ontology property, resulting in 272 classes and 1,300 properties as of DBpedia 3.6. However, heterogeneity remains a problem, especially for properties due to their large number and the difficulty of dealing with context-dependent mappings.

Although SPARQL is available for querying DBpedia, it remains difficult for typical Web users to query its knowledge base (KB). They must simultaneously master SPARQL, explore the large number of ontology terms, and deal with term heterogeneity. To simplify access, systems like True Knowledge and PowerAqua [2] provide natural language interfaces (NLIs) receiving users' queries and automatically finding answers in their underlying KBs. While they are good at answering simple questions (*Who were Richard Nixon's children?* and *Who*

^{*} This research was supported in part by a gift from Microsoft, NSF award IIS-0326460 and the Human Language Technology Center of Excellence.

did Julie Nixon marry?) they often fail at slightly more complex ones (*Who did President Nixon’s children marry?*) due to difficulties in understanding complex natural language (NL) questions.

GoRelations (*Graph of Relations*) is an open domain, intuitive query system that is easy to learn and use. It has two components: a *semantic graph interface* (SGI) allowing users to ask queries with complex relations and an effective and efficient automatic translator mapping the semantic graph into a corresponding SPARQL query to produce an answer. While our approach is tailored to DBpedia, the idea is generic and can be adapted to other LOD collections.

2 Semantic Graph Interface

Our interface uses an intuitive concept we call a *semantic graph* (SG) as a representation allowing a user to express a question or description. A semantic graph consists of nodes denoting entities and links representing binary relations between them. Each entity is described by two unrestricted terms: its name or value and its concept in the query context. Figure 1 shows an example of a semantic graph that comprises three entities: a place, person and book, which are linked by two relations, *born in* and *author*. Users flag entities they want to see in the results with a ‘?’ and those they do not with a ‘*’.

Terms for concepts can be nouns (*book*) or simple noun phrases (*soccer club*) and relations can be references as verbs (*wrote*), prepositions (*in*), nouns (*author*) or simple phrases (*born in*). Users are free to name concepts and relations in their own ways as in composing a NL question with a current constraint that concept names should be at most two words and relation names three. One reason for the constraint is that most class and property names in the underlying DBpedia Ontology are no longer than two words and three words, respectively. A more fundamental reason is to encourage users to decompose their queries into simple entity and relation terms rather than use complex linguistic descriptions.

The value of entities can be something other than a name, for example, a number or date. If the value of an entity is a number, “Number” should be used as the entity’s concept. Numerical attributes such as population, area, height, and revenue can be thought of as either relations or concepts, but since *Number* is already used as a concept, we require them to be relations. We enforce this rule because in DBpedia’s ontology numerical attributes only have data types, which we uniformly treat as *Number* instances.

We circumvent the difficult task of understanding sentential semantics by asking users to directly supply the compositional relations between the lexical terms while users are not required to know a formal language and ontology.



Fig. 1. A SG for “Where was the author of the Adventures of Tom Sawyer born?”

3 Translation

We start by laying out a novel, three-step approach that maps terms in the semantic graph to ontology terms. The approach focuses on vocabulary or schema mapping, which is done without involving entities. We then discuss how to generate a SPARQL query from the mappings. Finally, we describe the ontology statistics and semantic similarity components used in the mapping approach.

3.1 Mapping Approach

Step One: Finding Semantically Similar Ontology Terms. For each concept or relation in the semantic graph, we generate a list of the k candidate ontology classes or properties that are most semantically similar. (See Section 3.4 for semantic similarity computation). A minimum similarity threshold, currently experimentally set at 0.1, is used to guarantee that all the terms have at least some similarity. If the relation is a very general term such as *in*, *has* and *from*, which we call “default relation”, we generate $\frac{k}{2}$ most semantically similar ontology properties to each of its connected concepts. We do so because using one concept’s name as relation name is the typical way to represent *has-relation* or *in-relation* in Wikipedia and because the semantics of a default relation is often conveyed in one of its connected concepts. The value k (currently 20) depends on the degree of heterogeneity in the underlying ontologies, how well semantic similarity measure is implemented, and the allowed computation time.

In Figure 2, candidate lists are generated for the five user terms in the semantic graph query. Classes starting with # are virtual classes¹, which we assign only half similarity to make them subordinate to native classes. Datatype properties are indicated by a starting @ character to distinguish them from object properties. Candidate terms are ranked by their similarity scores, which are displayed to the right of the terms. The example shows that our semantic similarity measure is well implemented but still has a large space to improve.

Step Two: Disambiguation. Each combination of ontology terms, with one term coming from one candidate list, is a potential interpretation of the user query, but some are reasonable and others not. Disambiguation in this context means, among all the interpretations, chooses the most reasonable ones. An intuitive measure of reasonableness is the degree to which the ontology terms,

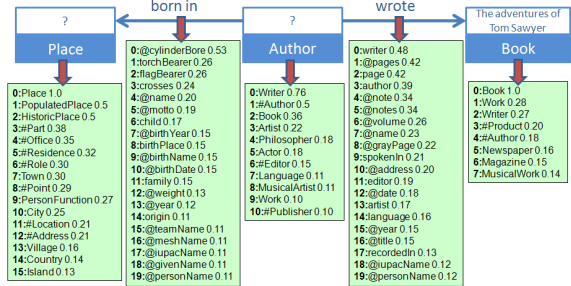


Fig. 2. Lists of candidate ontology terms

¹ They are inferred from the object properties (e.g. “publisher”). We create them to facilitate the mapping.

in one interpretation, associate in the way that their corresponding user terms connect in the semantic graph. DBpedia is a knowledge representation of the world’s facts made by humans and a semantic graph is a description of some facts about the world in the user’s mental model. Since both of them are mirrors of the world, they share an important feature – associations. Consider the example in Figure 2. In the query graph the relation *wrote* connects the two entities whose concepts under the query context are *Author* and *Book*. This implies that the relation *wrote* should have good associations with the concepts *Author* and *Book*. What should be reflected in the DBpedia Ontology is that the property corresponding to the relation *wrote* should also have good statistical associations with the classes corresponding to the concepts *Author* and *Book*.

Using association to resolve ambiguity is a common practice, as often seen in word sense disambiguation. However, many of previous researches only made use of coarse-grained association. That is, their contexts used for disambiguation are only a bag of words without considering the compositional structure of the knowledge in the sentences. With the coming of DBpedia, a large machine-readable KB, we are now able to compute fine-grained associations. We use Pointwise Mutual Information (PMI) [3] to compute statistical associations between classes and between classes and properties (See Section 3.3 for details).

If candidate ontology terms ideally contained all the near-synonyms, we could rely solely on their fine-grained associations for disambiguation. However, in practice many other related terms are also included and therefore the similarity of candidate ontology terms to the user terms is an important feature to identify correct interpretations. We experimentally found that by simply weighting their associations by their similarities we obtain a better disambiguation algorithm.

Below, we present a simple but novel disambiguation algorithm that exploits fine-grained associations. Suppose the query graph G has m links and n nodes. We need find a combination of m ontology properties p_1 to p_m , and n ontology classes, c_1 to c_n , from the space H of all interpretations that maximize the goodness or reasonableness of the mapping on the query graph G . This is computed as the summation of goodness of the mapping on each link L_i , i from 1 to m . More specifically,

$$\operatorname{argmax}_{p_1 \dots p_m \ c_1 \dots c_n \in H} \operatorname{goodness}(G) = \operatorname{argmax}_{p_1 \dots p_m \ c_1 \dots c_n \in H} \sum_{i=1}^m \operatorname{goodness}(L_i) \quad (1)$$

Note that the global optimal mapping on the whole graph is not necessarily composed of all the local optimal mappings on the individual links. Since a node can be involved in multiple links, the mapping decision on the node is affected by all the links it participates in. For example, the “Author” node in Figure 2 is involved in both the left link *born in* and the right link *wrote*. The local optimal mapping decision for “Author” from one link may be given up if it produces low goodness score on the other link. The same principle can be recursively spread to all other nodes and links in the entire graph. Therefore, our approach maps the semantic graph *jointly*.

Each link L_i is a tuple with three elements: subject concept S_i , relation R_i and object concept O_i . Let their corresponding ontology terms of current

interpretation be $c(S_i)$, $p(R_i)$ and $c(O_i)$. Before we compute the goodness of link L_i , we need first resolve the direction of the property $p(R_i)$ because $p(R_i)$ is semantically similar to R_i but they may have opposite directions. For example, the relation *wrote* in Figure 2 is semantically similar to the property *author* which, however, connects from *Book* to *Author*. We invent the statistical association measure $\overrightarrow{\text{PMI}}$ (see Section 3.3) to help determine the direction of $p(R_i)$. $\overrightarrow{\text{PMI}}$ measures statistical association between a class and a property. Unlike the traditional PMI, $\overrightarrow{\text{PMI}}$ also considers direction. $\overrightarrow{\text{PMI}}(\text{Class } c, \text{Property } p)$ measures the strength of association between c as subject and p as predicate whereas $\overrightarrow{\text{PMI}}(\text{Property } p, \text{Class } c)$ measures the strength of association between p as predicate and c as object. Whether the direction of $p(R_i)$ should be inverse to the one of R_i is decided in Formula 2.

$$\begin{aligned} &\text{If } [\overrightarrow{\text{PMI}}(c(O_i), p(R_i)) + \overrightarrow{\text{PMI}}(p(R_i), c(S_i))] \\ &\quad - [\overrightarrow{\text{PMI}}(c(S_i), p(R_i)) + \overrightarrow{\text{PMI}}(p(R_i), c(O_i))] > \alpha \\ &\quad \text{Then } S_i' = O_i, O_i' = S_i \\ &\quad \text{Else } S_i' = S_i, O_i' = O_i \end{aligned} \tag{2}$$

The association term $\overrightarrow{\text{PMI}}(c(O_i), p(R_i)) + \overrightarrow{\text{PMI}}(p(R_i), c(S_i))$ measures the degree of reasonableness of the inverse direction and the term $\overrightarrow{\text{PMI}}(c(S_i), p(R_i)) + \overrightarrow{\text{PMI}}(p(R_i), c(O_i))$ measures the degree of reasonableness of the original direction. If the inverse direction is much more reasonable than the original direction, we inverse the direction by switching the classes that $p(R_i)$ connects; otherwise we respect the original direction. Currently, the reverse threshold α is 2.0. The Formula 2 worked very well empirically. Further verifying the formula and the hypothesis behind it using statistical techniques is one of our future work.

Finally, the goodness on link L_i is the sum of three pairwise associations: the directed association from subject class $c(S_i')$ to property $p(R_i)$, the directed association from property $p(R_i)$ to object class $c(O_i')$, and the undirected association between subject class $c(S_i')$ and object class $c(O_i')$, all weighted by semantic similarities between ontology terms and their corresponding user terms.

$$\begin{aligned} \text{goodness}(L_i) = &\max(\overrightarrow{\text{PMI}}(c(S_i'), p(R_i)) \cdot \text{sim}(S_i', c(S_i')) \cdot \text{sim}(R_i, p(R_i)) \\ &+ \overrightarrow{\text{PMI}}(p(R_i), c(O_i')) \cdot \text{sim}(O_i', c(O_i')) \cdot \text{sim}(R_i, p(R_i)), \beta) \\ &+ \text{PMI}(c(S_i'), c(O_i')) \cdot \text{sim}(S_i', c(S_i')) \cdot \text{sim}(O_i', c(O_i')) \end{aligned} \tag{3}$$

We use a parameter β (currently 0.05) to shield the effect of the first two pairwise terms on the occasions when the property $p(R_i)$ fits too poorly with its two classes to be a valid choice (their value can be negative infinite). For these cases, the goodness is determined only by the last pairwise term.

Of the best interpretation yielded by the disambiguation algorithm for the example in Figure 2, the concepts *Place*, *Author* and *Book* are mapped to the ontology classes *Place*, *Writer* and *Book* respectively. The relation *born in* and *wrote* are mapped to the ontology property *birthPlace* and *author* with direction

unchanged and reversed respectively. Although the property *writer* has larger semantic similarity to the user term *wrote* than the property *author*, it is not selected because it is mainly used for describing films or songs but rarely for books. Although the property *birthPlace* has relatively low similarity with *born in*, it is selected because all the candidate terms with higher similarity do not associate well with the classes similar to the concept *Place* and *Author*.

The computation complexity of a straightforward disambiguation algorithm is $O(k^{n+m})$ simply because the total number of interpretations is k^{n+m} . However, we can significantly reduce this complexity to $O(k^n \frac{m}{n} k)$ by exploiting locality because the optimal mapping choice of a property can be determined locally when the two classes it links are fixed.

Step Three: Refinement. The best interpretation typically gives us the most appropriate classes and properties that the user terms can map to. However, for properties there can be some issues requiring additional work. First, the concepts are mapped to correct classes but occasionally the relation connecting them cannot find any mapping. Second, although the disambiguated property is appropriate, sometimes it is not the major property used in the context. Because the concepts are already disambiguated, we can narrow down to the disambiguated context where we can have more information about all properties that actually connect the two known classes and their conditional probabilities. In the case of a missing property, we map the relation to its most semantically similar property in the context. In the case of a minor property, we add other properties in the context, which are less similar to the user relation than the disambiguated property but have much higher conditional probabilities.

3.2 SPARQL Generation

After users terms are disambiguated and mapped to appropriate ontology terms, the translation of a semantic graph query to SPARQL is straightforward. Figure 3 shows the SPARQL query produced for the semantic graph in Figure 2. Classes are used to type the instances, such as *?x a dbo:Writer*.

Properties are used to connect instances just as relations do for entities as in *?o*

dbo:author ?x. If a user relation is mapped to multiple properties, the SPARQL UNION operator is used to combine them. *bif:contains* is a virtuoso built-in text search function which find literals containing specified text.

```

PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?x, ?y WHERE {
  ?o a dbo:Book .
  ?o rdfs:label ?label0 .
  ?label0 bif:contains "'The adventures of Tom Sawyer"' .
  FILTER(lang(?label0) = "en") .
  ?x a dbo:Writer .
  ?y a dbo:Place .
  ?o dbo:author ?x .
  ?x dbo:birthPlace ?y . }

```

Fig. 3. SPARQL Query Generated

3.3 Ontology Statistics Component

GoRelations uses three kinds of ontology statistics: directed association between classes and properties, undirected association between classes themselves, and conditional probability of properties given two connected classes. Computing

these statistics requires information about the number of occurrences of a term and the number of co-occurrences of two or three terms in the universe consisting of all relations. In DBpedia, the universe is represented by the dataset *Ontology Infobox Properties*, which contains RDF triples describing all relations between instances, and the dataset *Ontology Infobox Types*, which provides all type definitions for the instances.

The example in Figure 4 explains how we count term occurrences and co-occurrences by observing one relation in the universe. On the left of the figure, we give an RDF triple describing a relation and the type definitions for the subject and object in the triple. On the right, we list the resulting co-occurrences and co-occurrences of terms. The directed co-occurrences are indicated by the arrow character \rightarrow between two terms, for example *Book* \rightarrow *author*. The occurrences of directed classes (e.g. *Book* \rightarrow) are counted separately from the occurrences of undirected classes (e.g. *Book*).

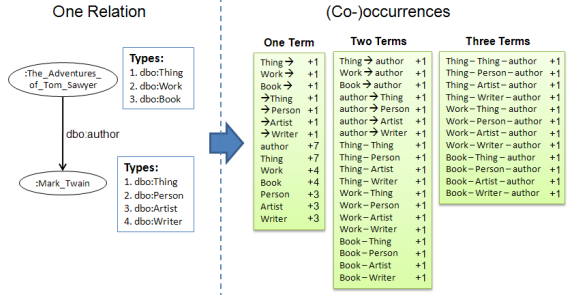


Fig. 4. An example for counting (co-)occurrences

PMI is used to measure the strength of statistical association between two terms. Equation 4 gives the PMI formula where f_{t_1} and f_{t_2} are the marginal occurrence counts of the two terms t_1 and t_2 and $f(t_1, t_2)$ is the co-occurrence count of t_1 and t_2 in the universe. N is a constant for the size of the universe. $\overrightarrow{\text{PMI}}$ is computed the same way as PMI except that its class term is directed.

$$\text{PMI}(t_1, t_2) \approx \log_e \left(\frac{f(t_1, t_2) \cdot N}{f_{t_1} \cdot f_{t_2}} \right) \quad (4)$$

3.4 Semantic Similarity Component

We assume that the semantic of a phrase is compositional on its component words and we apply an algorithm to compute semantic similarity between two phrases using word similarity. As in Mihalcea’s approach [4], we pair up words from two phrases in a way such that it maximizes the sum of word similarities of the resulting word-pairs. We differ, however, in that we do not allow a constituent word to participate in multiple pairs. The maximized sum of word similarities is further normalized by the number of words in the longer phrase to get the output similarity for two phrases.

Our word similarity measure is based on distributional similarity and latent semantic analysis, which is further enhanced using knowledge from WordNet. The distributional similarity approach that we use is described in [5] while we give higher similarity to word pairs which are in the same WordNet synset or one of which is the immediate hypernym of the other.

4 Evaluation

To evaluate ontology-based QA systems, the 2011 Workshop on Question Answering over Linked Data (QALD) provided 50 training and 50 test questions on DBpedia dataset along with their true answers. We use the 50 QALD training questions to tune our system, setting the thresholds and coefficients. Of the 50 test questions, 33 questions can be answered using only the DBpedia Ontology, while the rest need knowledge from the YAGO ontology. We used the 33 questions to evaluate our system. We modified seven of them that required Boolean answers or operations not supported by our semantic graph notation for the time being, such as grouping and counts. Our changes included changing the answer type or removing the unsupported operations but preserving the relations and thus the question schemata. Our collection of 33 test questions with their true answers are available at <http://ebiq.org/r/326>.

Three human subjects unfamiliar with the DBpedia ontology independently translated the test questions into semantic graph queries. Three versions of 33 semantic graphs were given to our system which automatically translated them into SPARQL queries. The average time to translate a semantic graph to its corresponding SPARQL queries was 0.38 second. The queries were then run on public DBpedia SPARQL endpoints to produce answers. The precision, recall and f-measure of our system, averaging on three versions, are 0.687, 0.722 and 0.704, respectively.

5 Conclusion

GoRelations is an intuitive query system that allows people to query DBpedia without mastering SPARQL or acquiring detailed knowledge of the classes and properties used in the underlying ontologies. It's interface uses a simple *semantic graph* notation for queries that is automatically translated into a corresponding SPARQL query. We developed a novel three-step mapping approach that disambiguates user terms in a semantic graph query and maps them to DBpedia ontology terms. Our system was evaluated on 33 QALD test questions and the result shows the approach works decently well.

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. Lopez, V., Fernández, M., Motta, E., Stieler, N.: Poweraqua: Supporting users in querying and exploring the semantic web content. *Semantic Web Journal* (2011)
3. Church, K., Hanks, P.: Word association norms, mutual information and lexicography. In: Proc. 27th Annual Conf. of the ACL, pp. 76–83 (1989)
4. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: Proc. 21st AAAI Conf., pp. 775–780 (2006)
5. Rapp, R.: Word sense discovery based on sense descriptor dissimilarity. In: Proc. 9th Machine Translation Summit, pp. 315–322 (2003)

Proposed SKOS Extensions for BioPortal Terminology Services

Cui Tao¹, Natalya F. Noy², Harold R. Solbrig¹,
Nigam H. Shah², Mark A. Musen², and Christopher G. Chute¹

¹ Division of Biomedical Statistics and Informatics,
Mayo Clinic College of Medicine, Rochester, MN

² Stanford Center for Biomedical Informatics Research,
Stanford University, Stanford, CA

Abstract. The National Center for Biomedical Ontology (NCBO) BioPortal provides common access for browsing and querying a large set of ontologies that are commonly used in biomedical communities. One of our missions is to align lexical features (i.e., textual definitions) that are commonly used in these ontologies across different representation formats with standard tags and to represent them in a standard way to the users. The Simple Knowledge Organization System (SKOS) is a recommendation of the World-Wide-Web Consortium (W3C) for a common data model for sharing and linking knowledge organization systems on the Semantic Web. The BioPortal is in the process of adopting SKOS in the backend representation for its content. During this process, we discovered that there exists a set of commonly-used lexical features shared by the biomedical ontologies that SKOS does not yet represent. In this paper, we discuss our proposed SKOS extensions to cover this set of commonly used lexical features, the rationales, and the detailed description of each proposed construct.

1 Introduction

The use of ontologies in the biomedical domain has accelerated dramatically over the last decade [5,6,7]. Biomedical ontologies provide the essential domain knowledge needed for different purposes such as data integration, data sharing, semantic annotation, information extraction, and natural language processing. The National Center for Biomedical Ontology (NCBO) BioPortal [4] is an open-source repository of ontologies, terminologies, and thesauri of importance in biomedicine. As of Oct. 2011, BioPortal is hosting 296 biomedical ontologies, including more than 5 million terms. These ontologies cover domain knowledge from basic science, clinical terms, to translational studies.

One important goal of BioPortal is to provide a common access for browsing and querying the shared ontologies that are commonly used in the biomedical communities [10]. BioPortal is designed for harmonizing these resources in a uniform representation and delivering this content in a consistent, standardized fashion for use in biomedical, clinical, and research applications.

This goal coincides with one of the missions of the Semantic Web community: creating technologies to share heterogeneous content and to distribute it over the Web. The World Wide Web Consortium (W3C) has produced recommendations for a stack of Semantic Web languages for structured data, terminologies, and ontologies, including the Resource Description Framework (RDF) for representing data on the Web [13], the SPARQL language for querying RDF data [16], the Web Ontology Language (OWL2) for representing ontologies on the Web [12], the Simple Knowledge Organization System (SKOS) [14], and SKOS eXtension for Labels (SKOS-XL) [15] for representing terminologies and thesauri.

One necessary step toward this goal is to represent commonly existing properties in the biomedical domain in a standard, Semantic-Web compliant way. BioPortal is a host for different kinds of resources: formal ontologies, terminologies, thesauri, and so on. Formal ontologies are mapped to a formal semantics, thus allowing well defined and rather powerful inferences, and use a precise and rigorous set of constructs to represent their content. Thesauri and other knowledge-organization systems, however, have less emphasis on formal structure and focus more on lexical components such as synonyms, multilingual definitions, examples, comments, cross references. The latest SKOS specification describes a standard set of tags for most of these constructs and has been adopted by many organizations, including the Library of Congress [8], NASA [9], and the United National Food and Agriculture Organization [3].

BioPortal assists authors in the identification of the related constructs during terminology submission. For example, when submitting their terminology or ontology, the authors indicate which property in their ontology stores synonyms, authors of each term, definitions of a term and so on. We then relate the corresponding lexical components to the SKOS standard, which, in combination with a structured terminology model, can serve as a baseline for the representation of the next generation of terminological resources. During this process, we have identified several properties and constructs that are missing from SKOS, but are critical to biomedical terminologies, including:

- **multiple definitions** - resources may be accompanied by multiple (textual definitions), derived from multiple sources. The resource authors need to indicate which of these definitions is preferred in a given language or context.
- **flavors of annotation** - resource annotations include information about what was changed and when, the current status of the entry and targeted directions, when and where the resource is applicable, etc. Applications need to be able to differentiate these various flavors of notes as some are applicable to end user situations while others are only of value in editorial or historical contexts. In addition, applications need to be able to clearly differentiate definitions from comments from examples, as when and where each of these is used is different.
- **lexical semantics** - as ontologies begin to be consumed in the NLP space, it becomes important to be able to identify the various forms of labels - the noun form, adjectival form, singular, plural as well as the label derivation - acronym, abbreviation, eponym, etc. In addition, the introduction of

multi-lingual lexical systems creates a need to be able to identify the derivation of specific labels, notes, etc. The fact that a German definition is a literal translation of a corresponding English definition is important when attempting to understand the intended meaning of a term. Similarly, an acronym or abbreviation needs to be associated with the language and term that it is an acronym and abbreviation for.

Building on our experience with LexGrid [1], which provides common terminology services for biomedical terminologies, as well as standard ontology representation guidelines [17], we propose a set of constructs to cover these common lexical properties in the biomedical domain.

2 SKOS Extensions

Table 1. SKOS Extension Summary

New Construct	Description	Comment
skosxl_plus:Comment	Notes for a resource excluding examples and definitions	see Figure 1
skosxl_plus:Note	Similar to skosxl:Label but works with skos:noteRelation	
skosxl_plus:Source	Reified provenance details.	
skos_plus:prefDefinition	The preferred definition of a resource given a specific context or language	subproperty of <i>skos:definition</i>
skos_plus:altDefinition	Alternative definitions of a resource	subproperty of <i>skos:definition</i>
skos_plus:designationType	Discriminant that determines the type of a particular description	synonyms, acronyms, short names, etc
skosxl_plus:noteRelation	Relationships between two lexical properties	a super property of <i>skosxl:labelRelation</i> , domain and range are skosxl:Note
skosxl:literalForm		Expand its domain and range to cover skosxl:Note

Comments and Notes. “Notes are used to provide information relating to SKOS concepts.” [14]. The current SKOS specification identifies 6 subclasses, *skos:changeNote*, *skos:definition*, *skos:editorialNote*, *skos:example*, *skos:historyNote*, *skos:scopeNote*. Applications that consume terminology services need to be able to distinguish *definitions*, *examples* from other forms of notes as definitions are used to clarify the meaning of a concept or term, examples help to refine it, while other types of comments provide a variety of secondary

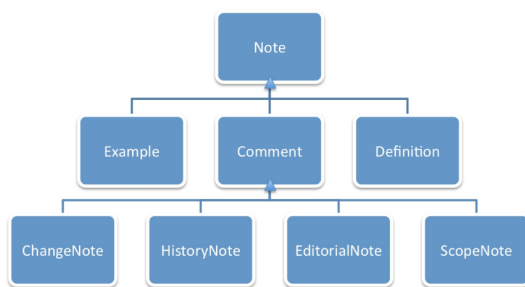


Fig. 1. Proposed Hierarchy of `skos:note`

purposes. We propose the addition of one additional class, *skos_plus:Comment*, which provides this primary distinction.¹

Preferred Definition. Textual definitions are becoming increasingly common in biomedical ontologies. They provide an important link between the computational formalism of the ontology and the (intended) meaning of the term, class or property. A given class may be accompanied by several definitions that are drawn from multiple reference sources. Definitions may be provided in different languages and may be applicable in different contexts. SKOS currently defines *prefLabel* and *altLabel*, but no analogous distinctions are provided for definitions. To remedy this, we propose the creation of *prefDefinition* and *altDefinition*, as subproperties of *skos:definition*, with the assertion that each concept may have at most one preferred definition given a language.

Type of Descriptions. The use and a meaning of a term label can require additional lexical semantics. It is often important to know whether a label is a noun form, adjectival form, whether it is singular or plural, whether it is a common acronym, an abbreviation or a full form. The existing SKOS specification provides some ability to make these distinctions, as labels can be marked as “preferred”, “alternate” or “hidden”, which indicates whether they are the primary identifiers, synonyms or deprecated forms such as common misspellings, deprecated acronyms, etc. It doesn’t, however, provide the sort of granularity many sophisticated applications need. To address this issue, we propose a new construct, *designationType*, to define the type of a description of a given resource. Typical types would include noun, adjective, acronym, eponym, and abbreviation, which we propose can be drawn from the ISO TC37 Data Category Registry².

¹ The reason that the existing tag, *rdfs:comment* was not used is that it is often heavily overloaded in existing ontologies— being used to carry definitions, examples, various flavors of comments, etc.

```

<C1> skosxl:altLabel <L>.
<L> rdf:type skosxl:Label;
    skosxl:literalForm "FAO";
    skos_plus:designationType ISOcat:acronym.

ISOcat:acronym rdfs:subClassOf ISOcat:abbreviatedForm.

```

Fig. 2. An Example of Designation Type

Figure 2 shows an example using *designationType*. Here we use the SKOS-XL data model to represent that concept <C1> has an alternative label “FAO” which serves the role of *acronym*.

Relations between Lexical Properties. SKOS-XL defines *skosxl:labelRelation* to represent binary relations between instances of the class *skosxl:Label*, which allows assertions such as the label, “FAO” is an *ISOcat:acronymFor* “Food and Agriculture Organization”. This pattern, however, does not extend to *definition-definition* or *comment-definition* relationships, etc. To address this issue, we propose a new construct, *noteRelation*, which can be used for representing relations between not only two labels, but also any two lexical properties, such as definitions, comments, and examples.

We also propose a new class—*skosxl_plus:Note*—as the domain and range of *noteRelation*. Similar to *designationType*, the types of these relations could be adopted from the ISO TC37 Data Category Registry. Figure 3 provides an example of a *noteRelation*.

```

<C1> skos:editorialNote <E1>.
<E1> rdf:type skosxl_plus:Note;
    skosxl:literalForm "needs to be updated later"@en.
<C1> skos:editorialNote <E2>.
<E2> rdf:type skosxl_plus:Note;
    skosxl:literalForm "moeten later worden bijgewerkt"@de.
<E2> ISOcat:translation <E1>.

ISOcat:translation rdfs:subPropertyOf skosxl_plus:noteRelation.

```

Fig. 3. An Example of Property Relation

² ISOcat does not follow the convention of capitalizing the first letter of class names - all identifiers begin with a lower case letter. We have included a type definition in the examples to help clarify this.

³ Note that ISO TC37 does not, at the moment, appear to provide a property “translation of”. We will need to address the difference between the verb / noun (property / class) with this organization.

SKOS-XL has defined *skosxl:prefLabel*, *skosxl:altLabel*, and *skosxl:hiddenLabel* as instances of the class *skosxl:Label*. These new constructs are necessary because *skos:prefLabel*, *skos:altLabel* and *skos:hiddenLabel* are sub-properties of *rdfs:label*, and therefore inherit its range of *rdfs:Literal* vs. the class *skosxl:label*. *skos:note* and its sub properties are already defined as instances of *owl:AnnotationProperty*, which means that the target does not have the same restrictions and we can use them directly when connecting a concept (e.g., $\langle C1 \rangle$ to note (e.g., $\langle E1 \rangle$ or $\langle E2 \rangle$ in Figure 3).

Provenance Annotations. We also need the ability to provide provenance information for the lexical resources in an ontology.⁴ As an example, the OBO [11] allows the specification of the source roles and documents of definitions. Figure 4 shows the definition of “reproduction” in the Gene Ontology.

```
def: "The production by an organism of new individuals that contain some
portion of their genetic material inherited from that organism."
[GOC:go_curators, ISBN:0198506732 "Oxford Dictionary of
Biochemistry and Molecular Biology"]

G0:0000003 rdf:type skos:Concept;
           skos_plus:prefDefinition <D1>.
<D1> rdf:type skosxl_plus:prefDefinition;
     skosxl:literalForm "The production by an organism of new ...";
     dc:source <S1>.
<S1> rdf:type skosxl_plus:Source;
     crdf:role GOC:go_curators;
     crdf:sourceDocument URN:ISBN:0198506732;
     crdf:sourceDescription "Oxford Dictionary of Biochemistry
and Molecular Biology".
```

Fig. 4. Definition of “reproduction” in OBO and RDF

In this case, we need a general class for source to allow further annotations of the instances of the source. Here we propose a new class, *skosxl_plus:Source*, to represent the overall source information for reification purposes. Note that we use the *crdf* namespace for tags that describe further source information. We do not propose that these tags are included in the SKOS extension because we believe that they are out of scope for SKOS. Note that the “ $\langle S1 \rangle$ ” tag is used in this example because this represents a description of URN:ISBN:019506732 *by* the GO curators rather than the resource itself.

3 Discussion

The definition of a logic formalism consists of two sections. The first section defines *syntax* of the logic—describing the set of symbols that are valid in the logic

⁴ Fine grained provenance is also required in the *formal* or “semantic” assertions, but that is out of the scope of this document.

and the set of possible transformations that may be applied to them. The second section describes the *semantics*—a set of rules that describe how the various logic symbols and operations correspond with equivalent structures the (or some) “real world”. Most of the Ontology efforts to date have been focused on the first section—the representation and manipulation of the formal symbols. The *semantics* of an ontology is represented by the *lexical* aspects, and is the part that allows human beings to map the symbols from and to their intended referents. An ontology that fails to maintain this second, lexical component becomes nothing more than an interesting mathematical artefact—a set of logically consistent symbols that may or may not apply to the real world.

The SKOS [14] specification has formed an excellent starting point for formalizing the lexical aspects of an ontology. Today, many of the ontologies that pay attention to the lexical components at all use ontology specific tags for these components, or, in some situations, embed the meanings of these tags lexically inside *rdfs:comment* constructs. The conversion of these idiosyncratic approaches into one that is semantically consistent requires a sufficiently robust target that the majority of the information that needs conversion can be migrated without significant loss.

The proposals that we discuss in this document suggest one such approach. It should be noted, however, that this approach depends on either (a) semantics that do not formally include the distinction made by OWL between *AnnotationProperty* and *ObjectProperty* and *DataProperty* or (b) the use of OWL 2. The reason for this is because the OWL 1 series of specifications restrict *AnnotationProperties* to the point that they cannot be used as first class resources without lapsing into the computational marshland of OWL Full. This restriction has driven many existing ontologies to resort to a variety of tricks and embedding schemes to allow them to carry the information they need while, at the same time, taking advantage of the formal classifications that are available.

The proposals presented here seem to work in principal. To be truly useful, however, more standardization work still remains. While the ISOCat Data Category Registry [2] appears to carry many of the markup properties that are needed, there are still issues about format, rights and completeness that need to be resolved. The specific provenance tags such *sourceDocument*, *sourceDescription* need standardization as well before the work done here will begin to bear significant fruit.

4 Concluding Remarks

In this paper, we discussed our proposed SKOS extension for representing common lexical information in BioPortal ontologies. One important goal of BioPortal is to provide a common access for browsing and querying the shared ontologies that are commonly used in the biomedical communities. We believe these additional properties will improve interoperability among biomedical ontologies, and therefore enhance the common service provided by BioPortal.

These proposed extensions were based on our more than a decade experience with common terminology service in the biomedical domain. We believe that SKOS has already provided a reasonably good foundation for common lexical information representation. The proposed tags will be a useful addition to SKOS for harmonizing biomedical ontologies.

Acknowledgement. This research is partially supported by the National Center for Biomedical Ontology (NCBO) under the NIH Grant #N01-HG04028 and the NSF under Grant #0937060 to the Computing Research Association for the CIFellows Project.

References

1. LexGrid: The Lexical Grid (2009), <https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/LexGrid>
2. ISOcat - data category registry (October 2011), <http://www.isocat.org>
3. AGROVOC Thesaurus, Food and Agriculture Organization of the United Nations (FAO), <http://www.fao.org/agrovoc>
4. NCBO Biportal, <http://biportal.bioontology.org/>
5. Bodenreider, O.: Biomedical Ontologies in Action: Role in Knowledge Management, Data Integration and Decision Support. In: Geissbuhler, A., Kulikowski, C. (eds.) IMIA Yearbook of Medical Informatics, vol. 47, pp. 67–79. International Medical Informatics Association (2008)
6. Chute, C.G.: The Copernican Era of Healthcare Terminology: A Re-Centering of Health Information Systems. In: AMIA Annual Symposium, pp. 68–73 (1998)
7. Chute, C.G.: Clinical Classification and Terminology: Some History and Current Observations. *JAMIA* 7(3), 298–303 (2000)
8. Library of congress subject headings, the library of congress cataloging distribution service, <http://www.loc.gov/cds/lcsh.html>
9. NASA Taxonomy, <http://nasataxonomy.jpl.nasa.gov/fordevelopers/>
10. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.D., Chute, C.G., Musen, M.A.: Biportal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research* 37, 170–173 (2009)
11. The open biomedical ontologies, <http://www.obofoundry.org/>
12. OWL 2 web ontology language structural specification and functional-style syntax, <http://www.w3.org/TR/owl2-syntax/>
13. The RDF vocabulary, <http://www.w3.org/1999/02/22-rdf-syntax-ns>
14. SKOS vocabulary, <http://www.w3.org/2006/07/SWD/SKOS/reference/20090315/skos.rdf>
15. SKOS XL vocabulary, <http://www.w3.org/2006/07/SWD/SKOS/reference/20090315/skos-xl.rdf>
16. SPARQL Query Language for RDF, www.w3.org/TR/rdf-sparql-query/
17. Tao, C., Pathak, J., Solbrig, H.R., Wei, W., Chute, C.G.: Common terminology guidelines for representing biomedical ontologies in semantic web notations. *Journal of Biomedical Informatics* (submitted)

Learning Complex Mappings between Ontologies

Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu

State Key Laboratory for Novel Software Technology,
Nanjing University, China

{whu,yzqu}@nju.edu.cn, jf_chen@ymail.com,
hangzhang@smail.nju.edu.cn

Abstract. In this paper, we introduce a new approach for constructing complex mappings between ontologies by transforming it to a rule learning process. Derived from the classical Inductive Logic Programming, our approach uses instance mappings as training data and employs tailoring heuristics to improve the learning efficiency. Empirical evaluation shows that our generated Horn-rule mappings are meaningful.

1 Introduction

As the amount of ontologies grows with the development of the Semantic Web, it is now common to have different ontologies in a single domain. Ontologies can be *heterogeneous* in various forms including terminological and conceptual heterogeneities. Such heterogeneities are dealt with the *ontology matching* process, which plays a vital role in semantic interoperability between applications.

There exist plenty of works that tackle the ontology matching problem [2], however they mainly focus on finding simple 1 : 1 equivalence mappings between named classes and/or named properties in ontologies. In the real world, *complex m : n mappings* are also pervasive and needed by many applications for query rewriting, distributed reasoning, instance migration, etc. Neither the discovery methods to complex mappings nor their formal semantics has been well studied in literature yet.

In this paper, we propose a new approach which transforms the problem of constructing complex mappings to rule learning across ontologies. Our approach extends a pioneering *Inductive Logic Programming* (ILP) method named FOIL [8], and generates complex mappings in the form of first-order *Horn-rules*. Furthermore, our approach exploits instance mappings as the training data for variable bindings, and heuristically reduces the search and storage space by tailoring irrelevant data. Empirical evaluation shows the feasibility of our approach.

The rest of this paper is organized as follows. Preliminaries are introduced in Section 2. Section 3 presents our learning approach while Section 4 reports experimental results. Finally, we conclude this paper in Section 5.

2 Problem Statement

The Web Ontology Language (OWL), as a W3C recommendation, is proposed for authoring ontologies on the Web [5]. OWL DL is a species of OWL, whose

logic foundation is Description Logic (DL), a subset of First-Order Logic (FOL). An *OWL DL ontology* declares axioms and facts for its classes, properties and instances, where a *class* corresponds to a unary predicate in FOL with one free variable, a *property* to a binary predicate with two free variables, and an *instance* to a constant.

Regarding Horn-rules, a *clause* is a disjunction of literals, where a *literal* is a (class or property) predicate that is applied on constants or variables. A *positive* literal is a literal that can be satisfied; otherwise it is *negative*. A clause is called a *Horn-clause* if it has at most one positive literal. A *Horn-rule* is a category of Horn-clause that has one positive literal and at least one negative literal, which can be written as: $H \leftarrow L_1 \wedge L_2 \wedge \dots \wedge L_n$, where H is called the *head* of the rule, and $L_1 \wedge L_2 \wedge \dots \wedge L_n$ is called the *body*. \leftarrow gives an *implication* from the body to the head. A *variable binding* is a substitution which maps a variable to a constant in a Horn-rule.

We define the construction of complex mappings as a rule learning process across two ontologies. The mappings between classes or properties in different ontologies are expressed in the form of Horn-rules to reflect their (directional) semantics.

Definition 1. Let $\mathcal{O}_1, \mathcal{O}_2$ be the source and target ontologies, respectively. Finding complex mappings from \mathcal{O}_1 to \mathcal{O}_2 is to learn a set of Horn-rules: $\mathcal{M}_{\mathcal{O}_1 \rightsquigarrow \mathcal{O}_2} = \{M_1, M_2, \dots, M_k\}$. A complex mapping M_i ($i = 1, 2, \dots, k$) is as follows:

$$\mathcal{O}_2 : H \leftarrow \mathcal{O}_1 : L_1 \wedge \mathcal{O}_1 : L_2 \wedge \dots \wedge \mathcal{O}_1 : L_n,$$

where H, L_j ($j = 1, 2, \dots, n$) are literals in $\mathcal{O}_2, \mathcal{O}_1$, respectively. A complex mapping satisfies that its rule body has at least two literals ($n \geq 2$).

In practice, we often match \mathcal{O}_1 to \mathcal{O}_2 , and switch them to match \mathcal{O}_2 to \mathcal{O}_1 . It is worth noting that the above Horn-rule mappings can only cover a kind of “sequential” complex mappings, rather than some “combinational” mappings [3] or formula-like ones [1]. The semantics of Horn-rule mappings may be interpreted using the binding semantics [11], where each ontology has a “subjective semantics” based on local interpretation and a “foreign semantics” based on semantic binding to matched ontologies. In general, reasoning with OWL and Horn-rules is an undecidable problem [6].

A Motivating Example. Fig. 1 shows the source and target ontologies $\mathcal{O}_1, \mathcal{O}_2$. \mathcal{O}_1 has two instances `Jianfeng.Chen` and `Yuzhong.Qu` of the class `foaf:Person`, and an instance `Semantic.Web` of `Course`. There are two properties `takeCourse` and `teachBy` that link these instances. \mathcal{O}_2 defines in a similar way. Let us assume that $\mathcal{O}_1 : \text{Jianfeng.Chen}$ and $\mathcal{O}_2 : \text{jfchen}$ denote the same person (which can be identified by some instance matching algorithms), and so do $\mathcal{O}_1 : \text{Yuzhong.Qu}$ and $\mathcal{O}_2 : \text{yzqu}$, we can learn complex mappings as follows:

$$\begin{aligned} \mathcal{O}_2 : \text{hasTeacher}(x, y) &\leftarrow \mathcal{O}_1 : \text{takeCourse}(x, z) \wedge \mathcal{O}_1 : \text{teachBy}(z, y), \\ \mathcal{O}_2 : \text{Student}(x) &\leftarrow \mathcal{O}_1 : \text{Person}(x) \wedge \mathcal{O}_1 : \text{takeCourse}(x, z). \quad \square \end{aligned}$$

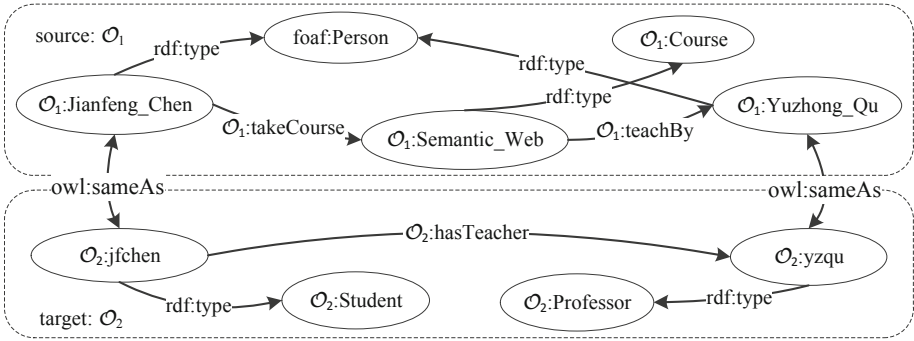


Fig. 1. A motivating example

Related Work. The state of the art ontology matching techniques are limited to detect simple mappings between atomic classes and properties, while finding complex mappings is relatively new. Ritze et al. [9] defined various hand-tailored patterns and used heuristic search methods, and Dhamankar et al. [11] developed a similar system called iMAP. He et al. [3] proposed to apply correlation mining to find co-occurrence entities. Qin et al. [7] gave a systematic approach to create executable mapping rules in the Web-PDDL language. Stuckenschmidt et al. [10] recently summarized the challenges of applying ILP to learn complex mappings, which inspires us to present a concrete approach in this paper.

3 Approach

The outline of our algorithm for learning complex mappings is shown in Algorithm 1, which starts with two ontologies with their instances as input, and its goal is to create as output a set of complex mappings in the form of Horn-rules. In general, the algorithm contains three phases:

- *Instance matching phase* matches instances, which are then used as correspondences between ontologies for finding and validating complex mappings.
- *Data tailoring phase* conducts a class-based extraction mechanism to eliminate irrelevant data from the source ontology.
- *Mapping learning phase* performs a general-to-specific search to construct Horn-rule mappings for classes and properties.

We will describe the details of these phases in the rest of this section.

3.1 Instance Matching Phase

We use a set of OWL terms, namely owl:sameAs , $\text{owl:FunctionalProperty}$, $\text{owl:InverseFunctionalProperty}$, owl:cardinality and $\text{owl:maxCardinality}$, to infer instance mappings (see Line 2 in Algorithm 1). For example, foaf:mbox is

Algorithm 1: ComplexMatch**Input:** Source and target ontologies $\mathcal{O}_s, \mathcal{O}_t$, including their instances $\mathcal{I}_s, \mathcal{I}_t$.**Output:** A set of complex mappings \mathcal{M} .

```

1 begin
2    $\mathcal{A} = \text{MatchInstances}(\mathcal{I}_s, \mathcal{I}_t);$  // Instance matching phase
3    $\mathcal{C}_t = \text{SelectClassesFromTargetOntology}(\mathcal{O}_t, \mathcal{A});$ 
4   foreach  $C \in \mathcal{C}_t$  do // Data tailoring phase
5      $(\mathcal{P}_t, \mathcal{I}_t^C) = \text{SelectPredicatesAndInstances}(C, \mathcal{O}_t, \mathcal{I}_t, \delta);$  //  $|\mathcal{I}_t^C| \leq \delta$ 
6      $(\mathcal{P}_s, \mathcal{F}_s) = \text{SelectPredicatesAndFactsFromSourceOntology}(\mathcal{O}_s, \mathcal{I}_s, \mathcal{I}_t^C, \mathcal{A});$ 
7     foreach  $P \in \mathcal{P}_t$  do // Mapping learning phase
8        $M = \text{ConstructMapping}(P, \mathcal{P}_s, \mathcal{F}_s, \mathcal{A});$ 
9       Add  $M$  to  $\mathcal{M}$ ;
10  return  $\mathcal{M}$ ;
11 end

```

an inverse functional property. If \mathcal{O}_1 : `Jianfeng_Chen` and \mathcal{O}_2 : `jfchen` have the same email address, they constitute an instance mapping. These five built-in terms in OWL are frequently used in data fusion systems [4], and we integrate them as a transitive closure to form a set of semantically equivalent mappings.

We employed seven students to evaluate 500 instance mappings in [4], and observed that the average precision of our approach is about 0.95. Additionally, this matching phase can be completed in a few microseconds. Please note that our approach would lost some mappings, which leads to a low recall. However, we argue that the ILP-based learning relies more on the quality of training data than its quantity.

3.2 Data Tailoring Phase

We extract relevant predicates and facts from the source ontology by leveraging the instance mappings (see Line 6 in Algorithm 1). As a result, we reduce both the searching space for candidate predicates and the storage space for negative variable bindings. We propose different strategies for classes and properties to extract the corresponding training data:

- Classes in an ontology represent the conceptual classification. We guide the data tailoring by typing information, e.g., `rdf:type` and `rdfs:subClassOf`. Specifically, we start from each instance in the source ontology and prefer to collect the facts about those instances having the same type.
- Properties express the relationships between instances. We collect the facts for the instances that can constitute property chains, i.e., a set of properties that are chained with compatible `rdfs:domain(s)` and `rdfs:range(s)`.
- We also set a boundary for the relevant data search, which is terminated once five facts have been crossed at a search direction. Additionally, for each class, we randomly select at most δ instances (denoted by \mathcal{I}_t^C) and associated predicates (\mathcal{P}_t) from the target ontology to avoid the combinatorial explosion (see Line 5 in Algorithm 1).

Example. Supposing that the data tailoring phase utilizes $\mathcal{O}_2 : \text{jfchen}$ as the start and obtains the matched instance $\mathcal{O}_1 : \text{Jianfeng_Chen}$ from \mathcal{O}_1 . Then, we collect the facts local to $\mathcal{O}_1 : \text{Jianfeng_Chen}$. For the class `Student`, the phase prefers to the facts about the instances that have the same class `foaf:Person` as $\mathcal{O}_1 : \text{Jianfeng_Chen}$ does, e.g., $\mathcal{O}_1 : \text{Yuzhong_Qu}$, whose facts can provide the evidence for negative bindings. For the property $\mathcal{O}_2 : \text{hasTeacher}$, it prefers to the fact `takeCourse(Jianfeng_Chen, Semantic_Web)`. \square

3.3 Mapping Learning Phase

In Algorithm 2, we learn a complex mapping in the form of Horn-rules (called at Line 8 in Algorithm 1). The algorithm conducts a greedy search to select the best literals, where the goodness of a candidate literal is measured by `Gain()` in Line 7. This learning process terminates when the number of positive variable bindings versus negative ones is insignificant or the length of rule body exceeds a threshold (e.g., $\epsilon = 10, \theta = 7$ in our case).

Algorithm 2: ConstructMapping

Input: A target predicate P , a set of source predicates \mathcal{P}_s , a set of source facts \mathcal{F}_s and a set of instance mappings \mathcal{A} .

Output: A complex mapping M .

```

1 begin
2   Initialize a Horn-rule  $M: P \leftarrow$ ;
3    $PVB(M) = |\text{PositiveVariableBindings}(M, \mathcal{P}_s, \mathcal{F}_s, \mathcal{A})|$ ;
4    $NVB(M) = |\text{NegativeVariableBindings}(M, \mathcal{P}_s, \mathcal{F}_s, \mathcal{A})|$ ;
5   while  $\frac{PVB(M)}{NVB(M)} > \epsilon$  &&  $\text{BodyLength}(M) < \theta$  do
6      $\mathcal{L} = \text{GenerateAllCandidateLiterals}(M, \mathcal{P}_s)$ ;
7      $L_{\max} = \arg \max_{L \in \mathcal{L}} \text{Gain}(L, M)$ ;
8     Append  $L_{\max}$  to the body of  $M$ ;
9     Update  $PVB$  and  $NVB$  w.r.t. the new  $M_{L_{\max}}$ ;
10  return  $M$ ;
11 end
```

Whether appending a new literal to the current mapping is evaluated based upon the numbers of positive and negative variable bindings, with an objective to cover more positives and fewer negatives. The two sets of bindings keep updating during the learning. More specifically, our algorithm constructs positive variable bindings in terms of the instance mappings, whereas inferring negative variable bindings in terms of the constants extracted from ontology facts. For each matched instance within the source ontology, we create the negative variable bindings that only refer to the constants from its close facts. For example, in Fig. 1 $\{x/\text{Jianfeng_Chen}, y/\text{Yuzhong_Qu}\}$ is a positive variable binding for $\mathcal{O}_2 : \text{hasTeacher}(x, y)$, while $\{x/\text{Jianfeng_Chen}, y/\text{Semantic_Web}\}$ and $\{x/\text{Yuzhong_Qu}, y/\text{Jianfeng_Chen}\}$ are negative ones, due to no corresponding fact can be found in the source ontology. It worth noting that all constants in

the negative variable bindings are local to `Jianfeng_Chen`, departing from those of other instances, such as the constants from the facts close to `Hang_Zhang`.

We use the FOIL gain [8] to measure whether a candidate literal should be appended, which is computed as follows:

$$\text{Gain}(L, M) = t \times \left(\log \frac{PVB(M_L)}{PVB(M_L) + NVB(M_L)} - \log \frac{PVB(M)}{PVB(M) + NVB(M)} \right),$$

where $PVB(M)$, $NVB(M)$ are the numbers of positive and negative variable bindings of the complex mapping M , respectively. t is the number of positives of M that are still covered after appending a new literal L to M (denoted by M_L).

Example. Let $\mathcal{O}_2 : \text{hasTeacher}(x, y) \leftarrow$ be the initial complex mapping with an empty body. Given the three possible constants in \mathcal{O}_1 , there are nine possible variable bindings for the initial mapping, where only one is positive. PVB is 1 and NVB is 8. Consider the candidate literals $\mathcal{O}_1 : \text{foaf:Person}(x)$ and $\mathcal{O}_1 : \text{takeCourse}(x, z)$. To add `foaf:Person`, PVB remains 1 but NVB reduces to 5, so its Gain equals 0.85. Similarly, the Gain of `takeCourse` is 1.58. The new complex mapping (in progress) is: $\mathcal{O}_2 : \text{hasTeacher}(x, y) \leftarrow \mathcal{O}_1 : \text{takeCourse}(x, z)$. \square

4 Evaluation

We developed the proposed approach in Java and conducted empirical evaluation on its performance. Our three test cases are constituted by ontologies in various domains, namely Restaurants from IM@OAEI2010, SwetoDBLP vs. ESWC, and Mondial vs. Factbook, each of which contains tens of classes or properties and thousands of instances.

Table 1. Statistics of three datasets

\mathcal{O}_1	#Classes	#Props.	#Insts.	\mathcal{O}_2	#Classes	#Props.	#Insts.
Restaurant1	3	7	339	Restaurant2	3	7	2,256
SwetoDBLP	17	46	83	ESWC	35	31	1,261
Mondial	17	40	15,398	Factbook	36	171	24,990

We found six complex mappings from the three cases excluding simple 1 : 1 ones, where the average body length is 2.17. According to human observation, all the mappings are correct, so the precision equals to 1.0. It also shows that the number of complex mappings are relatively few, because the schemas of the ontologies in the experiment are not complicated. However, we believe that, with the wider acceptance of OWL 2 [5], especially the property chain feature, there will be more complex mappings that can be learnt from our ILP-based method. So, an interesting future work is to see what kinds of complex mappings can be obtained in more realistic cases.

We also counted the learning time w.r.t. the number of instance mappings, and found that our approach accelerated twice more than the original approach that does not tailor irrelevant data. The optimized approach spent nearly two minutes to complete the learning on 100 instance mappings.

We illustrate two complex property mappings for Restaurants and Mondial vs. Factbook respectively, which give some clear meanings and are difficult to be logically entailed by the union of simple mappings and the input ontologies:

$$\begin{aligned} \mathcal{O}_{\text{Rest1}} : \text{city}(x, y) &\leftarrow \mathcal{O}_{\text{Rest2}} : \text{is_in_city}(x, z) \wedge \mathcal{O}_{\text{Rest2}} : \text{name}(z, y), \\ \mathcal{O}_{\text{Mondial}} : \text{neighbor}(x, y) &\leftarrow \mathcal{O}_{\text{Factbook}} : \text{border}(x, z) \wedge \mathcal{O}_{\text{Factbook}} : \text{country}(z, y). \end{aligned}$$

Additionally, a complex class mapping for SwetoDBLP vs. ESWC is as follows:

$$\begin{aligned} \mathcal{O}_{\text{ESWC}} : \text{InProceedings}(x) &\leftarrow \mathcal{O}_{\text{DBLP}} : \text{Publication}(x) \\ &\wedge \mathcal{O}_{\text{DBLP}} : \text{isIncludeIn}(x, y) \wedge \mathcal{O}_{\text{DBLP}} : \text{Proceedings}(y). \end{aligned}$$

These complex mappings can be easily translated to SPARQL queries, such as using the Named Graph to query the two ontologies and fuse the query results together in applications.

5 Conclusion

In this paper, we proposed a new approach for learning complex mappings between ontologies. We utilized instance mappings as training data and applied data tailoring to improve efficiency. We implemented the approach and experimentally evaluated it on three pairs of real-world ontologies, which showed the high precision and clear meanings of the constructed Horn-rule mappings. The work reported here is a first step, and many issues still need to be addressed in future. For example, we look forward to exploring more possible semantics and rules for learning and testing our approach steadily on new datasets in Linked Data. We also want to formally analyze the learning complexity of our approach, especially for the negative example construction.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61003018 and 60903010, in part by the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20100091120041, and in part by the National Science Foundation of Jiangsu Province under Grant Nos. BK2011189 and BK2009268.

References

1. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: iMAP: Discovering Complex Semantic Matches Between Database Schemas. In: SIGMOD 2004, pp. 383–394 (2004)
2. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)

3. He, B., Chang, K.C.-C.: Automatic Complex Schema Matching Across Web Query Interfaces: A Correlation Mining Approach. *ACM Transactions on Database Systems* 31(1), 346–395 (2006)
4. Hu, W., Chen, J., Qu, Y.: A Self-training Approach for Resolving Object Coreference on the Semantic Web. In: *WWW 2011*, pp. 87–96 (2011)
5. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: *OWL 2 Web Ontology Language Profiles*. W3C Recommendation (2009)
6. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and Logic Programming Live Together Happily Ever After? 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. 501–514. Springer, Heidelberg (2006)
7. Qin, H., Dou, D., LePendu, P.: Discovering Executable Semantic Mappings Between Ontologies. In: Meersman, R., Tari, Z. (eds.) *OTM 2007, Part I*. LNCS, vol. 4803, pp. 832–849. Springer, Heidelberg (2007)
8. Quinlan, J.R.: Learning Logical Definitions from Relations. *Machine Learning* 5(3), 239–266 (1990)
9. Ritze, D., Meilicke, C., Šváb-Zamazal, O., Stuckenschmidt, H.: A Pattern-Based Ontology Matching Approach for Detecting Complex Correspondences. In: *ISWC Workshop on Ontology Matching* (2009)
10. Stuckenschmidt, H., Predoiu, L., Meilicke, C.: Learning Complex Ontology Alignments – A Challenge for ILP Research. In: *ILP 2008* (2008)
11. Zhao, Y., Wang, K., Topor, R., Pan, J.Z., Giunchiglia, F.: Semantic Cooperation and Knowledge Reuse by Using Autonomous Ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 666–679. Springer, Heidelberg (2007)

Discovering and Ranking New Links for Linked Data Supplier

Nansu Zong, Sungkwon Yang, Hyun Namgoong, and Hong-Gee Kim

Biomedical Knowledge Engineering Lab.
School of Dentistry, Seoul National University, Korea
{zongnansu1982, sungkwon.yang, nghyun, hgkim}@snu.ac.kr

Abstract. For new data supplier who wants to join the web of data club, it's difficult to find new links between local repository and data sets in the web of data to make local data well-connected or harmonize with other data. The purpose of this research is not for finding similar entities but discovering new potential link for helping users have more choice for using multiple links instead of only using "owl:sameAs". The approach use information retrieval technique index the data sets and Page Rank and graph theory analyze RDF document to filter links. We implemented our method using Dbpedia data sets and two open ontologies, the results showed our approach can discover new links with highly accuracy.

Keywords: Link Discovering, Linked Data, Entity Ranking.

1 Introduction

Along with the development of the semantic web, linked data as a new data publish format become popular for connecting distributed data sets across the web instead of using traditional ontology which rich in the schema but limit on the data layer. However, for a new data supplier who wants to join the Linked Open Data club, it's difficult to establish new links between local repository and data sets in the web of data for making local data well-connected or harmonize with other data. Not only, because of establishing links to similar entities in the web of data is not easy to filtrate through heterogeneous data resources, but also because multiple meaningful links to close related entities is hard to navigate. Traditional semantic search engine like Swoogle [5], sindice [11] and Falcon [14] in general domain or NCBO bioportal [1] in a specific domain like Bio-Medical focus on locating the similar entities but not the links to close related entities. These methods have an obvious weakness that it can only navigate the user with the data resource or entities most matching the query word. This makes users have limited choice to make new links between local repository and data in the web of data.

The purpose of this research is not for finding similar entities but discovering potential new links in Linked Open Data. Through our method, users could have

¹ <http://bioportal.bioontology.org/>

more choice for using multiple links instead of only using “owl:sameAs”. An easy example may help for understanding our scenario. Assume one small graph to describe the Tim Berners-Lee who can be considered as the father of the internet, for the data supplier, simply using “owl:sameAs” to link other similar entities may not be satisfied as a user may visit more than two pages to find the useful property of Tim Berners-lee which consuming time and energy. So different with others focusing only on similar entity (Tim Berners-Lee) in Dbpedia², we are trying to find important links of similar entities like WWW³, FellowsOfTheRoyalAcademy-OfEngineering⁴, and ComputerPioners⁵. However, if potential links are diversity, we should rank every linking entity in order to show the importance of the links. In the later section, we will discuss how we compute the importance of links for ranking. The contribution of this paper will be:

- Proposed an approach for the local repository to discover new potential meaningful links to diversity entities in the Linked Open Data.
- Based on the IR techniques, we proposed an approach which combined RDF document Ranking and our entity ranking algorithm for discovering new important links in Linked Open Data.
- We proposed an entity ranking algorithm based on Page Rank and graph theory. Trough locating similar entities of query entity, we traced links connecting to similar entities and ranked all links using our entity ranking algorithm.

We implemented the proposed approach and perform an evaluation where we compare proposed algorithm under different index graph patterns. The evaluation showed our approach can efficiently help users discovering multiple potential links in the web of data.

Outline we will introduce the methodology of this method in Sect.2. Later, we will describe the detail of our entity ranking algorithm. Further more, we will introduce evaluation in the Sect.4 . Finally, in Sect.5, we will make the conclusion and discuss the new issue should be studied in future work.

2 Method Overview

In the Linked Open Data, most data resources are stored in various types of data bases by triples or RDF documents with index in the file system. Although supported with the SPARQL end point, most entities still difficult to be located by simple query sentences. For accurate navigate the related entities of query entity, we implemented traditional Information Retrieval techniques and our own algorithm to discover potential possible new links in heterogeneous data resources. For realize our approach, we proposed two steps for processing. First

² http://dbpedia.org/page/Tim_Berners-Lee/

³ <http://dbpedia.org/page/Www/>

⁴ <http://dbpedia.org/class/yago/FellowsOfTheRoyalAcademyOfEngineering/>

⁵ <http://dbpedia.org/class/yago/ComputerPioners/>

part is through querying the web of data index to filter the most relevant document. Second part is locating the most important links by tracing the links from potential candidate documents get from the RDF document ranking.

2.1 RDF Document Ranking

The inverted index has been widely used in the information retrieval domains. The semantic community also gave a lot of attention to searching the RDF document based on the inverted index and other technologies. Generally, IR uses the Vector Space Model (VSM) and Probabilistic model to rank the web pages, In the conventional Vector Space Model, the TF-IDF which is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. Some approach inherited the TF-IDF, like Siren, which using the triple frequency-inverse source frequency (TF-ISF) and index the RDF based on the node indexing scheme[4]. [2] showed the probability of relevance of a document increase as the term frequency of a term increase, and the increase is non-linear. Our approach implemented the Siren TF-ISF, TF-ISF is inheritance from the TF-IDF.

2.2 Link Discovering and Ranking

In the web of data, every entity is connected by the links. If a new entity hopes to link the data in the Linked Open Data, Firstly, the entity should links to similar entities with “owl:sameAs”. In our approach, we assume a link could be the potential link to the new entity if the link is connected to the similar entity of the local entity. So we defined our problem as discovering entities, which connect to similar entities of the local entity.

Definition 1. *An entity could have a potential relationship with the local entity if the entity has a connection with the similar entities of local entity.*

Each document we get from the index with a local entity query, we will first locate the similar entity in different searched documents. Then set the similar entities as anchor node, and collect all links which connect with the similar entities. We sort all possible links through calculating the importance of every link. Pseudocode below shows the processing of discovering the importance links. Classical method for computing the importance of a link could be considering both the innate link weights, like Semantic Associations[1] and entity importance. In this paper, we will only compute the importance of entity as the importance of the link.

In world-wide web search engine, the importance of web pages is determined by the number of pages linking to and the importance of linking pages, famous Page Rank, HITS[9] and SALSA[10] algorithms are used to rank the web pages based on this theory. In Semantic Web domain, many algorithms used links to compute the importance of resources, like OntoRank[6], Ding[12], ObjectRank[2], ReConRank[8], Triple Rank[7] and Reverse PageRank[13]etc. [3]through calculating the Subject weight, object weight and class weight which

Algorithm 1. Processing of Discovering the important links

Input: Query entity, Index index, TOPdoc, TOPlink.**Output:** List list

```

1:  $D_{TOP} \leftarrow$  Get TOPdoc documents using Entity-Query-Index;
2: for all  $d \in D_{TOP}$  do
3:    $L_n \leftarrow$  All links inside the document  $d$ ;
4:    $E \leftarrow$  Similar entity of input entity in  $d$ ;
5:   for all  $l \in L_n$  do
6:     if  $l$  have connection with similar entity  $E$  then
7:       Compute importance of  $l$ ;
8:     end if
9:   end for
10:  Sort  $L_n$  by importance;
11:   $list \leftarrow list \cup L_n$ ;
12: end for
13:  $list \leftarrow$  list  $L_n$  by importance;
14:  $list \leftarrow$  Get TOPlink links from list;

```

all the weights are based on the connection between entities in the non-class resource. In our approach, we consider the importance of entities not only related to the links but also related on the entity frequency and importance inside graph structure. Our approach will calculate the importance of a link based on the link computing, term frequency and centrality (importance inside graph structure), and more detail will be introduced in Sect.4.

3 Ranking Algorithm for Choosing Important Entities

Most of entity ranking approaches only consider the importance of links, and many mutations of page rank have been implemented. In our paper, we both consider importance of entity inside the RDF document and frequency of document, which contains the specific entity besides page rank.

The merge function for the Page Rank score, Centrality and entity document Frequency is shown:

$$score(E_i) = \alpha PR(E_i) + \beta C(E_i) + \gamma Freq(d, e) \quad (1)$$

where Where the α , β , γ are used to control the influence in each sub-functions, and $\alpha + \beta + \gamma = 1$.

We changed Page Rank function to fit the entity ranking and the formulae is shown below:

$$PR(E_i) = \frac{i-d}{N} + d \sum_{E_j \in D(E_i)} \frac{PR(E_j)}{L(E_j)} \quad (2)$$

where $E_1, E_2 \dots E_n$ are the entities under consideration, where $D(E_i)$ is the Set of entities that links to E_1 , $L(E_j)$ is the number of outbound links to the entity E_j , and N is total number of entities.

The centrality shown below:

$$C(E_i) = \varphi C_c(E_i) + (1 - \varphi)C_B(E_i) \quad (3)$$

$$\text{where } C_B(E_i) = \sum_{s \neq E_i \neq t \neq V} \frac{\sigma_{st} E_i}{\sigma_{st}} \quad (4)$$

$$\text{where } C_c(E_i) = \sum_{t \in V \setminus E_i} d_G(E_i, t)/n - 1 \quad (5)$$

The Entity Document Frequency:

$$Freq(d, e) = \left(\frac{Num_{d \in D}(d, e)}{N} \right)^2 \quad (6)$$

$Freq(d, e)$ means the document frequency which the entity appeared.

The final entity importance function will merge the SIREn query score and entities Ranking Score, the merge function is:

$$Score(e, q) = \rho score(E_i) + (1 - \rho) score(d, q) \quad (7)$$

Where $score(d, q)$ is the document ranking core for the query in the RDF Document Ranking.

4 Preliminary Experiment

For evaluating our approach, we indexed DBpedia pagelinks⁶ as the linked data resource and 60 important entities as query, which randomly selected from Bible Ontology⁷ and Disease Ontology⁸ in linked Life Data. During the index and page rank computing part, we were using the hadoop⁹ cluster with one master and two slaves based on 12 core computing unit with 300 GB hard disk and 3 GB RAM in Cloud environment. In the local searching part, we used the Intel, I-5 CPU with 4 GB RAM and 1TB hard disk on the window XP system. Our assessment of this part can be found in our website¹⁰. Using Siren API, we indexed the DBpedia page link with star-shaped context and muti-shaped context. Star-shaped means we only considered every document with one subject and multiple objects. Muti-shaped index means we put specific entity into one document whatever it appeared in the subject position or object position. This was different with star-shaped index because muti-shaped index could have richer information than the star-shaped index. Table.11 shows information of data sets and index. For each query entity, we collected Top-10 documents and traced potential links. Fig.4. Showes Top-10 documents searched by 30 bible and disease queries.

⁶ <http://wiki.dbpedia.org/Downloads36>

⁷ <http://home.bibleontology.com/>

⁸ <http://linkedlifedata.com/>

⁹ <http://hadoop.apache.org/>

¹⁰ http://bike.snu.ac.kr/~nansu/link_discover_eval/

Table 1. Data sets and index Size

File	Size
DBpedia page link(decompressed)	9.17 GB
Star-shaped index	10.6 GB
Muti-shaped index	21.4 GB
Page Rank Map	598 MB

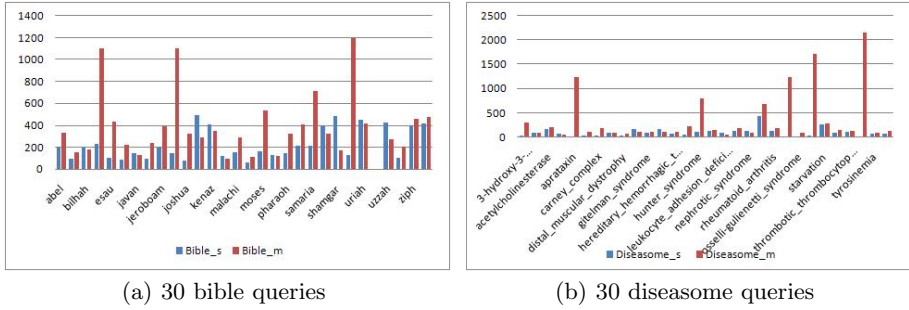


Fig. 1. Top-10 documents searched from star-index and muti-index of 30 bible queries and disease queries

We computed all potential links in Top-10 documents and got five most important links from both star-shaped graph index and muti-shaped graph index. Table.2 shows one example of query. We used “tyrosinemia” in disease as query word and both got 5 related links from disease muti-shaped index and star-shaped index. We invited domain expert to score each link scale from 0 to 5 as the standard expectation. 5 means most important or related links for the entity. We defined two standards for evaluated the link as related, using the 2.0 and 3.0 as scale marks for each links. We used the Precision, Recall and F-Measure to evaluate the links.

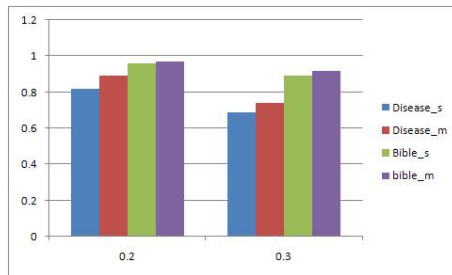


Fig. 2. F-measure results of bible and disease

Table 2. Query results of “tyrosinemia” from muti-shaped index and star-shaped index in Diseasome

Index	Entity Name
Muti-shaped index	Dbpedia:Alkaptonuria
	Dbpedia:Tyrosine
	Dbpedia:Ochronosis
	Dbpedia:Alkaptonuria
Star-shaped index	Dbpedia:amino_acid
	Dbpedia:Histidinemia
	Dbpedia:Fanconi_syndrome
	Dbpedia:Chromosome_15_%28human%29
	Dbpedia:Alkaptonuria

As the Fig.2 shown, the F-measure for Bible_m and Bible_s are 97% and 96% when the scale mark is 0.2. Diseasome_m and Diseasome_s are little worse than Bible, the F-measure are 89% and 82%. when scale mark raised to 0.3, the performance decreased 12%. Muti-shaped index has better results than Star-shaped index, which shows more related connection be considered, better performance the algorithm will give.

5 Conclusion and Future Work

In this paper, we introduced a method to discover new links in the Linked Open Data for web suppliers. Based on information retrieval technique, the approach indexed the data sets and used Page Rank and graph theory to analyze RDF document to filter links. We implemented our method using Dbpedia data sets and two open ontologies. The results showed our approach can discover new links with highly accuracy.

However, we still find some weakness of our approach. Firstly, we didn't consider the context of local data, for example, if local sets describe “Tim Berners-Lee” in the semantic domain, the results will not be constrained in Semantic Web. All the possible links in the different domain will be listed. Secondly, we only compute the entity importance and ignore the links' weights. This is not objective in computing importance of a link as every link in not flat and can influent the importance. Due to the weakness of our method, our future work will focus on context detection of local data set and link's importance calculation. The improvement, will help our approach discovering more accurate links for data suppliers.

Acknowledgment. Thanks professor Myungdae Cho for supporting us with Bible ontology and Doctor Soonjeong Koh for supporting us with evaluation. This research is supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2011.

References

1. Aleman-Meza, B., Halaschek, C., Arpinar, I.B., Sheth, A.: Context-aware semantic association ranking. In: Proceedings of SWDB, vol. 3, pp. 33–50. Citeseer (2003)
2. 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, vol. 30, pp. 564–575. VLDB Endowment (2004)
3. Bamba, B., Mukherjea, S.: Utilizing Resource Importance for Ranking Semantic Web Query Results. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372, pp. 185–198. Springer, Heidelberg (2005)
4. Delbru, R., Campinas, S., Tummarello, G.: Searching web data: an entity retrieval and high-performance indexing model. *Web Semantics: Science, Services and Agents on the World Wide Web* (2011)
5. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C., Sachs, J.: Swoogle: A semantic web search and metadata engine. In: Proc. 13th ACM Conf. on Information and Knowledge Management (2004)
6. Ding, L., Pan, R., Finin, T., 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)
7. Franz, T., Schultz, A., Sizov, S., Staab, S.: TripleRank: Ranking Semantic Web Data by Tensor Decomposition. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 213–228. Springer, Heidelberg (2009)
8. Hogan, A., Harth, A., Decker, S.: Reconrank: A scalable ranking method for semantic web data with context. In: 2nd Workshop on Scalable Semantic Web Knowledge Base Systems. Citeseer (2006)
9. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46(5), 604–632 (1999)
10. Lempel, R., Moran, S.: Salsa: the stochastic approach for link-structure analysis. *ACM Transactions on Information Systems (TOIS)* 19(2), 131–160 (2001)
11. 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), 37–52 (2008)
12. Toupikov, N., Umbrich, J., Delbru, R., Hausenblas, M., Tummarello, G.: Ding! dataset ranking using formal descriptions (2009)
13. Wu, G., Li, J.: Swrank: An approach for ranking semantic web reversely and consistently. In: SKG, pp. 116–121 (2007)
14. Wu, H., Cheng, G., Qu, Y.: Falcon-s: An ontology-based approach to searching-objects and images in the soccer domain. In: Supplemental Proceedings of ISWC (2006)

Probabilistic Multi-Context Systems

Marco Sotomayor^{1,*}, Kewen Wang¹, Yidong Shen², and John Thornton¹

¹ Griffith University, Australia

² Institute of Software, Chinese Academy of Sciences, China

marco-vinicio.sotomayorsanchez@griffithuni.edu.au

Abstract. The concept of contexts is widely used in artificial intelligence. Several recent attempts have been made to formalize multi-context systems (MCS) for ontology applications. However, these approaches are unable to handle probabilistic knowledge. This paper introduces a formal framework for representing and reasoning about uncertainty in multi-context systems (called p-MCS). Some important properties of p-MCS are presented and an algorithm for computing the semantics is developed. Examples are also used to demonstrate the suitability of p-MCS.

1 Introduction

The formalization of context is a critical building block towards the achievement of the semantic web vision [5]. In order to deliver accurate and unambiguous information, ontology-driven applications rely significantly on context modeling [6]. An *ontology* is a formal representation of shared terms and their relationships for an application domain. Ontologies have been widely applied in situations where the use and management of shared information are core issues (e.g. in medical applications [12,14]). According to the vision of the semantic web, information on the internet will also be represented as ontologies [5]. In this setting, explicit models of semantic information are needed in order to support information exchange. Since shared ontologies define a common understanding of terms for an application of interest, the use of ontologies makes it possible to communicate and exchange information between different users and systems on a semantic level. However, ontologies can be used only when a consensus about their contents is reached. Moreover, building and maintaining ontologies can become difficult in a dynamic, open and distributed domain such as the internet. To enhance the use and management of highly distributed ontologies, the framework of *contextual ontologies* has been established and an extension of OWL called *C-OWL* has been introduced in [1]. *C-OWL* is based on the theory of *multi-context systems* (MCS) [4]. To the best of our knowledge, *the problem of incorporating probabilistic information into multi-context systems is still open*

Multi-Context Systems (MCS), constitute one of the most recognized and mature formalizations of context in AI [15]. MCS are a generalization of Natural Deduction systems, which allow the use of different languages through a mechanism of tagged formulae [15]. This implies that in different languages or contexts, a logical proposition

* Corresponding author.

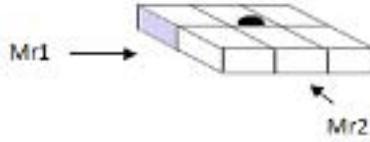


Fig. 1. Magic Box

can be interpreted in different ways. However, classical logic cannot express the degree of certainty of premises, nor the degree of certainty in conclusions derived from these premises (Williamson cited by [8]).

Figure 1 illustrates this situation using a typical Magic Box example [4] where Mr1 and Mr2 are unable to distinguish the depth of a ball inside a magic box. It is assumed that Mr1 and Mr2 are both almost blind but have knowledge about the compatibility relation between their different perspectives. So, Mr1 cannot answer with certainty if there is a ball on the right, he can only assume according his knowledge that “there is a ball on the right” with probability p . The same reasoning applies to Mr2, except that the probabilities for Mr2 are calculated in relation to Mr2’s context. A probabilistic multi-context system provides a language for representing what Mr1 and Mr2 know about their environment that allows them infer new probabilistic knowledge based on what they already know. Existing semantics of MCS are unable to handle probabilistic knowledge of contexts. For this reason the aim of this paper is to introduce a semantic framework for representing and reasoning about probabilistic knowledge in logic-based MCS.

The probabilistic logic approach in this paper is based on the work of [3,9,10], which dealt with formalization and semantics for uncertainty in logic programming. An important contribution of the current research is the introduction of probability theory into MCS, which provides more expressive languages in different contexts without losing the original logic of multi-context systems proposed by Giunchiglia [4]. The paper also shows that probabilistic multi-context systems can be reduced to MCS by simply assigning a probability of one to every proposition. This shows that MCS are a particular case of more general probabilistic multi-context systems. In addition, the idea of minimal information in context proposed by [13] is preserved in order to emphasize and contrast the relationship between MCS and p-MCS, and at the same time provide a probabilistic notion of the information entailed logically.

2 Probabilistic Multi-Context Systems

This section introduces the p -MCS framework for representing and reasoning about probabilistic information in multi-context systems. The first step in formalizing p-MCS is to specify the type of language used in each context.

Suppose that we have a set of contexts called K and denote context k as an element of K . Each context k is associated with a finite set A_k of labeled atoms of the form $k : a$ (a denotes an atom). Informally, $k : a$ means that atom a belongs to context k . Note that A_k contains only the atomic propositions needed to express the basic knowledge in context k . A propositional language L_k is constructed over A_k in a standard sense. A formula in context k can be expressed as $\phi \in L_k$. The next step is to introduce an uncertainty degree to a formula in L_k .

Let $k : \phi$ be a labeled formula and μ be a point probability between $[0, 1]$, then a formula function in a particular context ($k : \phi$) μ is called ***p-labeled formula***. Intuitively, ($k : \phi$) μ means that the probability of formula ϕ in context k is μ .

Definition 1. A *p-labeled rule* r is of the form $(k : F)\mu \leftarrow (k_1 : F_1)\mu_1, \dots, (k_n : F_n)\mu_n$ where $n \geq 0$, $(k : F)\mu$ and each $(k_i : F_i)\mu_i$ are *p-labeled formulas*.

Informally, this rule reads that if the probability of each F_i in context k_i is equal to μ_i for $i = 1, \dots, n$, then the probability of F in context k is equal to μ . The *p-labeled formula* $(k : F)\mu$ is called the head of the *p-labeled rule* r , denoted $head(r)$. The set $\{(k_1 : F_1)\mu_1, \dots, (k_n : F_n)\mu_n\}$ of *p-labeled formulas* is called the body of r , denoted $body(r)$. We remark that in the traditional probabilistic logic programming approaches [10,9], probabilities are assigned to atoms and the premises and head of the rule belong to the same context.

A local *p-rule* r for context k is a *p-labeled rule* without premises or body. Local knowledge of context k is a set of its local *p-rules*.

A bridge rule r for context k is a *p-labeled rule* such that (1) $head(r)$ has label k and (2) a *p-labeled formula* in $body(r)$ belongs to at least one context apart from context k . Such a rule provides a way for inferring new knowledge in context k from other contexts.

Definition 2. A *probabilistic multi-context system* (or *p-MCS*) T is of the form $[(R_1, B_1), \dots, (R_m, B_m)]$ where R_k is a set of local *p-rules* for context k and B_k is a set of bridge rules for context k for $k = 1, \dots, m$.

In the same way as for ordinary MCS [13], a probabilistic multi-context system is defined as a specification of contextual probabilistic information and inter-contextual probabilistic information flow. Contextual information can be specified through local *p-rules* (facts) and inter-contextual information flow through bridge rules.

Given a *p-MCS* $T = [(R_1, B_1), \dots, (R_m, B_m)]$, each context (R_k, B_k) represents two types of knowledge: (1) the knowledge \mathcal{B}_k directly derived from the local context k and other contexts through bridge rules; and (2) the knowledge η_k inferred from \mathcal{B}_k through probabilistic reasoning [10]. This type of reasoning problem is called probabilistic entailment [10], and is formally expressed as follows:

$$\zeta_k = \mathcal{B}_k \cup \eta_k \quad (1)$$

Example 1. The scenario illustrated in Figure 1 can be formalized as a p -MCS as follows:

$l =$ “The ball is on the left” $r =$ “The ball is on the right” $c =$ “The ball is in the center”.

$T = [(R_1, B_1), (R_2, B_2)]$. The two contexts are defined as $(R_1 = \{r_1\}, B_1 = \{r_3\})$, $(R_2 = \{r_2\}, B_2 = \{r_4\})$.

$$\begin{aligned} r_1 &: (1 : \neg r)0.5 && \leftarrow \\ r_2 &: (2 : c)0.5 && \leftarrow \\ r_3 &: (1 : l \vee r)0.75 && \leftarrow (2 : l \vee c \vee r)0.875 \\ r_4 &: (2 : l \vee c \vee r)0.875 && \leftarrow (1 : l \vee r)0.75 \end{aligned}$$

Then $\mathcal{B}_1 = \{(1 : \neg r)0.5, (1 : l \vee r)0.75\}$.

To explain how to determine η_1 , we need some basics of probabilistic reasoning.

One important concept in probability theory is the notion of worlds or atomic events [11]. Given a propositional language, we define a world as a Herbrand interpretation.

Let N_k be the number of propositions in A_k . Then there are 2^{N_k} possible worlds for context k . For example, if $A_1 = \{1 : l, 1 : r\}$, then there are four possible worlds W_j ($j = 1, 2, 3, 4$) and each such world W_j is associated with a probability w_j . We use LM_k to denote a finite set of possible worlds for context k : $LM_k = \{W_1, \dots, W_m\}$.

Definition 3. A contextual world probability density function for a context k is defined as a function $WP_k : LM_k \rightarrow [0, 1]$ satisfying $\sum_{W \in LM_k} WP_k(W) = 1$. Denote $WP_k(W_j) = w_j$; $0 \leq w_j \leq 1$, $1 \leq j \leq m$ (m denotes possible worlds for context k).

Because the possible worlds for every context are different, a world probability density function has to be defined for every context.

A p -local interpretation for context k is a pair $W : \mu$ of an interpretation W of L_k and an (associated) probability value μ . Let M_k be the set of p -local interpretations for context k : $M_k = \{W_1 : w_1, \dots, W_m : w_m\}$

Definition 4. A p -labeled chain is of the form: $c = \{c_1, \dots, c_m\}$ where $c_k \subseteq M_k$ for $k = 1, \dots, m$.

A p -labeled chain describes a world probability density function for every context k .

Every world is mutually exclusive and a proposition is equal to the disjunction of all the worlds where it holds [11]. For example: $l = \{W_1 \cup W_2\}$.

The general laws of the probability theory can be deduced through Kolmogorov’s axioms [11]. One of Kolmogorov’s axioms states that: $p(W_1 \vee W_2) = p(W_1) + p(W_2) - p(W_1 \wedge W_2)$. Because W_1 and W_2 are mutually exclusive, then: $p(l) = p(W_1 \vee W_2) = p(W_1) + p(W_2) - 0 = w_1 + w_2$

Given this demonstration it can be stated that the probability of a proposition is equal to the sum of the probabilities of the worlds where it holds (where a proposition is true).

Definition 5. A contextual probabilistic interpretation is a mapping from L_k to $[0, 1]$ defined as follows: For each $\phi \in L_k$: $\mathcal{I}_{wp_k}(\phi) = \sum_{W \models \phi} WP_k(W)$

Given a set of contextual probabilistic interpretations $\mathcal{I}_{wp_k}(\phi)$ in context k , a set of equations is generated under the following constraints:

- I. $\sum_{W_j \models \phi} W_j = \mu$, for all $\mathcal{I}_{wp_k}(\phi) = \mu$.
- II. $\sum_{j=1}^m w_j = 1$.
- III. $0 \leq w_j \leq 1, 1 \leq j \leq m$.

Now the notion of satisfiability for p-MCS can be introduced:

- A p-labeled chain c satisfies a p-labeled formula $(k : \phi)\mu$ iff $\mathcal{I}_{wp_k}(\phi) = \mu$
- A p-labeled chain c satisfies a p-labeled rule, iff whenever c satisfies the body of the rule $\text{body}(r)$ then the head of the rule $\text{head}(r)$ must be also satisfied.
- A p-labeled chain c satisfies a system T (i.e p-MCS) iff it satisfies every p-labeled rule of the system.

3 Minimal Probabilistic Entailed Chain/Fixpoint

The following section describes the process of constructing the probabilistic solution chain and the minimal probabilistic entailed chain and shows how to test if the contextual probabilistic interpretations are consistent in the set of equations that are generated. Finally, it is shown how to construct the contextual world probability density.

Let \mathcal{C} be the set of all p-labeled chains. It is possible to order the p-labeled chains according to the amount of information that they contain. A p-labeled chain c is less informative than c' ($c \preceq c'$), if for every context k , $c_k \supseteq c'_k$ [13].

Definition 6. A p-labeled solution chain c_p of a p-MCS is a p-labeled chain such that satisfying the p-MCS.

Based on [13], it can be argued that a minimal solution chain c_s in a non-probabilistic MCS contains all the logical entailments for every context k .

Definition 7. A minimal probabilistic entailed chain c_e contains all the probabilistic entailments η_k per every context k . $c_e : c_s \rightarrow [0, 1]$; $c_e = \{\eta_1, \dots, \eta_m\}$

i.e. c_e is the result of a mapping of all Herbrand interpretations of c_s to a probability between [0,1].

Proposition 1. Let c_p and c_e be a probabilistic solution chain and a minimal probabilistic entailed chain of a p-MCS, respectively. Then $c_p \supseteq c_e$

A non-probabilistic multi-context system is a particular case of a probabilistic multi-context system where every formula in the system is associated with a probability of one. The minimal solution chain c_s [13] in a non-probabilistic multi-context system discards Herbrand interpretations because these worlds have a probability of zero. However probabilistic multi-context systems have to keep these Herbrand interpretations in the solution chain because these worlds can have probabilities associated to them. For

this reason a probabilistic solution chain has to provide information about the minimal solution chain in non-probabilistic multi-context systems and the description of the contextual world probability density function. In cases where the probabilistic interpretations of the system do not comply with the probability theory, a probabilistic solution chain cannot be determined.

Roelofsen and Serafini [13] prove that every non-probabilistic multi-context system S has a unique minimal solution chain c_s . Then, according to proposition 1, c_p contains a unique minimal probabilistic entailed chain c_e .

In order to find the probabilistic solution chain c_p and at the same time the minimal probabilistic entailed chain c_e , the following analysis is conducted.

According to formula 4, \mathcal{B}_k denotes a finite set of sentences or formulas in a context k .

$$\mathcal{B}_k = \{(k : \varphi_1)\mu_1, \dots, (k : \varphi_m)\mu_m\}$$

Given this set of sentences, it can be inferred that: $\eta_k = \{(k : \varphi_1)\mu_1 \wedge, \dots, \wedge (k : \varphi_m)\mu_m\} = \{(k : \psi)\mu\}$. For a finite set of contexts K for $k = 1, \dots, m$, then: $c_e = \{\eta_1, \dots, \eta_m\} = \{(k_1 : \psi_1)\mu_1, \dots, (k_1 : \psi_m)\mu_m\}$.

Because this chain is unique in a system and constitutes the probabilistic entailment for system T , it is necessary to introduce an operator that computes this chain.

We can say that \mathcal{B}_k , the set of formulas that constitute the knowledge base in context k , is composed of facts and information obtained through bridge rules. For a chain c , if $c \models \text{body}(r)$ then $\text{head}(r) \in \mathcal{B}_k$.

Assuming that \mathcal{B}_1 (for context 1) contains two formulas φ and ψ : $\mathcal{B}_1 = \{(k_1 : \varphi)\mu_1, (k_1 : \psi)\mu_2\}$

It can be inferred: $\mathcal{N}_1 = \{(k_1 : \varphi)\mu_1 \wedge (k_1 : \psi)\mu_2\} = \{(k_1 : v)\mu\}$

In order to address the probabilistic consistency, the following Kolmogorov's axiom can be applied:

$$P(A \wedge B) = P(A) + P(B) - P(A \vee B)$$

This axiom can be expressed as:

$$\mathcal{I}_{wp_1}(\varphi \wedge \psi) = \mathcal{I}_{wp_1}(\varphi) + \mathcal{I}_{wp_1}(\psi) - \mathcal{I}_{wp_1}(\varphi \vee \psi) = \mu_1 + \mu_2 - \mathcal{I}_{wp_1}(\varphi \vee \psi)$$

Then: $\mathcal{N}_1 = (k_1 : v)\mu = (k_1 : v)\mathcal{I}_{wp_1}(\varphi \wedge \psi)$; $\mathcal{I}_{WP_1}(\varphi \wedge \psi) = \sum_{W \models \varphi \wedge \psi} WP_1(W) = \mu$; $\mu \in [0, 1]$

As can be seen, being given the probabilities of two formulas is not sufficient to find the probability of their conjunction. Also, the probabilistic interpretation $\mathcal{I}_{wp_1}(\varphi \wedge \psi)$ provides information of the Herbrand interpretations or worlds where $\varphi \wedge \psi$ holds, which is its logical entailment v . For example, if φ holds in worlds W_2, W_4 and ψ holds in worlds W_1, W_2, W_3 then $\varphi \wedge \psi$ holds in W_2 . This means that the logical entailment v is equal to W_2 .

Every probabilistic interpretation generates an equation. For example, if $m = 4$ (four worlds): $\mathcal{I}_{wp_1}(l) = w_1 + w_2 = 0.5$

The first row in Table 1 represents the default constraint and the second row represents the equation $w_1 + w_2 = 0.5$. Table 2 depicts a system of equations, where the

Table 1.

w_1	w_2	w_3	w_4	$0 \leq \mu \leq 1$
1	1	1	1	1
1	1	0	0	0.5

Table 2.

$w_1 w_2 \dots w_m$	$0 \leq \mu \leq 1$
$\mathcal{I}_{WP_1}(\varphi)$	μ_1
$\mathcal{I}_{WP_1}(\psi)$	μ_2
$\mathcal{I}_{WP_1}(\varphi \vee \psi)$	$\mu_1 + \mu_2 - \mathcal{I}_{WP_1}(\varphi \wedge \psi)$
$\mathcal{I}_{WP_1}(\varphi \wedge \psi)$	μ

goal is to obtain the probabilistic entailment \mathcal{N}_1 , given φ and ψ . These equations are processed in such way that the probabilistic consistency in the right column of the table can be verified and $\mathcal{N}_1 = (k_1 : v)\mu$ is obtained in the last row. The logical entailment v is implicit in the equation ($w_1 + w_2 = 0.5$, Table 2), because Herbrand interpretations that hold correspond to a value of one in the equation and Herbrand interpretations that do not hold correspond to a value of zero in the equation. Let this process be the \mathcal{KE} function.

Definition 8. *The \mathcal{KE} function for $k = 1, \dots, m$ is of the form :*

$$\mathcal{KE}(\mathcal{I}_{WP_k}(\psi)) = \mathcal{I}_{WP_k}(\mathcal{B}_k \wedge \psi) = \mathcal{N}_k; \text{ if a p-chain } c \models \text{body}(r) \text{ then head}(r) \in \mathcal{B}_k$$

\mathcal{B}_k represents the knowledge base in a context k . The \mathcal{KE} function can be applied iteratively for many p-formulas. Also, if the same process is applied for all contexts then c_e can be obtained.

Theorem 1. *\mathcal{KE} is monotonic.*

Proof. Given a default constraint in \mathcal{LE}_k and an interpretation φ in \mathcal{KE} :

$$(i) \mathcal{KE}(\mathcal{I}_{wp_k}(\varphi)) = \mathcal{I}_{wp_k}(\varphi \wedge 1) = \mathcal{I}_{wp_k}(\varphi)$$

Then adding another interpretation ψ in \mathcal{KE} : $(ii) \mathcal{KE}(\mathcal{I}_{wp_k}(\psi)) = \mathcal{I}_{wp_k}(\varphi \wedge \psi)$.

If \mathcal{KE} is monotonic: whenever $\mathcal{I}_{WP_k}(\psi) = \mathcal{I}_{WP_k}(\varphi)$, $(iii) \mathcal{KE}(\mathcal{I}_{wp_k}(\psi)) \leq \mathcal{KE}(\mathcal{I}_{wp_k}(\varphi))$

Replacing (i) and (ii) in (iii): $\mathcal{I}_{wp_k}(\varphi \wedge \psi) \leq \mathcal{I}_{wp_k}(\varphi)$.

A Kolmogorov’s axiom states that:

If ϕ logically implies λ then $P(\phi) \leq P(\lambda)$ because $(\varphi \wedge \psi) \subseteq \varphi$ then $(\varphi \wedge \psi) \rightarrow \varphi$

Then if $(\varphi \wedge \psi)$ implies φ : $\mathcal{I}_{WP_k}(\varphi \wedge \psi) \leq \mathcal{I}_{WP_k}(\varphi)$

Because \mathcal{KE} is *monotonic* and sets of all p-labeled chains (\mathcal{C}, \ll) form a complete lattice, according to the Knaster-Tarsky theorem (cited by [7]), it can be stated that:

Theorem 2. *\mathcal{KE} has a least fixpoint*

This least fixpoint contains c_e or the set of all the probabilistic entailments \mathcal{N}_k .

Example 2. *Continuing Example 1 to find c_p and c_e :*

Step 1:

$$r_1 : (1 : \sim r)0.5 \quad \leftarrow$$

c^\perp denotes the initial p -labeled chain (containing all p -local models with unknown probabilities). Because $\text{body}(r_1)$ is empty then $c^\perp \models \text{body}(r_1)$. That means that $\text{head}(r_1) \in \mathcal{B}_1$

$$\mathcal{I}_{wp_1}(\sim r) = w_2 + w_4 = 0.5$$

$$c_1 = \left\{ \begin{array}{l} [\{l, r\} : w_1, \{1, \sim r\} : w_2, \{\sim l, r\} : w_3, \{\sim 1, \sim r\} : w_4]_1, \\ [\{1, c, r\} : w_1, \{1, c, \sim r\} : w_2, \{l, \sim c, r\} : w_3, \{l, \sim c, \sim r\} : w_4, \{\sim l, c, r\} : w_5, \\ \{\sim l, c, \sim r\} : w_6, \{\sim l, \sim c, r\} : w_7, \{\sim l, \sim c, \sim r\} : w_8]_2 \end{array} \right\}$$

Step 2:

$$r_2 : (2 : c)0.5 \quad \leftarrow$$

Because $\text{body}(r_2)$ is empty then $c_1 \models \text{body}(r_2)$. That means that $\text{head}(r_2) \in \mathcal{B}_2$

$$\mathcal{I}_{wp_2}(c) = w_1 + w_2 + w_5 + w_6 = 0.5$$

$$c_2 = \left\{ \begin{array}{l} [\{1, r\} : w_1, \{1, \sim r\} : w_2, \{\sim l, r\} : w_3, \{\sim l, \sim r\} : w_4]_1, \\ [\{1, c, r\} : w_1, \{1, c, \sim r\} : w_2, \{l, \sim c, r\} : w_3, \{l, \sim c, \sim r\} : w_4, \{\sim 1, c, r\} : w_5, \\ \{\sim 1, c, \sim r\} : w_6, \{\sim l, \sim c, r\} : w_7, \{\sim l, \sim c, \sim r\} : w_8]_2 \end{array} \right\}$$

Step 3:

$$r_3 : (1 : (l \vee r))0.75 \quad \leftarrow (2 : (l \vee c \vee r))0.875$$

$$\mathcal{I}_{wp_2}(l \vee c \vee r) = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 = 0.875$$

$$w_8 = 1 - 0.875 = 0.125$$

$$c_2 \models (2 : (l \vee c \vee r))0.875 \text{ then } \text{head}(r_3) \in \mathcal{B}_1$$

$$\mathcal{I}_{wp_1}(l \vee r) = w_1 + w_2 + w_3 = 0.75$$

Step 4:

$$r_4 : (2 : (l \vee c \vee r))0.875 \quad \leftarrow (1 : (l \vee r))0.75$$

$$\mathcal{I}_{wp_1}(l \vee r) = w_1 + w_2 + w_3 = 0.75$$

w_1	w_2	w_3	w_4	
1	1	1	1	1
0	1	0	1	0.5
1	1	1	1	$1.5-w_2-w_4$
0	1	0	1	0.5
1	1	1	0	0.75
1	1	1	1	$1.25-w_2$
0	1	0	0	0.25
1	1	1	0	0.75
1	1	1	0	$1-w_2$
0	1	0	0	0.25

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	
1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	0	0	0.5
1	1	1	1	1	1	1	1	$1.5-w_1-w_2-w_5-w_6$
1	1	0	0	1	1	0	0	0.5
1	1	1	1	1	1	1	0	0.87
1	1	1	1	1	1	1	0	$1.37-w_1-w_2-w_5-w_6$
1	1	0	0	1	1	0	0	0.5
1	1	1	1	1	1	1	0	0.87
1	1	1	1	1	1	1	0	$0.87-w_1-w_2-w_5-w_6$
1	1	0	0	1	1	0	0	0.5

$$c_3 \models (1 : (l \vee r))0.75 \text{ then } \text{head}(r_4) \in \mathcal{B}_2(2 : (l \vee c \vee r))0.875$$

$$\mathcal{I}_{wp_2}(l \vee c \vee r) = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 = 0.875$$

$$c_4 = \left\{ \begin{array}{l} \{ \{l, r\} : w_1, \{l, \sim r\} : \mathbf{0.25}, \{\sim l, r\} : w_3, \{\sim l, \sim r\} : 0.25 \}_1, \\ \{ \{l, c, r\} : w_1, \{l, c, \sim r\} : w_2, \{l, \sim c, r\} : w_3, \{l, \sim c, \sim r\} : w_4, \\ \{ \sim l, c, r\} : w_5, \{ \sim l, c, \sim r\} : w_6, \{ \sim l, \sim c, r\} : w_7, \{ \sim l, \sim c, \sim r\} : 0.125 \}_2 \end{array} \right\}$$

$$c_3 = c_4 ; c_p = c_4$$

$$c_e = \left\{ \begin{array}{l} \{ \{l, \sim r\} : \mathbf{0.25} \}_1, \\ [\{ \{l, c, r\}, \{l, c, \sim r\}, \{ \sim l, c, r\}, \{ \sim l, c, \sim r\} \} : \mathbf{0.5}]_2 \end{array} \right\}$$

Then, c_e can be interpreted as: “The probability that there is a ball on the left and not on the right (relative to Mr1) is 0.25” and “The probability that there is a ball in the center (relative to Mr2) is 0.5”.

Although the precise connection between contextual ontologies (e.g C-OWL) and p-MCS is not being developed in this paper, we can extend Example 2, creating a mapping between the probabilistic entailment obtained in context 1 and an ontology, using the following bridge rule:

$$(onto : Ball(ball, left))0.25 \quad \leftarrow \quad (1 : l \wedge \neg r)0.25$$

4 Conclusion

This paper has proposed a theoretical approach to the introduction of uncertainty in multi-context systems. A more expressive semantics has been presented in order to extend the notion of probability in multi-context systems. This additional expressiveness brings new constraints that have to be harmonious and consistent with probability and logic theory. In order to address this situation, a probabilistic logic semantic approach based on the works of [39,10] has been extended to MCS. Also, a technique that deals with probabilistic inconsistency and the deduction of a minimal entailed chain has been incorporated to the framework.

There are some additional observations worth making about the characteristics of the framework. Firstly, MCS can be embedded in p-MCS assigning to the propositions a probability of one. This characteristic and the deduction of a minimal entailed chain mean MCS can be incorporated into a more general framework. However, there are practical limitations that need to be addressed in future work. For example, the joint probability function or contextual world probability density function has to be specified explicitly in tabular form, which requires exponentially many parameters. This circumstance could be a limitation in practical applications [2]. Nevertheless, this strategy can be used as a theoretical foundation for different approaches [11]. Finally, Bayesian Networks have been successful the last two decades in reducing the complexity of computation of joint probability functions [2]. This suggests that future work should look at incorporating the notion of conditional probabilities into p-MCS through Bayesian Networks.

References

1. 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)
2. Cozman, F., Haenni, R., Romeijn, J., Russo, F., Wheeler, G., Williamson, J.: Combining probability and logic. *Journal of Applied Logic* (2007)
3. Dekhtyar, M., Dekhtyar, A.: Revisiting the Semantics of Interval Probabilistic Logic Programs. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 330–342. Springer, Heidelberg (2005)
4. Giunchiglia, F., Ghidini, C.: Local model semantics, or contextual reasoning = locality+compatibility. Technical report. Istituto Trentino di Cultura (1997)
5. Hendler, J., Berners-Lee, T.: From the semantic web to social machines: A research challenge for AI on the world wide web. *Artificial Intelligence* (2010)
6. Bao, J., Tao, J., McGuinness, D., Smart, P.: Context representation for the semantic web. In: *Proceedings of the WebSci 2010* (2010)
7. Lloyd, J.: *Foundations of logic programming*. Springer, Heidelberg (1987)
8. Ng, K., Lloyd, J.: Probabilistic reasoning in a classic logic. *Journal of Applied Logic* (2009)
9. Ng, R., Subrahmanian, V.: Probabilistic logic programming. *Information and Computation* (1992)
10. Nilsson, N.: *Probabilistic logic*. *Artificial Intelligence* (1986)
11. Norvig, P., Russell, S.: *Artificial intelligence: a modern approach*. Prentice Hall/Pearson Education (2003)
12. Rector, A., Nowlan, W.: The GALEN project. *Computer Methods and Programs in Biomedicine* (1994)
13. Roelofsen, F., Serafini, L.: Minimal and absent information in contexts. In: *Proceedings of 19th IJCAI* (2005)
14. Schulz, S., Cornet, R., Spackman, K.: Consolidating SNOMED CT's ontological commitment. *Applied Ontology* (2011)
15. Serafini, L., Bouquet, P.: Comparing formal theories of context in AI. *Artificial Intelligence* (2004)

Web Schema Construction Based on Web Ontology Usage Analysis

Jamshaid Ashraf and Maja Hadzic

DEBII, Curtin University, Western Australia

jamshaid.ashraf@gmail.com, maja.hadzic@cbs.curtin.edu.au

Abstract. The ultimate vision of the semantic web is to enable computers to understand and process the information published on the web. This vision is being primarily achieved by web ontologies which semantically annotate the data. In order to effectively access the structured data mainly published in RDF format, one needs to understand not only the prevalent vocabularies being used by the community, but also the extent and the patterns of its usage. In this paper, we achieve this by proposing a framework that analyzes the domain ontology usage and rank terms (classes, properties and attributes) based on multi-criteria characteristics that include population, coverage, and structure. We consider a purpose-built RDF dataset to select the popular terms and construct the schema based on the ranking, enabling the semantic web application to acquire information from the web of data effectively and efficiently.

Keywords: Ontology Usage Analysis, RDF, Semantic Web, eCommerce.

1 Introduction

Humans have been the only primary actors in the traditional web, whereas in the semantic web, computers have become equal citizens in this emerging social machines (digital) ecosystem. This phenomenon has extended the web into a decentralized knowledge platform for dissemination and sharing of information. Ontologies are at the heart of this process, providing terms and relationship between terms, agreed upon by users, in a machine understandable and process-able format. Terms defined by ontologies are then used to annotate the data, providing semantics and structure to the published information. The ever-growing availability of semantics and structures on the web [1] is transforming the web into a global database where information can be queried, not just merely searched.

Over the past five years, a few significant developments have given momentum to the widespread adoption of semantic web technologies. Simplicity of Linked Data principles, the tremendous success of the Linked Open Data (LOD) project¹ and the recognition and consumption of semantic data by search engines (Yahoo and Google)

¹ <http://richard.cyganiak.de/2007/10/1od/>

are a few of the most significant events that have helped to bootstrap semantic data on the web. Hence, we now have billions of triples on the web in different domains such as health care and life sciences (HCLS), social spaces, digital libraries, financial services, oil and gas exploration, and eCommerce, to name a few. Given all this, now we are at the *point* where we can start conducting empirical studies to analyze the data and perform meaningful measurements. We can provide checkpoints allowing us to analyze the data from different aspects as highlighted in [5] such as data quality, data management and ontology usage.

Having reached the point where we have billions of triples and thousands of ontologies [6], it is of paramount importance to conduct a systematic study on the actual use of domain-focused terminological knowledge on the web and to provide feedback on the improvement and refinement process of the semantic web data life cycle. Therefore, there is a need to develop a systematic approach for evaluating and analyzing the particular ontology usage, its adoption and uptake by different users.

In this paper, we present a framework to evaluate the RDF dataset and the domain specific ontology/vocabulary usage. The result of usage analysis along with the structural characteristics is then used to rank the terms based on multi-criteria characteristics that include population, coverage and structure. To conduct the empirical study, in order to reflect the real-world setting, an RDF dataset comprised of hundreds of different websites is collected from the web in eCommerce. We propose a set of metrics to measure the ontology usage, ontology instantiation and population.

The remainder of the paper is organized as follows. Section 2 discusses the overall approach used in this research. In section 3, we discuss the dataset collection, dataset characteristics and the experiment. Section 4 discusses related work and presents the conclusion, with the direction of future work indicated in section 5.

2 Analyzing the Ontology Usage in RDF Dataset

In the following, we present the details of our approach to evaluate the RDF dataset and identify the different domain ontologies being used and based on their usage, construct the web schema. The approach we follow is generic and can be virtually applied to any domain. However, in order to empirically ground our study, we selected the web eCommerce domain (vertical industry) for this research as we believe that this domain has the largest real instantiation of semantic data on the web after the HCLS (Healthcare and Life Science) domain. The approach is comprised of the following steps and details are presented in subsequent sections.

- Construct the vocabulary graph. A populated vocabulary graph, where each node represents the vocabulary used in the dataset, is the result of this step.
- Analyse the vocabularies to find the usage patterns and ontology instantiation. For detailed investigation, we use the Ontology USage Analysis Framework (OUSAF) [3] which provides a set of metrics for measuring the terminology usage defined by the domain ontologies.

- Rank the terms of each ontology (represented by vocabulary graph) based on multi-criteria characteristics that include population, semantic coverage and semantic relationships.
- Construct the web schema covering the key concepts and attributes based on their ranking.

2.1 Construction of the Vocabulary Graph

In this section, first we present the definitions used throughout the paper and then discuss the construction of a vocabulary graph.

Vocabulary Graph. A vocabulary graph VG is a tuple $\langle V, E \rangle$, where n is a node ($n \in V$) such that n is the vocabulary namespace mentioned in the dataset based on a simple heuristic: a triple with *rdf:type* predicate, object is the class and the subject is the instance of that class. $e \in E$ is an edge of graph V linking two nodes n_1 and n_2 such that there is a triple in the dataset where n_1 is the namespace of the subject and n_2 is the namespace of the object. Hence, we will have a graph representing all the vocabulary namespaces (ontologies) available in the dataset. The vocabulary graph provides the list of the vocabularies being used in the RDF dataset. The vocabulary graph has as many nodes as there are vocabularies used to annotate the structured data on the web. We queried the dataset to access the triples with a pattern of having *rdf:type* used as predicate, indicating that the object is an RDF resource defined as a class in the ontology document. The URIs of the RDF resource matching the abovementioned patterns are used to extract the namespace URI of the vocabulary and the term defined by the namespace. The nodes' content of vocabulary graph is then aggregated to build the list of vocabularies used and the number of terms (terminological knowledge) of each vocabulary used in the dataset.

2.2 Vocabulary/Ontology Usage Analysis

Vocabulary usage analysis is conducted using the OUSAF framework [3] which provides a set of metrics to measure the usage, taking into consideration the ontology population² and the structural characteristics.

Concept Richness (CR). The concept richness of a concept $CR(C)$ of a given vocabulary is calculated by adding the number of non-hierarchical relationships it has with other concepts and the data properties defined in the ontology document. $CR(C)$ is calculated using the following formula:

$$CR(C) = |P_C| + |A_C|$$

P_C returns the number of object properties of C while A_C returns the number of data properties.

² [2] defines ontology population as having occurred when an ontological term (i.e. concept, property or individual) is *used* to annotate data.

Concept Usage (CU). In concept usage, we measure the instantiation of a particular concept in a dataset. In the RDF graph, we calculate the triples $\langle ?instance \text{ rdf:type } v:Con \rangle$, where Con is the concept defined by the ontology v . The concept usage $CU(C)$ is measured as follows:

$$CU(C) = |t = (s,p,o) \mid p = \text{rdf:type}, o=C|$$

$CU(C)$ helps in ranking the concepts based on their instantiation in the dataset.

Relationship Value (RV). In ontology, a concept is linked with other concepts using a typed relationship, allowing semantic web applications to use the information in a more effective manner. Typed relationships are known as object properties linking the instances of the concepts defined as the domain with the instances of the concept defined as range of the property.

$$RV(P) = |dom(P)| + |range(P)|$$

dom and $range$ are the domain and range of the property P respectively in an ontology document.

Relationship Usage (RU). Relationship usage calculates the triples in which object property is used as a predicate of the RDF statement. Relationship value is helpful in ranking and indexing the properties based on their usage for efficient information retrieval.

$$RU(P) = |t := (s,p,o) \mid p = P|$$

Similarly, we have Attribute Value (AV) and Attribute Usage (AU) to measure the data property value and usage in the dataset as follows:

$$AV(A) = |dom(A)|$$

$$AU(A) = |t := (s,p,o) \mid p \in A, o \in L|$$

The aforementioned metrics are implemented as part of the ontology usage analysis framework and can be used for any number of ontologies.

2.3 Ranking the Terms

To rank the terms based on the empirical data, the rank of a given term t of vocabulary/ontology v is calculated by aggregating the *TermUsage* in the dataset. To offer the preferential aspect to the ranking, weights are used to adjust the priority of each measure accordingly. To normalize the measures and cast them into a similar range, we divide each measure value by the maximum measure value of each term represented in the vocabulary graph.

$$TermUsage_t = \frac{w_r R_t + w_u U_t}{2w_r w_u} \times \frac{U_t}{R_t}$$

$$\text{where } \begin{cases} \text{if term is Concept then } R_t = CR(C) \text{ and } U_t = CU(C) \\ \text{if term is Object Property then } R_t = RV(P) \text{ and } U_t = RU(P) \\ \text{if term is Data Property then } R_t = AV(A) \text{ and } U_t = AU(A) \end{cases}$$

The ranking of each term of ontologies extracted from the RDF dataset helps in constructing the Web Schema to represent the prevailing structure and semantic metadata for semantic web client applications. Additionally, vocabulary graphs help us to better understand the co-usability of different vocabularies in a specific domain and identify the relationship(s) between them.

3 Dataset and Experiment

To conduct the empirical study on the RDF data and analyse the vocabulary usage in a specific (focused) domain, we build the dataset to serve as a representative sample of the web of data currently on the web. The collected dataset is sufficiently representative and provides a snapshot of actual domain specific semantic data enabling us to undertake meaningful measurements and understand how data is really being used. We selected the eCommerce domain for this study and collected a dataset that is predominantly comprised of the RDF data published in eCommerce websites. In the following, we discuss the dataset collection approach and composition characteristics.

3.1 Dataset Collection

In our previous works [4], we observed that approximately 90% of the semantic eCommerce data (RDF data) is published on the web by embedding the structured data snippets within HTML pages using different formats such as RDFa, microformats and microdata. We focused on websites using the RDFa format for mainly two reasons: first, several tools such as crawlers and parsers are available for RDFa to extract the triples and populate the database; secondly, the new breed of semantic search engines such as Sindice, Swoogle, Falcons and Watson index the web documents with RDFa snippets. We used the Sindice API to retrieve all the URIs (URLs to be specific, since we were interested in knowing the web domains hosting semantic eCommerce data) returned by search engine in response to query against eCommerce, Product, Price, Offer and Company keywords. Currently, most of the web documents containing RDFa do not have links to any of the other documents in the same web domain (website) and it was impractical to use a crawler based on RDF link traversal. Therefore, we provided the list of web document URLs to the crawler to extract the RDF data and transform it into a single format, ie. RDF/XML syntax using the Any23 APIs.

3.2 Analysis of Vocabulary Usage

The generated vocabulary graph (VG) from the collected data set contains 33 unique URIs. Each URI denotes an ontology/vocabulary used to describe the eCommerce related information and the rank is the aggregated average value of the ontology usage (including all the terms which have instantiation in the dataset) as listed in Table 1.

Table 1. List of ontologies found in dataset

Prefix	Ontology URI	Rank
gr	http://purl.org/goodrelations/v1#	0.982
foaf	http://xmins.com/foaf/0.1/	0.235
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	0.222
rdfs	http://www.w3.org/2000/01/rdf-schema#	0.216
v	http://rdf.data-vocabulary.org/#	0.085
vocab	http://www.w3.org/1999/xhtml/vocab	0.063
dcterms	http://purl.org/dc/terms/	0.028
og	http://opengraphprotocol.org/schema/	0.021
rev	http://purl.org/stuff/rev#	0.018
<i>logic</i>	http://www.logicpass.com/semanticweb.owl	0.014
owl	http://www.w3.org/2002/07/owl#	0.012
<i>kica</i>	http://www.kica-jugendstil.com/semanticweb.rdf	0.011
<i>bunt</i>	http://www.buntegeschenke.de/semanticweb.rdf#	0.009
vCard	http://www.w3.org/2006/vcard/ns#	0.007
audio	http://purl.org/media/audio#	0.005
virt	http://www.openlinksw.com/schemas/virtrdf#	0.004
dc	http://purl.org/dc/elements/1.1/	0.004
sioc	http://rdfs.org/sioc/ns#	0.003
frbr	http://vocab.org/frbr/core#	0.003
vso	http://purl.org/vso/ns#	0.003
commerce	http://purl.org/commerce#	0.002
vCard-rdf	http://www.w3.org/2001/vcard-rdf/3.0#	0.002
<i>acig</i>	http://www.acigroup.co.uk/semanticweb.rdf#	0.001
vahoo	http://search.yahoo.com/searchmonkey/commerce/	0.001
void	http://rdfs.org/ns/void#	0.001
powder	http://www.w3.org/2007/05/powder-s#	0.001
<i>oplweb</i>	http://data.openlinksw.com/oplweb#	0.000
<i>loko</i>	http://lokool.com/semanticweb.owl#	0.000
scovo	http://purl.org/NET/scovo#	0.000
wgs84	http://www.w3.org/2003/01/geo/wgs84_pos	0.000
eClass	http://www.ebusiness-unibw.org/ontologies/eClass/5.1.4/#	0.000
<i>lokoo</i>	http://lokool.com/extendedgoodrelations.owl	0.000
pto	http://www.productontology.org/id/	0.000

In our calculation, to observe the unbiased usage pattern, we set all the weights to 1 which can be tuned to different values based on the preferential adjustment requirements. For example, if the extracted web schema is intended to be used for querying the semantic web structured data with high recall, then usage weights should be set higher to incentive measures. In the dataset, a few of the ontologies (prefixes in italic in Table 1) for which we could not find any known prefix in prefix.cc database are used by a few of the publishers (web domains) and URIs in red are the ones for which we were not able to retrieve the original ontology document.

Using the list of ontologies reported by the vocabulary graph, we retrieved the authoritative description of each ontology from its hosting service in order to validate the terms analyzed by the OUSAF framework before applying multi-criteria ranking approach. In Table 2, we report the terms usage analysis with their ranking for two of the highly populated ontologies of our dataset; **foaf** and **gr**. We present only a partial list of the gr terms so as to fit within the space to increase readability.

Note that a few of the terms have a ranking value close to zero, but this is after applying the 3 decimal precision format for readability. However, their usage is far less than others. Based on the usage analysis and ranking, we constructed the web

schema representing the prevalent structure of semantic data on the web in the eCommerce domain. In Table 3, we present the statistics of the web schema content to summarize the experiment results.

Our vocabulary graph (VG) initially had 33 namespaces of vocabularies and 29 of these were validated by their authoritative description which was used by the OUSAF framework to rank the terms. The resultant web schema is comprised of 11 ontologies and 49 concepts, 50 object properties and 49 attributes. gr is the dominating ontology in web schema and sioc is the least used vocabulary. Since we focused on the eCommerce domain, gr’s dominance is not surprising. However, it was quite interesting to discover that several other ontologies are being co-used in the eCommerce domain.

Table 2. List of ontologies found in dataset

Concept	foaf		gr	
	<i>term</i>	<i>rank</i>	<i>term</i>	<i>rank</i>
	Person	0.536	Offering	0.906
Document	0.016	UnitPriceSpecification	0.476	
Organization	0.008	TypeAndQuantityNode	0.431	
OnlineAccount	0.006	ProductOrServicesSomeInstancesPlaceholder	0.416	
		QuantitativeValueFloat	0.220	
		ProductOrServiceModel	0.209	
		BusinessEntity	0.057	
		QuantitativeValue	0.032	
Object Property (relationships)	<i>term</i>	<i>rank</i>	<i>term</i>	<i>rank</i>
	depiction	0.886	hasBusinessFunction	0.821
	thumbnail	0.689	offers	0.725
	page	0.656	hasPriceSpecification	0.485
	homepage	0.230	eligibleCustomerTypes	0.438
	topic	0.125	typeOfGood	0.427
	member	0.012	includesObject	0.424
	mbox	0.010	acceptedPaymentMethods	0.418
	primaryTopic	0.005	availableAtOrFrom	0.409
Datatype Property (attributes)	<i>term</i>	<i>rank</i>	<i>term</i>	<i>rank</i>
	name	0.604	eligibleRegions	1.000
	accountName	0.002	hasUnitOfMeasurement	0.480
			validFrom	0.379
			validThrough	0.379
			hasCurrencyValue	0.226
			hasCurrency	0.221
			amountOfThisGood	0.190
			hasStockKeepingUnit	0.139

Table 3. Web Schema content and statistics

	gr	foaf	v	dcterms	rev	vCard	audio	sioc	vso	void	scovo	TOT
Concepts	22	4	6	2	1	3	2	-	6	2	1	49
ObjProperty	21	11	-	-	3	6	-	1	8	-	-	50
DataProperty	31	2	-	-	1	10	-	-	5	-	-	49
TOTAL	68	17	6	2	5	19	2	1	19	2	1	

4 Related Work

In the last few years, semantic web technologies have increasingly been adopted and consequently, we are seeing a tremendous amount of structured data being published on the web in almost every domain [5]. Given this, the use of ontologies and vocabularies on the web has also increased [6] and subsequently has increased the importance of analyzing ontology usage in order to better to understand its adoption and usage, and the current trends.

Several researches have proposed different frameworks and sets of measurements to evaluate ontologies. For example, in [7], the authors present a framework and set of metrics to evaluate the richness, connectivity, fullness and cohesiveness of a given ontology. Although the metrics proposed in their framework provides insight into ontology structures, most of their study is carried out on either a small set of data or on test data generated to obtain a sample representation of a dataset. In [2], the authors have considered the actual social data by conducting their study on FOAF documents published on the web. This work focused solely on FOAF vocabulary and did not offer any generic framework to allow similar studies to be conducted on other vocabularies.

Keeping in view the abovementioned points, in this work, we discuss the implementation of a generic framework for analyzing semantic web data with focus on domain ontologies and vocabularies.

5 Conclusion and Future Work

In this paper, we analyzed the RDF data currently published on the web in the eCommerce domain in order to analyze the ontology usage and rank the terms to construct Web Schema providing the semantic coverage of eCommerce data on the web. As future work, we intend to extend this work to include data and ontologies from other domains such as entertainment, healthcare and eGovernment and analyze the data patterns and use of different vocabularies and ontologies.

References

1. Mika, P., Meij, E., Zaragoza, H.: Investigating the Semantic Gap through Query Log Analysis. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunaryan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 441–455. Springer, Heidelberg (2009)

2. Ding, L., Zhou, L., Finin, T., Joshi, A.: How the semantic web is being used: An analysis of foaf documents. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS 2005) - Track 4, vol. 04, p. 113C. IEEE Computer Society, Washington, DC, USA (2005),
<http://portal.acm.org/citation.cfm?id=1042435.1042928>
3. Jamshaid, A., Maja, H.: Domain Ontology Usage Analysis Framework. In: The International Conference on Semantics, Knowledge and Grids, SKG 2011 (2011) (in press)
4. Ashraf, J., Cyganiak, R., O'Riain, S., Hadzic, M.: Open ebusiness ontology usage: Investigating community implementation of GoodRelations. In: Linked Data on the Web Workshop (LDOW 2011) at WWW 2011, the Proceedings of the Linked Data on the Web WWW 2011 Workshop (LDOW 2011), Hyderabad, India, March 29 (2011)
5. Auer, S., Lehmann, J.: Creating knowledge out of interlinked data. *Semantic Web* 1(1-2), 97–104 (2010)
6. Hu, W., Chen, J., Zhang, H., Qu, Y.: How Matchable are Four Thousand Ontologies on the Semantic Web. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part I. LNCS*, vol. 6643, pp. 290–304. Springer, Heidelberg (2011)
7. Tartir, S., Arpinar, I.B., Moore, M., Sheth, A.P., Aleman- Meza, B.: Ontoqa: Metric-based ontology quality analysis. In: Proceedings of IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources (2005)

Building Linked Open University Data: Tsinghua University Open Data as a Showcase

Yuanchao Ma, Bin Xu, Yin Bai, and Zonghui Li

Knowledge Engineering Group, Department of Computer Science and Technology,
Tsinghua University, Beijing, China

{myc, xubin, baiy, lzh}@keg.cs.tsinghua.edu.cn

<http://keg.cs.tsinghua.edu.cn>

Abstract. Linked Open University Data applies semantic web and linked data technology to university data scenario, aiming at building interlinked semantic data around university information, providing possibility for unified inner- and inter- school information query and comparison. This paper proposes a general process of building linked open university data, with procedures covering choosing datasets and vocabularies, collecting and processing data, building RDF and interlink, etc. Tsinghua University Open Data is used to demonstrate the process. Tsinghua University consist of 5 well-formed, interconnected datasets, with a number of interesting applications has been built on top of them. Finally, remarkable points about data collecting and processing is discussed.

Keywords: Open Data, Linked Data, Linked Open University Data, SPARQL server, Semantic Portal.

1 Introduction

As tools and standards related to the Semantic Web are becoming comprehensive and stable, how to build high-quality semantic data and how to make use of these semantic data, become two major challenges in the development of the Semantic Web. *Linked Data*^[1] project describes a well-acknowledged standard method of publishing interlinked structured data based on the Semantic Web technology^[2]. The *LOD*^[3] project now has published hundreds of datasets using Linked Data standard, covering a large variety of domains. ^[4]

Linked Open University Data builds linked open data around universities and academic institutions. Linked open university data under a unified schema provides consistent data access to different universities, offering convenience for inner-university and inter-university information management.

However, building linked open university data faces some major challenges. First of all, there is no unified, well accepted vocabulary for describing university-related information. What is more, as existing well-structured data about university which is publicly accessible is very limited, collecting and organizing raw

¹ <http://linkeddata.org>

² <http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

data needs extra work. Establishing interlink to connect different datasets together is also great challenge, considering complex relationship between different pieces of university data.

In this paper, we present a general process of building linked open university data. We first list common data categories about university information, then give a summary of useful vocabularies for describing university datasets. We then discuss common approaches to retrieve raw data about university, and the preprocessing and cleaning work necessary to get these raw data ready for converting, followed by naming and linking strategies to make them correspond with Linked Data standard. Then we describe techniques and tools used to convert well-organized structured data of different formats to RDF format.

While describing the general process, *Tsinghua University Open Data* (<http://data.cs.tsinghua.edu.cn/>) is demonstrated as a showcase. Unlike other open datasets, our datasets are mostly built upon data crawled from the public university website. We currently have 5 core datasets, with downloadable RDF/XML descriptions and a standard SPARQL endpoint. There is also a HTML web interface available for SPARQL query and data browsing. We have also built some applications by ourselves, including CampusAssistant, CourseFinder, etc.

Processes and techniques described in this paper serves well in building Tsinghua University Open Data, and we believe that it is helpful when building open data for other institutions, and for publishing general linked data as well.

The rest of this paper is organized as follows: In Section 2, we discuss common categories of university data, and the vocabularies used to describe these data. In Section 3, we discuss different approaches of getting raw data, and necessary procedures to clean and normalize these data, as well as converting the processed data to RDF. A detailed description of Tsinghua University Open Data is given in Section 4. Then we give a brief introduction about related work in Section 5. Finally, we discuss key points and difficulties in building linked open university data in Section 6 and conclude the paper in Section 7.

2 Data Categories and Schema

2.1 Data Categories

Modern universities serve as educational institutions, academic research institutions, as well as living communities of their students and staff members. As a consequence, there are so many aspects of university data. However, several university datasets attract most attentions of people and act as core data connecting different aspects of university information, thus are of great importance in building linked open university data.

We categorize the most important university information into the following classes:

University Basic Information. General properties about the institution. These mainly consist of founding year, motto, location, etc. Basic information

also includes organization structure of the institution. Organizations - schools and departments - description has organization name, size, superior organization information, location of department building, etc.

Campus Information. Campus information contains geographic description of the campus, as well as building information within and around the campus.

Educational Administrative Information. As facility for education, educational administrative information is important to not only students within the university, but also to researchers and students outside campus. Courses, exam schedule are examples of datasets of this category.

Faculty Information. Describes staff members of the university, mostly teachers and researchers, their basic information, contact information, research related information, etc.

2.2 Vocabularies

On the Semantic Web, *vocabularies* (or *ontologies*, more strictly), define the concepts and relationships (also referred to as “terms”) used to describe and represent an area of concern. Vocabularies are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms³. A standard set of vocabularies not only provides unified access to data consumers, but also acts as an important role in data mashup and inference.

There is a number of vocabularies that can be used in describing university information. We have summarized a set of vocabularies for university datasets, mainly considering popularity, comprehensiveness and quality. Most of the vocabularies we use are strictly defined and public accessible.

FOAF ⁴ (Friend Of A Friend) is a universally acknowledged vocabulary devoted to describe information about people and their relationships. FOAF is so widely used that it appears in nearly every dataset of our site.

AIISO ⁵ The Academic Institution Internal Structure Ontology (AIISO) is designed to describe the internal organizational structure of an academic institution. In our datasets, we use *aiiso:Course* to describe university course information, with is used along with FOAF and Org, and newly defined extended properties to link courses with organizations and staff members.

Org ⁶ is a widely used vocabulary devoted to describe information about organization and their relationships.ORG is the main vocabulary used in the *Organization* dataset of Tsinghua University Open Data, with each work unit as a *org:Organization*. It describes basic relationship informations such as *subOrganizationOf*, *hasSubOrganization*, *hasUnit* etc.

³ <http://www.w3.org/standards/semanticweb/ontology>

⁴ <http://xmlns.com/foaf/spec/>

⁵ <http://vocab.org/aiiso/schema>

⁶ <http://www.w3.org/ns/org>

*CourseWare*⁷ is developed for describing courses and resources within the *ReSIST*⁸ project. This vocabulary represents various information about a course, including material, pre-requirement, language, etc.

As for those datasets which there is no suitable existing vocabulary defined, we have defined new terms, and linked our own terms to the existing vocabularies as supplements. The main vocabularies used are listed as follows.

We use OpenVocab to create our own terms. *OpenVocab*⁹ is a community maintained vocabulary intended for use on the Semantic Web, ideal for properties and classes that do not warrant the effort of creating or maintaining a full schema. OpenVocab allows anyone to create and modify vocabulary terms using their web browser. We defined several properties to construct interlink between different datasets. For example, `ov:deliveredBy` is used for linking courses with lecturers; `ov:offeredBy` is used for linking courses with organizations.

3 Data Collection and Structuring

3.1 Data Collection

Considering the variety among different universities, different approaches may apply to collect raw structured data for building linked open university data. We list the most commonly used methods as follows:

Some raw data can be retrieved from the university administration facility, which are allowed to be published under certain license. These data are usually well structured, high quality with few or no noise. However, this approach need firm cooperation between data publisher and university authority, which is sometimes difficult to achieve. What's more, many kinds of data do not have structured backends.

The university publishes some of its information on the web, either as downloadable structured format or HTTP queryable. We can download these data and organize them as our data sources. These data sources need slightly more preprocessing before data converting and interlinking. The organization and course schedule data sources are obtained by this approach.

Other data could come from webpage crawling, and this is often the major approach of getting raw data. We use information extraction technology to extract structured data out of description pages on the web, e.g. organizations' main page and people's homepage. Webpages are crawled following website structure of the university, we then analyze the pages and extract semantic properties to build structured data source. These data sources are usually of inferior quality, with a lot of noises and mistakes. After necessary alignment, they can be matched and linked with other data sources.

⁷ <http://courseware.rkbexplorer.com/ontologies/courseware>

⁸ <http://www.resist-noe.org/>

⁹ <http://open.vocab.org/>

3.2 Data Converting and Mashup

After collection and preprocessing, we have got structured tabular data in different formats, such as SQL database, MS Excel .xls file and plain text file. These data are then marked with URI using our naming strategy.

Different data sources are then linked together using usual mapping technique. For example, in the course data source, lecturer is stored using lecturer's name. Then the lecture is mapped to a instance entry in the staff member data source who has the same name.

Finally, these data sources are converted to RDF format using RDF generating tools like *D2R Server*^[10] and *ConvertToRDF*^[11].

4 Tsinghua University Open Data

4.1 Datasets

Tsinghua University Open Data currently has 5 core datasets, describing the basic information and structure of the institution.

Campus Buildings and Places. This dataset describes buildings, sightseings and other geospatial entities related to Tsinghua University. Properties include name, description, geographic location, type, image, etc. This dataset currently has 784 triples, with 114 instances described.

Organizations. This dataset contains description for the 87 organizations above department/school level. Organization name, homepage type and structure are described. This dataset currently has 372 triples.

Staffs. We crawled all the staff members listed in all organizations' reference pages using automatic extraction along with manual selection, totally 2758 instances. Considering data deficiency and error, this may not be the actual count of all Tsinghua staff members. Name, gender, title and contact information of the staff members are described with 21631 triples.

Campus Photographs. This dataset contains additional photographs about the campus. Each photograph is listed with its URL and the place it depicts. There are currently 316 photographs described with 977 triples.

Courses Schedule. We extract the public course information from the university's website and form this dataset. We have done 30 recent semesters, with course name, lecturer, delivering department and open type described. Each semester has less than 2000 courses and about 10,000 triples.

Relationships of these five dataset are illustrated in Fig. 1

4.2 System Infrastructure and User Interface

We have built a web portal (<http://data.cs.tsinghua.edu.cn>) for publishing our datasets using JSP and JavaScript. There is a dataset catalog page available

¹⁰ <http://www4.wiwiw.fu-berlin.de/bizer/d2r-server/>

¹¹ <http://www.mindswap.org/~mhgrove/convert/>

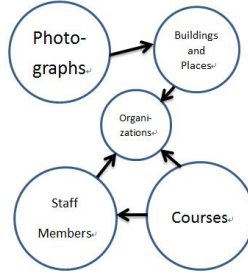


Fig. 1. Tsinghua University Open Data Cloud

for browsing and downloading datasets. We have also set up a standard SPARQL endpoint, with a HTML front end for query and browsing. We have also made a customized web page for every class, corresponding to HTTP request of an instance URI. Thus every URI can be dereferenced, as is required by Linked Data principle.

4.3 Applications

We aim at providing a unified open data platform about Tsinghua University and other academic institutes, and we hope that researchers and programmers can make full use of the datasets and build interesting and useful applications upon them. Meanwhile, we have built several interesting applications by ourselves. These applications all use the SPARQL endpoint to access data, and most of them use two or more datasets.

CampusAssistant^[12] provides searchable map for finding buildings, navigation, memo and paths in Tsinghua University. This application uses the *Buildings and Places dataset*, as well as the *Photographs dataset*.

CourseFinder^[13] provides various ways for searching course information. A course can be found by searching its name, the lecturer's name, the facility's name, or combination of these criteria. What's more, one can easily navigate between courses, lecturers and departments; view detailed information about the lecturer, e.g. contact information(if public); and find relevant courses and teachers. All these need no extra work and is automatically achieved using interlinks between datasets.

5 Related Work

There are numbers of projects related to building and linking open university data. *Waterloo Open Data Initiative*^[14] offers information about campus buildings, classes schedule and exam information about University of Waterloo. Their

¹² <http://iweb.cs.tsinghua.edu.cn/CampusAssistant>

¹³ <http://data.cs.tsinghua.edu.cn/OpenData/courses.jsp>

¹⁴ <https://opendata.uwaterloo.ca/drupal/>

data are provided with standard structured format like JSON and CSV file, but not in Linked Data format.

*University of Southampton Open Data*¹⁵ provides open access to some of the university's administrative data. Currently, they provide 29 datasets, covering from campus information to public phonebook of teachers and students, with RDF file and SPARQL endpoint provided. They have also built several applications, mostly school maps and phonebooks. They provided relatively wide-ranging datasets, but some of the datasets are not tidily bounded to the university, and some basic datasets are missing, e.g. course information and school affairs statistics.

Several other open university data portals are also on the go, *OU Linked Data*¹⁶ currently has 6 datasets, but without university organizations and staff members description; *Open Data about the University of Oxford*¹⁷ is under construction and not yet public. A incomplete list of European universities that have published open data can be found at *Linked Universities*¹⁸.

6 Discussion

In this section, we will talk about some key points and difficulties we have met throughout the procedure of building open data.

First of all, “Useful” is the first principle when publishing open data, thus making data quality a great concern. We notice that several aspects of data processing act as an important role for better data quality.

For data retrieved from webpage crawling, careful strategy for handling incomplete data records is needed. Some missing fields of a record just result in missing properties of the generated instance; however, missing of some key values may leave the instance pointless and omitted. We have developed an automatic filtering system when generating RDF data from the data sources.

Problems concerning data privacy and copyright also cannot be ignored when publishing open data, especially data involving personal information. We must be sure that our collected data is either public or properly licensed; and when processing data, we do data filtering (filter bad and false data) and add non-sensitive information to data (by adding links and meta data), but we never do any modification or correction to existing data. A formal disclaimer is provided along with the data as well.

7 Conclusion and Future Work

Linked Open University Data applies Linked Data to publishing information about universities and academic institutions. In this paper, we discuss university

¹⁵ <http://data.southampton.ac.uk/>

¹⁶ <http://data.open.ac.uk/>

¹⁷ <http://data.ox.ac.uk/>

¹⁸ <http://linkeduniversities.org/>

data category, vocabularies, data collection and cleaning, as well as method of build and mashup linked data about university. Tsinghua University Open Data is presented as a showcase. We describe its datasets and applications upon them. These applications, however, is also portable for any other datasets with the same or similar schema.

As future work, we will first improve our datasets. This include work in several aspects. Firstly, we will continue to find other data sources to add more dataset. Secondly, we plan to do deeper work at the data preprocessing and mapping phase, like doing name disambiguation.

Another important future work is to connect our datasets to the various existing Semantic Web data. With more raw data available, we can connect our datasets with academic publication datasets, geographic datasets, etc. This will further improve the power of our datasets and open new possibilities to applications.

We hope that more academic institutions can join the work of publishing open data, to build increasing number of datasets publish from different institutions using unified schema. Applications can then make full use of the data, creating useful and interesting tools and amazing user experience, which in turn inspire more institutions providing open data.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284(5), 34–43
2. Tim Berners-Lee: Design Issues: Linked Data (2009), <http://www.w3.org/DesignIssues/LinkedData.html>
3. Bizer, C., Cyganiak, R.: Tom Heath: How to Publish Linked Data on the Web, Tutorial (2007), <http://www4.wiwi.fu-berlin.de/bizer/pub/LinkedDataTutorial/>
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, C., 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. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)* 2009
6. Auer, S., et al.: Triplify - Light-Weight Linked Data Publication from Relational Databases. In: *Proceedings of the 18th World Wide Web Conference, WWW 2009* (2009)
7. Bizer, C., Cyganiak, R.: D2R Server - Publishing Relational Databases on the Semantic Web. Poster at the 5th International Semantic Web Conference, ISWC 2006 (2006)
8. Chen, H., Wang, Y., Wang, H., Mao, Y., Tang, J., Zhou, C., Yin, A., Wu, Z.: Towards a Semantic Web of Relational Databases: A Practical Semantic Toolkit and an in-use Case from Traditional Chinese Medicine. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006*. LNCS, vol. 4273, pp. 750–763. Springer, Heidelberg (2006)

9. Auer, S., Lehmann, J.: What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In: Franconi, E., Kifer, M., May, W. (eds.) *ESWC 2007*. LNCS, vol. 4519, pp. 503–517. Springer, Heidelberg (2007)
10. Metaweb Technologies, Freebase Wikipedia extraction (wex) (2009), <http://download.freebase.com/wex/>
11. Li, X., Bao, J., Hendler, J.A.: Fundamental analysis powered by semantic web. In: *Proceedings of IEEE Symposium on Computational Intelligence for Financial Engineering and Economics* (2011)

An Abductive CQA Based Matchmaking System for Finding Renting Houses

Jianfeng Du^{1,2}, Shuai Wang¹, Guilin Qi³, Jeff Z. Pan⁴, and Che Qiu¹

¹ Guangdong University of Foreign Studies, Guangzhou 510006, China
jfd@mail.gdufs.edu.cn

² State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China

³ Southeast University, Nanjing 211189, China

⁴ The University of Aberdeen, Aberdeen AB243UE, UK

Abstract. A matchmaking system for finding renting houses is required as the housing problem becomes serious in China and many people resort to rent a house. A semantic approach based on abductive conjunctive query answering (CQA) in Description Logic ontologies is exploited to provide more matches for a request about renting houses. Moreover, a matchmaking system based on this approach is developed. This demo will guide users to find suitable renting houses using this matchmaking system and show the advantages of the system.

1 Motivation

The housing problem is an important social problem in China due to the large scale of population. This problem becomes more serious recently. According to some survey, the housing price-to-income ratio in China is very high and such a high ratio implies that about 85 percent of the families cannot afford a house in cities [7]. Under this situation, many people resort to rent a house and use matchmaking systems to find suitable renting houses.

Keyword search based matchmaking systems for finding renting houses are backing up the current rental search engines which are available at the World Wide Web. Although these matchmaking systems are rather efficient, they are hard to provide sound and complete matches. On the one hand, keyword search based matchmaking makes little use of the background knowledge and will easily miss real matches. For example, HEMC (Higher Education Mega Center) locates in Panyu (a district in Guangzhou). When a request for renting houses in Panyu is posed, a matchmaking system based on keyword search will not output any offer which describes that the renting house is in HEMC, because it does not involve reasoning by using the relationship between HEMC and Panyu. On the other hand, keyword search based matchmaking makes little use of the semantics of offers or requests and will easily output wrong matches. For example, a matchmaking system based on keyword search does not manage the meaning of the word “top floor”. When a request for renting houses at the top floor is posed,

the system will output offers describing that the renting houses are at any floor, because the word “floor” appears in the offer descriptions.

To make the matchmaking results more sound and complete, ontology-based matchmaking systems have been proposed. There are two approaches to ontology-based matchmaking. One approach exploits a Description Logic (DL) [1] ontology to compute semantic distances between offers and requests, where offers and requests are expressed as DL concept descriptions. This approach is followed by many methods such as those proposed in [2,8,16]. They mainly focus on defining a reasonable distance function between two DL concept descriptions. The other approach exploits DL inference methods to compute different kinds of matches. Concept subsumption checking is the most popular one among such methods [12,15]. Other known DL inference methods include concept abduction and concept contraction [14]. They are used to compute possible matches in a negotiation framework. That is, an offer is regarded as a possible match for a request if it can get subsumed by the request after *abduction*, i.e. adding some information to itself, and *contraction*, i.e. removing some information from the request.

However, the two main ontology-based matchmaking approaches are not easy to scale to real-life applications that involve a large number of offers and requests, because composing the DL concept descriptions for offers and requests is time consuming and laborious. To alleviate human efforts to formalize offers or requests, our initial solution is to treat the matchmaking problem as the conjunctive query answering (CQA) problem. In this solution, offer information is expressed as individual assertions in the back-end ontology, request information is expressed as conjunctive queries, and a match for a request is defined as an answer to a conjunctive query that expresses the request. This solution allows offer information to be automatically extracted from text sources, such as Web pages, using off-the-shelf ontology population techniques [3], thus significantly reducing human efforts. However, this solution is still hard to provide complete matches due to incompleteness of information extracted from the World Wide Web. When some information about an offer is missing in the back-end ontology, the offer will not be an answer to a conjunctive query that expresses the given request, but it will turn to be after missing information is added, so this offer can also be considered as a match for the request.

To realize the above idea, in [6] we introduced the *abductive CQA* problem which computes all *abductive answers* to a conjunctive query in a consistent ontology. An abductive answer is an answer to the given query in some consistent ontology enlarged from the given one by adding a bounded number of individual assertions, where the individual assertions that can be added are confined by user-specified concept or role names. We developed an abductive CQA based matchmaking system for finding renting houses, where a match for a request is defined as an abductive answer to a conjunctive query that expresses the request.

In the next two sections, we highlight the novelty of our matchmaking system and give preliminaries about this demo, respectively. Then, before the last section in which we explain what will be shown in this demo, we describe the architecture of the matchmaking system.

2 Novelty

The matchmaking system provides a DL based framework for integrating information from different Web sources and handling user requests for renting houses. It has some advantages. One is that the system can accept a complex request considering nearby traffic lines and public facilities. These requests for renting houses cannot be handled by current rental search engines. Another advantage is that all output information for responding to a request is interpretable by existing DL reasoning facilities.

3 Preliminaries

3.1 OWL 2 and Conjunctive Query Answering

The World Wide Web Consortium (W3C) has proposed the Web Ontology Language (OWL), for which the newest version is OWL 2 [9], to model ontologies. OWL is based on DLs [1]. In particular, the most expressive and decidable species of OWL 2, OWL 2 DL, corresponds to the DL \mathcal{SROIQ} [11]. An OWL 2 DL ontology consists of an RBox, a TBox and an ABox. The RBox consists of a finite set of complex role inclusion axioms and role assertions declaring that a role is symmetric, transitive, reflexive, irreflexive, or disjoint with another role. The TBox consists of a finite set of concept inclusion axioms. The ABox consists of a finite set of individual assertions that declare memberships of concepts or roles, or equivalence relations between individuals. Since \mathcal{SROIQ} is a fragment of First-order Logic, its semantics can be defined by translating to First-order Logic. For example, the following two axioms expressed in \mathcal{SROIQ} , namely $\text{hasFacility} \circ \text{isA} \sqsubseteq \text{hasFacility}$ and $\text{House} \sqsubseteq \text{Building}$, can be translated to two First-order rules given below and inherit the standard First-order semantics, where the former rule tells that if x has a facility y and y is more specific than z , then x also has a facility z , while the latter one tells that if x is a renting house, then it is also a building.

$$\begin{aligned} \forall x, y, z : \text{hasFacility}(x, y) \wedge \text{isA}(y, z) &\rightarrow \text{hasFacility}(x, z) \\ \forall x : \text{House}(x) &\rightarrow \text{Building}(x) \end{aligned}$$

A *model* of an OWL 2 DL ontology is an interpretation on all entities in the ontology that satisfies all First-order rules translated from the ontology under the standard First-order semantics. An OWL 2 DL ontology is said to be *consistent* if it admits at least one model.

A *conjunctive query* is an expression of the form $\exists \vec{y} : \text{conj}(\vec{x}, \vec{y}, \vec{c})$, where \vec{x} and \vec{y} are both vectors of variables, and \vec{c} is a vector of individuals or constants. $\text{conj}(\vec{x}, \vec{y}, \vec{c})$ denotes a conjunction of *atoms* of the form $A(v)$ or $r(v_1, v_2)$, where A is an *atomic concept* (i.e. a concept name), r is an *atomic role* (i.e. a role name) or a built-in predicate, and v, v_1 and v_2 are variables in \vec{x} and \vec{y} , or individuals or constants in \vec{c} . A *Boolean conjunctive query* is a conjunctive query without distinguished variables.

Given an OWL 2 DL ontology \mathcal{O} and a Boolean conjunctive query $Q = \exists \vec{y} : \text{conj}(\vec{y}, \vec{c})$, a model \mathcal{I} of \mathcal{O} is said to *satisfy* Q if there exists a tuple of (possibly

anonymous) individuals or constants whose substitution for the variables in \vec{y} makes every atom in $\text{conj}(\vec{y}, \vec{c})$ satisfied by \mathcal{I} . Q is said to be *entailed* by \mathcal{O} , denoted by $\mathcal{O} \models Q$, if every model of \mathcal{O} satisfies Q . A tuple \vec{t} of individuals is called an *answer* to a conjunctive query $Q(\vec{x}) = \exists \vec{y} : \text{conj}(\vec{x}, \vec{y}, \vec{c})$ in \mathcal{O} if $\mathcal{O} \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, where $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ denotes a Boolean conjunctive query obtained from $Q(\vec{x})$ by replacing every variable in \vec{x} with its corresponding individual in \vec{t} . The *conjunctive query answering (CQA)* problem is to compute all answers to a conjunctive query in an ontology.

3.2 Abductive Conjunctive Query Answering

As mentioned before, the matchmaking problem can be treated as the CQA problem, where offer information and background knowledge are stored in a back-end ontology. For example, when we want to find all southward renting houses in Guangzhou, we can pose the following conjunctive query upon the back-end ontology:

$$\text{House}(x) \wedge \text{locatesIn}(x, \text{Guangzhou}) \wedge \text{towards}(x, \text{South}).$$

Then the answers to this query, namely the individuals substituting for the variable x , are renting houses to be found. However, these answers may not provide all choices to a requester. For example, when the orientation of a renting house is missing, possibly due to incomplete extraction from Web pages, this renting house will not be an answer to the aforementioned query, although its orientation is south in reality. To compensate these answers, a certain enlarged ontology should be considered. This ontology can be seen as the result of adding missing information about offers to the back-end ontology.

Hence, in [6] we introduced a new kind of answers, called *abductive answers*, to a conjunctive query. Abductive answers are formally defined as follows. Given a consistent ontology \mathcal{O} , a conjunctive query Q , a non-negative integer k , two disjoint sets of concept or role names S_A and S_C , an *abductive answer* \vec{t} to Q in \mathcal{O} w.r.t. k , S_A and S_C is an answer to Q in $\mathcal{O} \cup \mathcal{A}$ for some set \mathcal{A} of individual assertions such that the cardinality of \mathcal{A} is not greater than k , all individual assertions in \mathcal{A} are on abducible predicates, and any individual assertion on closed predicates that is entailed by $\mathcal{O} \cup \mathcal{A}$ is also entailed by \mathcal{O} , where \mathcal{A} is said to be *attached with* \vec{t} and the concept or role names in S_A (resp. S_C) are called *abducible predicates* (resp. *closed predicates*). In this definition, \mathcal{A} can be seen as the missing information about a certain offer. The definition says that \mathcal{A} should consist of at most k individual assertions, where k is a user-specified parameter which reflects the incompleteness of the given ontology \mathcal{O} . It also says that all concepts/roles appearing in \mathcal{A} should be abducible predicates, and appending \mathcal{A} to \mathcal{O} should not make \mathcal{O} entail any individual assertion α on closed predicates unless α is already entailed by the original \mathcal{O} . We call the problem of computing all abductive answers to a conjunctive query in a consistent ontology the *abductive CQA* problem.

The abductive CQA problem is similar to the ABox abduction problem proposed in [4] which, for a consistent ontology \mathcal{O} and a set G of individual assertions, computes all minimal sets \mathcal{A} of individual assertions on a set S of

predicates such that $\mathcal{O} \cup \mathcal{A}$ is consistent, and $\mathcal{O} \cup \mathcal{A}$ entails all individual assertions in G but \mathcal{A} does not. Compared to the ABox abduction problem, the abductive CQA problem also restricts \mathcal{A} to a set of individual assertions on abducible predicates such that appending it to \mathcal{O} does not introduce inconsistency, where abducible predicates are used to define which kind of information is incomplete. However, instead of computing certain \mathcal{A} , the abductive CQA problem computes abductive answers to a conjunctive query by considering all possible \mathcal{A} . Moreover, it introduces the use of the parameter k and a set of closed predicates. The parameter k is used to control the extensiveness of abducible answers, while closed predicates are used to simulate disjoint concept or role axioms (see [6] for the explanations on this usage) and enable some optimizations in computing abductive answers. These optimizations are exploited in a method for computing abductive answers [6]. Basically, the method encodes the abductive CQA problem into a Prolog program and solves it with Prolog engines. It computes exactly all abductive answers in a consistent *DLP* ontology, which is an ontology expressed in the *DLP* fragment [10] of OWL 2 DL and adopts the Unique Name Assumption [1] that any two different individual names must correspond to different elements in the interpretation domain.

4 The Architecture of the Matchmaking System

We developed an abductive CQA based matchmaking system for finding renting houses, where a match for a request for renting houses is defined as an abductive answer to a conjunctive query that expresses the request. The system consists of two parts, the offline part and the online part, as shown in Fig. 1. The offline part targets integrating rental related information from different Websites into a consistent *DLP* ontology, and consists of several components, including a controller, a triple converter, a consistency restorer and multiple wrappers. The online part targets responding to user requests about renting houses, and is built on a matchmaker which implements the method for abductive CQA [6]. More details for all these components are given below.

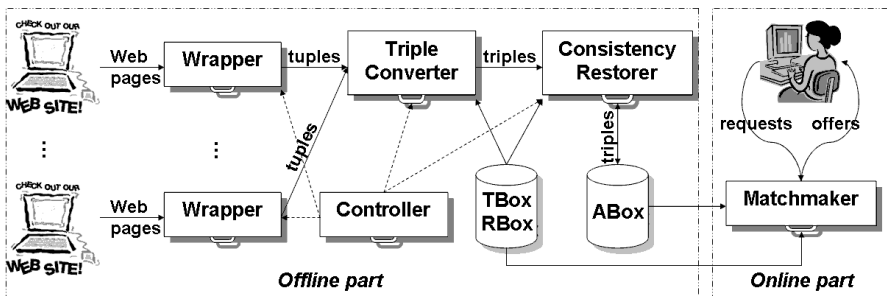


Fig. 1. The interactions among components of the matchmaking system

4.1 Wrapper

A wrapper is a component for extracting tuples from a Website. It first downloads Web pages from a Website, then extracts tuples from these Web pages. We developed different wrappers for different Websites, including those that provide rental search services and those publishing information on traffic lines and administrative regions. The extraction process relies on a set of extraction rules, some of which were manually written and some were automatically learned by machine learning methods [13].

4.2 Triple Converter

The triple converter is a component for converting the extracted tuples to individual assertions, which are stored as RDF triples in the ABox of the back-end ontology. In this converter, the TBox and the RBox of the back-end ontology, which were manually constructed using ontology editors, confine the concept or role names that can appear in a generated individual assertion.

Moreover, some methods for relation extraction were developed to add links between two individuals from different Websites. For example, the tuples extracted from a rental Website have a free-text field mentioning traffic lines and stops that a renting house locates near. We developed a pattern based method for extracting role assertions of the form `locatesNear(a, b)` from this free-text field, where a is a renting house and b is either a traffic line or a stop. The method first generates a pattern set based on the words about traffic lines or stops that are extracted from a traffic Website, then uses this pattern set to identify entities about traffic lines or stops in the free-text field and compose role assertions of the form `locatesNear(a, b)`. The individual b in a generated role assertion `locatesNear(a, b)` may not have the same name as the truly same individual extracted from other Websites. We used the same pattern set to determine whether two individuals with different names are the same in reality. That is, we defined that two individuals are the truly same if the core parts of their names are the same, where the core part of a name is computed by some manually written rules based on matched patterns of the name. On the other hand, two individuals with the same name but extracted from different Websites may not be the same in reality. We defined that such two individuals are the truly same if the concepts they belong to are not disjoint in the back-end ontology. To avoid introducing equality assertions, the triple converter only mentions a representative for all truly same individuals in the back-end ontology.

4.3 Consistency Restorer

The consistency restorer is a component for rendering the back-end ontology consistent when it is inconsistent. The method proposed in [5] is applied to compute a cost-minimal set of individual assertions that should be removed to restore consistency, where the cost of an individual assertion is determined by the importance of the concept or role name that the assertion is on. This kind of importance information is set by the administrator of the system.

4.4 Controller

The controller is a component for scheduling the execution of every wrapper, the triple converter and the consistency restorer. It periodically invokes all wrappers to extract tuples. After the tuple extraction process finishes, it invokes the triple converter to generate individual assertions and add them to the ABox. After the triple generation process finishes, it invokes the consistency restorer to render the evolved back-end ontology consistent.

4.5 Matchmaker

The matchmaker is the kernel component in the online part. To start this component, the administrator of the system should set the parameters k , S_A and S_C for computing abductive answers, and then trigger the component to perform preprocessing, i.e. to encode the back-end ontology and the parameters k , S_A and S_C to a Prolog program and load it into a Prolog engine. Afterwards, this component waits for user requests for renting houses. Once it receives a user request, it first translates it to a conjunctive query, then combines the conjunctive query with the loaded Prolog program and computes all abductive answers from the combination. These abductive answers correspond to renting houses that are offered. The abductive answers that can attach with (cardinality) smaller sets of individual assertions are output earlier. To show why a renting house is an abductive answer, a cardinality-minimal set of individual assertions attached with it is also displayed.

5 What Will Be Demonstrated?

This demo will guide users to find suitable renting houses in China using the proposed matchmaking system. In more details, we will show how to compose a request for renting houses in this system and how to analyze the output information to pick up a suitable renting house. The demo will also show some advantages of the system, including that it can handle complex requests considering nearby traffic lines and public facilities and that all output information is interpretable by existing DL reasoning facilities.

Acknowledgement. Jianfeng Du and Shuai Wang are partly supported by the NSFC under grant 61005043 and the Undergraduate Innovative Experiment Project in Guangdong University of Foreign Studies. Guilin Qi is partly supported by Excellent Youth Scholars Program of Southeast University under grant 4009001011, the NSFC under grant 61003157, Jiangsu Science Foundation under grant BK2010412, and the Key Laboratory of Computer Network and Information Integration (Southeast University). Jeff Z. Pan is partly supported by the EU K-Drive project and the RCUK dot.rural project.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Bianchini, D., Antonellis, V.D., Melchiori, M.: Flexible semantic-based service matchmaking and discovery. *World Wide Web* 11(2), 227–251 (2008)
3. Cimiano, P., Völker, J.: Text2onto - A Framework for Ontology Learning and Data-Driven Change Discovery. In: Montoyo, A., Muñoz, R., Métais, E. (eds.) *NLDB 2005*. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
4. Du, J., Qi, G., Shen, Y., Pan, J.Z.: Towards practical abox abduction in large OWL DL ontologies. In: Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI), pp. 1160–1165 (2011)
5. Du, J., Shen, Y.: Computing minimum cost diagnoses to repair populated DL-based ontologies. In: Proc. of the 17th International World Wide Web Conference (WWW), pp. 265–274 (2008)
6. Du, J., Wang, S., Qi, G., Pan, J.Z., Hu, Y.: A New Matchmaking Approach Based on Abductive Conjunctive Query Answering. In: et al. (eds.) *JIST 2011*. LNCS, vol. 7185, pp. 144–159. Springer, Heidelberg (2012)
7. Feng, W., Wu, N.: On the capital production of space and china's housing problem. *Journal of Wuling* 35(6), 55–59 (2010)
8. Fenza, G., Loia, V., Senatore, S.: A hybrid approach to semantic web services matchmaking. *International Journal of Approximate Reasoning* 48(3), 808–828 (2008)
9. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. *Journal of Web Semantics* 6(4), 309–322 (2008)
10. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. of the 12th International World Wide Web Conference (WWW), pp. 48–57 (2003)
11. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SRIOQ*. In: Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 57–67 (2006)
12. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. In: Proc. of the 12th International World Wide Web Conference (WWW), pp. 331–339 (2003)
13. Liu, B.: *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Data-Centric Systems and Applications. Springer, Heidelberg (2007)
14. Noia, T.D., Sciascio, E.D., Donini, F.M.: Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research* 29, 269–307 (2007)
15. Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: A system for principled matchmaking in an electronic marketplace. In: Proc. of the 12th International World Wide Web Conference (WWW), pp. 321–330 (2003)
16. Shu, G., Rana, O.F., Avis, N.J., Chen, D.: Ontology-based semantic matchmaking approach. *Advances in Engineering Software* 38(1), 59–67 (2007)

An Ontological Approach to Oracle BPM

Jean Prater, Ralf Mueller, and Bill Beauregard

Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065, USA
{jean.prater,ralf.mueller,william.beauregard}@oracle.com

Abstract. A modern business process management (BPM) operates using common tenants of an underlying Service Oriented Architecture (SOA) runtime infrastructure based on the Service Component Architecture (SCA) and supports the BPMN 2.0 OMG¹ standard. Semantically-enabling all BPM artifacts, from high-level design to deployment and the runtime model of a BPM application, promotes continuous process refinement, comprehensive impact analysis, and reuse to minimize process and service proliferation. A semantic database can manage semantically-enabled BPM ontologies and models, enable machine-driven inference to discover implicit relationships in the models, and perform pattern-matching queries to find associations.

This paper presents an ontology for BPM based upon BPMN 2.0, Service Component Architecture (SCA) and the Web Ontology Language (OWL 2) that can support a wide range of use cases for process analysis, governance, business intelligence and systems management. It has the potential to bring together stakeholders across an enterprise, for a truly agile, end-to-end enterprise architecture.

Keywords: BPM Ontology, OWL 2, BPMN 2.0, SOA, SCA, SPARQL, Semantic Technologies.

1 Introduction

Semantic technology can be used by business process management (BPM) to define contextual relationships between business processes. This allows ‘software agents’ (programs working on behalf of people) to automatically find the right information or processes and enables machine-driven decision-making based on the established contextual relationships.

Organizations can optimize their information technology resources through a Service Oriented Architecture (SOA) approach that embraces common business processes and semantics throughout the enterprise. The challenge, however, with applications built on BPM and SOA is the number of representation formats used to define artifacts, such as the Service Component Architecture Assembly Model, Web Service WSDL definitions, and XSLT transformations. This makes even simple queries about the entire BPM model difficult and complicated.

¹ Object Management Group, see <http://www.omg.org>

This heterogeneous modeling problem can be addressed with an ontology based upon the W3C OWL standard [7] that encompasses all artifacts of a BPM application. The ontology and associated models are stored in a semantically-enabled database. Support for W3C SPARQL query language allows pattern matching queries across the enterprise-wide model to find associations. Native inferencing in the database allows machine-driven discovery of new, previously undefined relationships within the model.

Business users can find, share, and combine information and processes across organizational boundaries more easily with the addition of contextual relationships. The combination of business process management and semantic technology driven by an underlying ontology makes it possible to:

- Evolve a business process model into a complete executable process in the same model.
- Analyze the impact on existing processes and web services of adding, modifying or deleting processes and process building blocks.
- Minimize proliferation of processes and services.

Oracle is actively involved in the standards process and is leading industry efforts to use ontologies for metadata analysis. Oracle is also investigating the integration of organizational and social aspects of BPM using the ontology FOAF². BPMN 2.0 task performers can be associated with a FOAF Person, Group or Organization and then used in Social Web activities to enable business users to collaborate on BPM models.

2 Customer Use Case

The US Department of Defense (DoD), Business Transformation Agency (BTA), under the purview of the Deputy Chief Management Officer (DCMO) is on a mission to achieve an Architecture-driven Business Operations Transformation. A key principle in the DoD Business Transformation is developing a common vocabulary and standard processes in support of business enterprise interoperability. The use of primitives and reuse of process patterns will reduce overhead costs, process duplication and resources needed for building and maintaining enterprise architectures. By aligning the Department of Defense Architecture Framework³ 2.0 (DoDAF 2.0) with Business Process Modeling Notation 2.0 (BPMN 2.0) and partnering with industry, the BTA is accelerating the adoption of these standards to improve government business process efficiency.

A vital tenet for success is ensuring that business process models are based on a standardized, semantically-enabled representation. This will enable analysis and comparison of end-to-end business processes and reuse of the most efficient and effective process patterns (style guide) and elements (primitives), throughout the DoD Business Mission Area.

² The Friend of a Friend (FOAF) project, see <http://www.foaf-project.org>

³ See [http://www.bta.mil/products/BEA_6.2/BEA/products/2009-04-27/Primitives Guidelines for Business Process Models \(DoDAF OV-6c\).pdf](http://www.bta.mil/products/BEA_6.2/BEA/products/2009-04-27/Primitives%20Guidelines%20for%20Business%20Process%20Models%20(DoDAF%20OV-6c).pdf)

The DoD implementation used Oracle BPM and the ontology for BPM discussed in this paper to generate RDF triples from the artifacts of the BPM Project. These artifacts included BPMN 2.0 process definitions, SCA assembly model, WSDL service definitions, XML-Schema and other metadata. The triples were managed in Oracle Database using the Semantic Technologies feature of Oracle Spatial [3] and a SPARQL endpoint was used to query the model.

3 An Ontology for BPM

An ontology for BPM encompasses and expands the BPMN 2.0 Ontology and Service Component Architecture (SCA) ontology. This ontology creates a composite model by establishing relationships between the OWL classes of the BPMN 2.0 ontology and the OWL classes of the SCA runtime ontology. The following diagram illustrates the layering of the BPM ontology on top of the BPMN 2.0 and SCA ontology and their relationship to other representation formats.

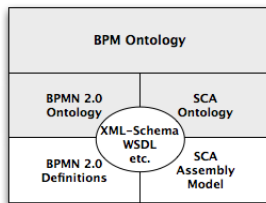


Fig. 1. The BPM ontology stack

Generally speaking, the ontology for BPM relates BPMN 2.0 tasks and events to corresponding SCA composite model entities like components, services and references. For instance, BPMN 2.0 Process, User and Business Rule tasks are related to components in the SCA composite model and BPMN 2.0 Send, Receive and Service tasks and Message Events are related to SCA Services and References. The BPM ontology creates appropriate relationships between these composite model artifacts. The relationships include the OWL representation of SCA wires. Figure 2 provides an example that illustrates the anatomy of a Business Rule Task “Determine Approval Flow”.

One can see in Figure 2 how the BPM ontology relates BPMN 2.0 tasks to corresponding SCA composite model, services and references. At the top of the figure the Business Rule Task “DetermineApprovalFlow” is of type BPMN 2.0 “BusinessRuleTask”. It is implemented by a SCA component “ApprovalRules” that is of SCA type “DecisionComponent” and related to another SCA Component “RequestQuote” through a SCA Wire.

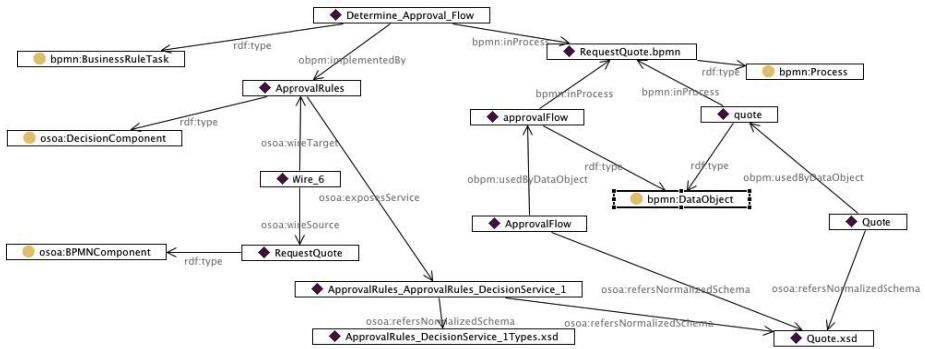


Fig. 2. Anatomy of a BPMN 2.0 Business Rule Task⁴

It is important to note that the SCA “DecisionComponent” in the middle left side of the figure exposes the Service in the bottom of the figure, “...AprovalRules_DecisionService_1” that in turn refers to the XML-Schema, “Quote.XSD” in the bottom right side of the figure. “Quote.XSD” is also referred to by data objects “approvalFlow” and “quote” in the BPMN 2.0 process “RequestQuote.bpmn” in the upper right side of the figure. This illustrates how the semantic relationships defined in the BPM ontology make it possible to perform a comprehensive impact analysis for process data objects, XML schema definitions and service contracts across the entire model.

4 An Ontology for BPMN 2.0

The OMG BPMN 2.0 standard specifies a serialization format for the semantic model of BPMN 2.0 processes in XMI and XML-Schema. The ontology for BPMN 2.0 is based on this model. Oracle BPM includes an implementation that comprises the following:⁵

OWL Classes and Properties for All BPMN 2.0 Elements That are Relevant for the Business Process Model.⁶ The OWL classes, whenever possible, follow the conventions in the BPMN 2.0 UML meta-model. OWL properties and restrictions are included by adding all of the data and object properties according to the attributes and class associations in the BPMN 2.0 model.⁷

⁴ Visualized using TopBraid Composer™.

⁵ All of the classes of the BPMN 2.0 meta model that exist for technical reasons only (model m:n relationship or special containments) are not represented in the ontology.

⁶ The work in [2] describes an ontology based on BPMN 1.x for which no standardized meta model exists.

⁷ Oracle formulated SPARQL queries for envisioned use cases and added additional properties and restrictions to the ontology to support those use cases.

OWL Classes and Properties for Instantiations of a BPMN 2.0 Process Model.

These OWL classes cover the runtime aspects of a BPMN 2.0 process when executed by a process engine. The process engine creates BPMN 2.0 flow element instances when the process is executed. Activity logging information is captured, including timestamps for a flow element instance's activation and completion, as well as the performer of the task.

The implicit (unstated) relationships in the ontology for BPM can be automatically discovered using semantic inferencing, for example as provided by the native inferencing engine included with Oracle Database Semantic Technologies. The explicit and implicit relationships in the ontology can be queried using SPARQL pattern matching queries and, in the case of Oracle Database, can also be queried using SPARQL patterns in SQL queries. [5] Example SPARQL queries are shown below:

Select all User Tasks in all Lanes

```
select ?usertask ?lane
  where {
    ?usertask rdf:type bpmn:UserTask .
    ?usertask bpmn:inLane ?lane
  }
```

Select all flow elements with their sequence flow in lane p1:MyLane (a concrete instance of RDF type bpmn:Lane)

```
select ?source ?target
  where {
    ?flow bpmn:sourceFlowElement ?source .
    ?flow bpmn:targetFlowElement ?target .
    ?target bpmn:inLane p1:MyLane
  }
```

Select all activities in process p1:MyProcess that satisfy service level agreement (SLA) p1:MySLA

```
select ?activity ?activityInstance
  where {
    ?activity bpmn:inProcess p1:MyProcess .
    ?activityInstance obpm:instanceOf ?activity .
    ?activityInstance obpm:meetSLA p1:MySLA
  }
```

Representative examples⁸ of the BPMN 2.0 ontology elements in OWL functional syntax⁹ are listed below.¹⁰

⁸ The complete BPMN 2.0 ontology comprises of about 100 OWL classes.

⁹ OWL 2 Structural Spec. and Functional-Style Syntax, <http://www.w3.org/TR/owl2-syntax>

Start Event:

```
Declaration(Class(bpmn:StartEvent))
SubClassOf(bpmn:StartEvent bpmn:CatchEvent)
SubClassOf(bpmn:StartEvent DataMaxCardinality(1
bpmn:isInterrupting))
```

Data Association:

```
Declaration(Class(bpmn:DataAssociation))
SubClassOf(bpmn:DataAssociation bpmn:BaseElement)
SubClassOf(bpmn:DataAssociation DataMaxCardinality(1
bpmn:hasAssignment))
SubClassOf(bpmn:DataAssociation DataMaxCardinality(1
bpmn:dataAssociationSource))
SubClassOf(bpmn:DataAssociation DataMaxCardinality(1
bpmn:dataAssociationTarget))
SubClassOf(bpmn:DataAssociation DataMaxCardinality(1
bpmn:hasTransformation))
```

Task:

```
Declaration(Class(bpmn:Task))
SubClassOf(bpmn:Task bpmn:Activity)
DisjointClasses(bpmn:Task bpmn:CallActivity)
DisjointClasses(bpmn:Task bpmn:SubProcess)
```

ItemAwareElement:

```
Declaration(Class(bpmn:ItemAwareElement))
SubClassOf(bpmn:ItemAwareElement bpmn:BaseElement)
SubClassOf(bpmn:ItemAwareElement DataMaxCardinality(1
bpmn:hasItemDefinition))
```

5 An Ontology for SCA

The SCA composite model ontology represents the SCA assembly model. The Oracle BPM implementation comprises OWL classes for Composite, Component, Service, Reference and Wire, which form the major building blocks of the assembly model. A SCA ontology can be specified with OWL classes for concrete services specified by WSDL and data structures specified by XML-Schema. The transformation of the SCA assembly model to the SCA ontology includes creating finer grained WSDL and XML-Schema artifacts to capture the dependencies and relationships between concrete WSDL operations and messages to elements of some XML-Schema and their imported schemata.

¹⁰ Generated from the source OWL via
<http://owl.cs.manchester.ac.uk/converter/>

The SCA ontology is primarily created for the purpose of governance and to act as a bridge between the ontology for BPM and an ontology that would represent a concrete runtime infrastructure. This enables the important ability to perform impact analysis, identify dependencies and prevent unnecessary proliferation.

6 Applications

The ontology for BPM enables a rich set of applications based on a common model.

6.1 Continuous Process Refinement and Optimization

In modern enterprises, business processes are continuously refined and optimized for better business performance and to adapt to market changes. Reading from the bottom, the diagram in Figure 3 visualizes this refinement and optimization loop.

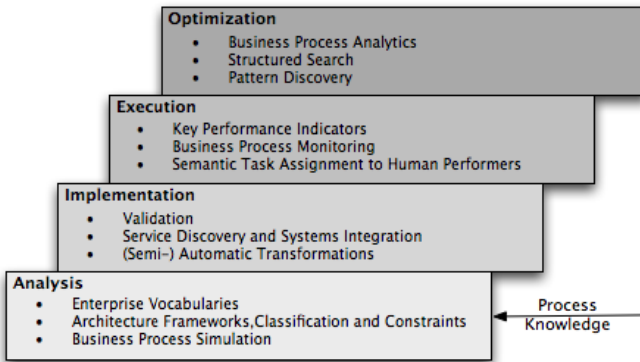


Fig. 3. Applications of Ontology for BPM

6.1.1 Process Analysis

In the process analysis stage, the ontology for BPM allows the process modeling toolset to constrain the palette of modeling elements depending on a chosen architecture framework (like DoDAF), and process tasks and data objects associated with concepts of the given enterprise vocabulary. The ontology can also help identify the people in the organization responsible for a specific task.

The ontology associates custom classification models, thesauri, and people's skill sets with process models and tasks, fosters re-use of existing tasks, constrains process models to appropriate modeling elements, and establishes a relationship between the process models and the broader concepts of the enterprise. It also enables impact and dependencies analysis for all artifacts (data, services, tasks, and people) involved in a business process. This can reduce the proliferation of services and processes and the risk of disrupting the business by introducing incompatible service contracts.

6.1.2 Implementation

An important aspect of implementing a business process for execution is validating the correctness of the process relative to the process blueprint from the analysis stage.

The ontology for BPM provides relationship metadata that the business process modeling toolset can use to validate semantic equivalence between the process model defined in the analysis stage and the process engine's execution. Validation includes ensuring the process and data flow conform with the organization's enterprise architectures.

6.1.3 Execution

During execution of a business process, the process engine generates events that can be used for business activity monitoring and identifies the human resources for the User Tasks in the business process.

The ontology for BPM can be used to provide a more precise identification of a human performer for the user tasks in the business process by relating specific attributes about an organization and skill sets. In addition, the events from activity monitoring can be investigated in the broader context by relating relevant information about business strategies and service level agreements.

6.1.4 Optimization

Optimization of a business process requires a holistic view of the process including the systems, services and people involved in the execution of business process.

The ontology for BPM can be referenced for semantically-enabled searches that provide a more complete perspective of the artifacts involved in a business process and discover patterns in a business process to foster re-use, reduce redundant work and identify bottlenecks.

6.2 Systems Integration

Modern business applications can expose thousands of SOA Services that have the potential to be re-used and integrated in custom business processes.

The ontology for BPM can be used as a source to search for services or composite applications that might be used in the implementation of a business process task. It can further support the developer in the modeling of data transformations by relating structural and semantic attributes of the source and target data objects. It is quite normal for the services exposed by commercial business applications to define hundreds of attributes for the service input and output. Semantic technology can help the systems integrator automate the process of mapping those attributes, a task that is time consuming and error prone if done manually.

7 Conclusion

An ontology for BPM, such as the one implemented for Oracle BPM, encompasses and expands the generic ontologies for BPMN 2.0 and the SOA composite model. It

covers all artifacts of a BPM application from a potentially underspecified¹¹ process model in BPMN 2.0 down to the XML-Schema type level. The combination of BPM with RDF/OWL data storage, inferencing and SPARQL querying, as supported by Oracle Database Semantic Technologies, provides the ability to discover implicit relationships in the BPM models and find patterns beyond what is possible with the classical approaches of XML-Schema, XQuery and SQL. This promotes continuous process refinement, comprehensive impact analysis, and reuse to minimize unnecessary proliferation of processes and services.

Acknowledgements. We'd like to thank Sudeer Bhoja, Linus Chow, Xavier Lopez, Bhagat Nainani and Zhe Wu for their contributions to the paper and valuable comments.

References

- [1] Business Process Model and Notation (BPMN) Version 2.0, <http://www.omg.org/spec/BPMN/2.0/>
- [2] Ghidini, C., Rospocher M., Serafini L.: BPMN Ontology, https://dkm.fbk.eu/index.php/BPMN_Ontology
- [3] Oracle Database Semantic Technologies, <http://www.oracle.com/technetwork/database/options/semantic-tech/>
- [4] Service Component Architecture (SCA), <http://www.osoa.org>
- [5] Kolovski, V., Wu, Z., Eadon, G.: Optimizing Enterprise-Scale OWL 2 RL Reasoning in a Relational Database System. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 436–452. Springer, Heidelberg (2010)
- [6] “Primitives and Style: A Common Vocabulary for BPM across the Enterprise”; Dennis Wisnosky, Chief Architect & CTO ODCMO and Linus Chow Oracle; BPM Excellence in Practice 2010; Published by Future Strategies (2010)
- [7] Web Ontology Language (OWL), <http://www.w3.org/owl>

¹¹ A BPMN 2.0 model element is considered underspecified, if its valid but not all attribute values relevant for execution are specified.

Shining Light on Complex RDF Data through Advanced Data Visualization

Francois Bertault, Wendy Feng, Austris Krastins,
Liangrong Yi, and Arturs Verza

Tom Sawyer Software, 181 Montecito Avenue, 94610 Oakland, USA
{fbertault,wfeng,akrastins,lyi,averza}@tomsawyer.com

Abstract. This Demonstration paper discusses ongoing work at Tom Sawyer Software in the area of advanced visualization and analysis of very large data sets, including RDF data. There is a growing imperative to explore large data sets as part of opportunity and threat analysis in areas such as national defense, financial risk analysis, market intelligence, and disease epidemiology. An increasing volume of this type of information is represented as RDF graphs. By visualizing and visually analyzing data, it is possible to see patterns, trends, and outliers in complex RDF graphs that would otherwise be difficult or even impossible to discover. Since RDF graphs are by nature difficult for humans to read, Tom Sawyer Software has been developing an innovative, graphical approach to defining the schema of RDF data, visualizing salient parts of the RDF graph, and integrating social network analysis into the visualization process to provide intuitive visual navigation, query, and understanding of information. This Demonstration paper discusses the underlying technology and its realization in sophisticated software for building advanced data visualization and analysis applications for making sense of large RDF graphs.

Keywords: RDF Data Visualization, Graph Drawing, Application Development.

1 Introduction

The Semantic Web data model of Resource Description Framework [1] (RDF) is a suite of W3C recommendations for representing information. It provides a simple data model where information is represented as a labeled directed graph, formal semantics that enable the definition of inference rules on the data, and a clear XML based syntax. In addition, the SPARQL query language for RDF [2] provides an efficient mechanism for retrieving RDF data. This makes RDF a natural format for representing large data sets, in particular data sets that convey relationships between entities.

A number of applications [3,4,5,6] allow for the visualization and editing of RDF data. These applications are meant to help users view and manipulate the RDF graph itself or to run queries on the RDF graph. They are, in essence, the equivalent of the administration tools for viewing or querying tables and table content in relational databases. They give insight on how the information is structured and represented in RDF form, and appeal to users who are interested in the RDF representation aspect of the data.

Tom Sawyer Perspectives is a departure from these tools. It allows users to create data visualization and social network analysis applications, where the data is stored in RDF or other formats, and then presented to the user in a way that matches the user’s mental representation of what the data actually conveys. This allows for the creation of applications that leverage the technical power of RDF while isolating the users from the fact that the application uses RDF as a way to store the information.

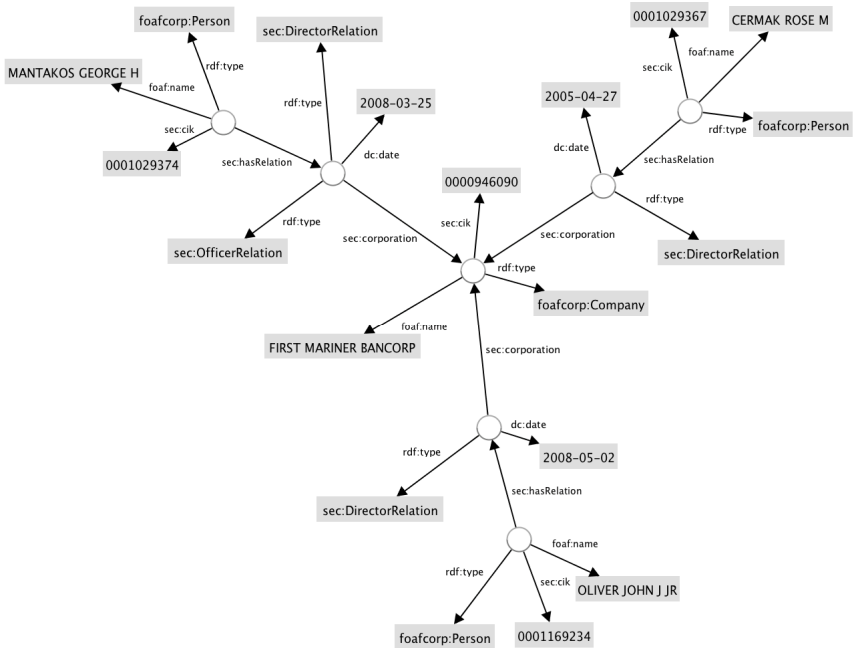


Fig. 1. Representation of corporate filing data as a RDF graph

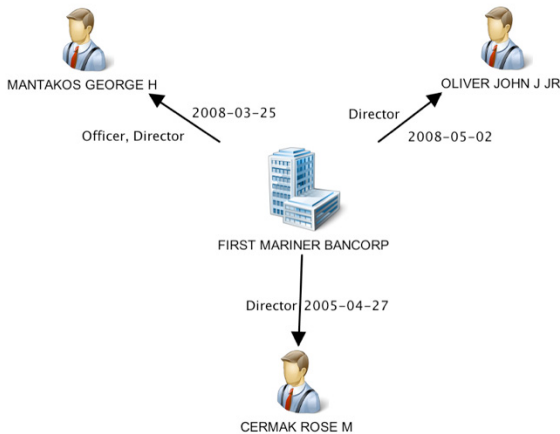


Fig. 2. Representation of corporate filing data using Tom Sawyer Perspectives

2 System Architecture

Tom Sawyer Perspectives is a software package for creating advanced data visualization and social network analysis applications. It consists of a graphical design and preview environments, with a set of API libraries. The overall architecture consists of different layers (Fig. 3).

The data model layer defines the type of data to be visualized. This allows for the definition of domain specific data models relevant to the application being developed.

The data integration layer provides read-and-write access to disparate data sources and a mechanism for aggregating the data into one unified view of the data. The data integration layer supports several types of data sources, including databases or XML, RDF, text and Excel documents. It also provides conflict detection and merging capabilities to support multi-users environments.

The data visualization layer provides the different views of data. It adds support for hierarchical tree, tabular, and drawing representations. The drawing view allows for the representation of relationships in the data as a nested graph structure. That view supports advanced formatting with automatic layout capabilities in different styles commonly used in various application domains, such as electrical and CAD-oriented schematics, software engineering, or data and network modeling.

The visual interaction layer provides user interactions to help end users view, navigate, and edit the data.

The design environment is a graphical application used to define the components from the different layers. Users define the type of data the application is handling, how to access the data sources, and how to represent and interact with the data. The resulting design file can be deployed to create either desktop or web applications.

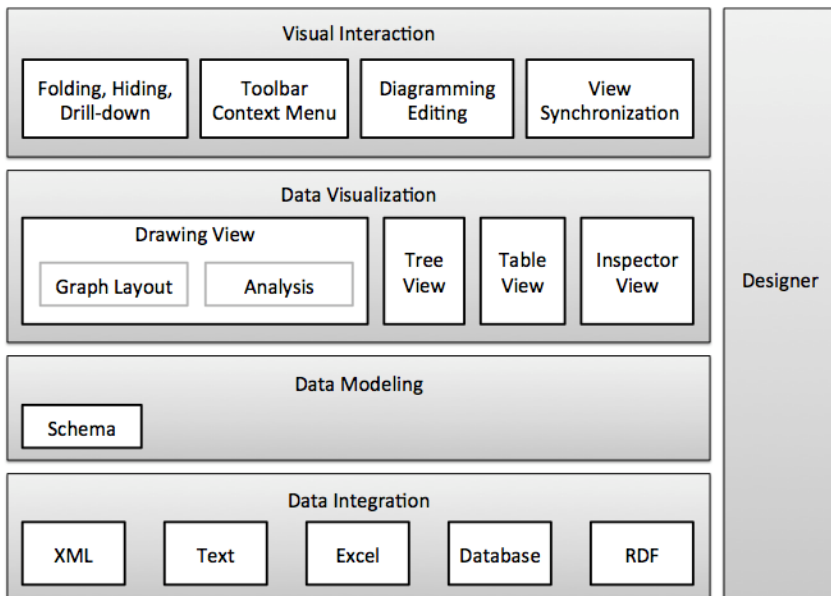


Fig. 3. Overall architecture of Tom Sawyer Perspectives

3 Demonstration

The demonstration covers two aspects of Tom Sawyer Perspectives. The first part of the demonstration shows the overall functionality of a typical application built using Tom Sawyer Perspectives. The second part of the demonstration shows how to build a typical visualization application using the Tom Sawyer Perspectives Designer. The data for this demonstration consists of corporate filing data as reported to the U.S. Securities and Exchange Commission. The data is stored in RDF format in an Oracle database, and consists of 1.8M triplets.

The demonstration application consists of different views, each supporting dynamic filtering, where information can be selectively shown based on type or connectivity.

The first view (Fig. 4) shows corporations and people related to these corporations. It lets users search for particular corporations of interest, then retrieve additional content by selecting individual items. Users may also view only a subset of the data and load additional information on demand, providing a user-guided traversal of the data space.

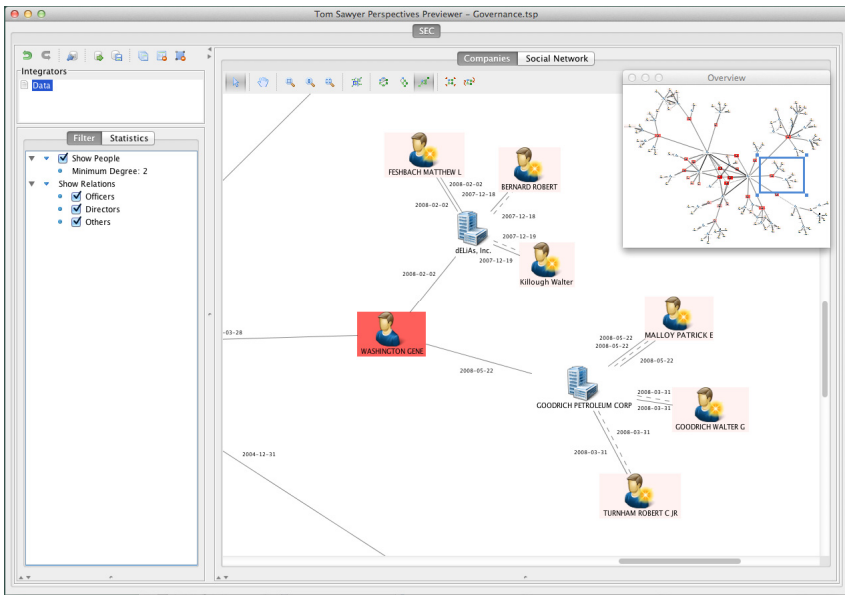


Fig. 4. View of corporate filing data. Elements with a highlighted icon indicate that related information is available.

The second view (Fig. 5) shows the same data, but with the corporations omitted. This view shows indirect relationships between people by inferring a relationship if two people are linked to a same corporation. It assumes that two people who work together are likely to know each other and be part of a larger network. This view allows us to run some social network analysis on the network of people, and deduce who are the influential people in the network (Fig. 5 and Fig. 6).

The second part of the demonstration (Fig. 7) shows how to build a typical visualization application using the Tom Sawyer Perspectives Designer. It covers defining the data model, connecting to a data source, defining of rule-based views and the filter.

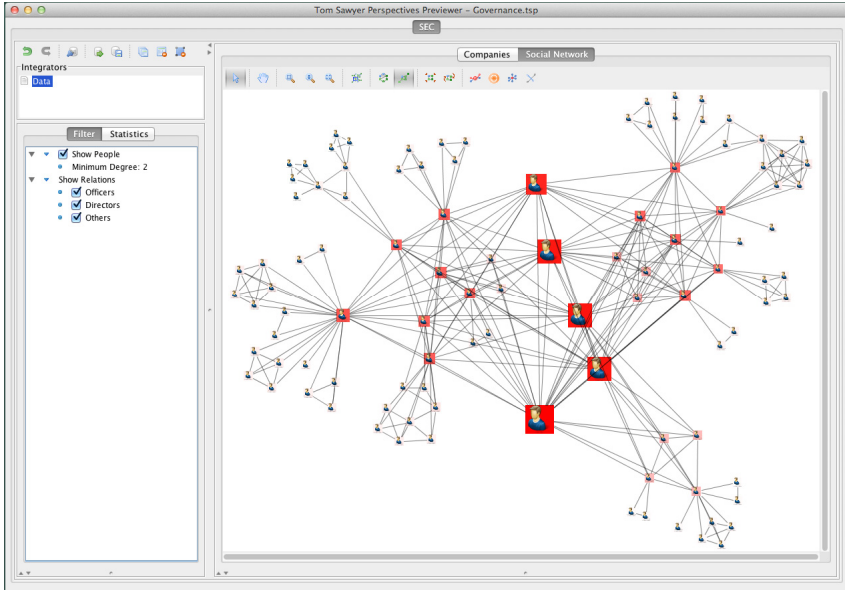


Fig. 5. View of a social network with overlaid centrality measures

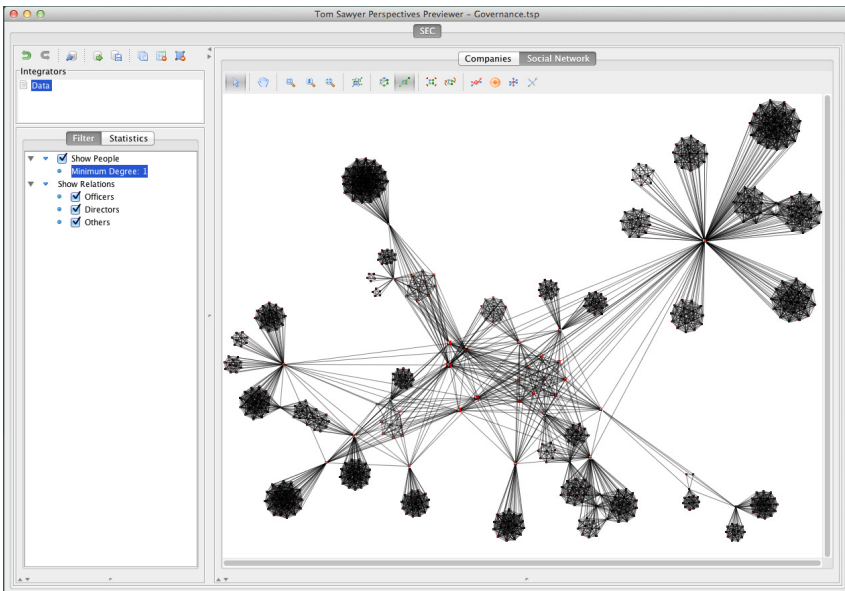


Fig. 6. View of a large social network

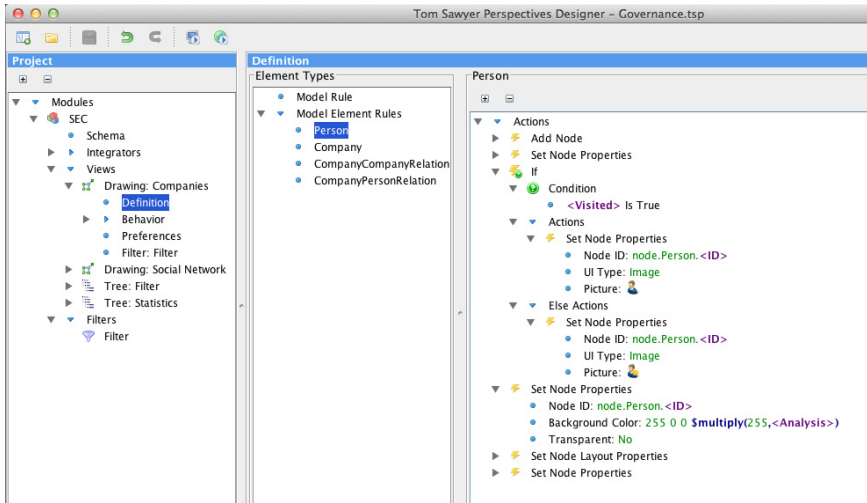


Fig. 7. Tom Sawyer Designer with a rule-based definition of how to represent a person in the Drawing view

4 Conclusion

We presented a demonstration of an application built using Tom Sawyer Perspectives. The demonstration illustrates how combining an RDF data store with advanced visualization techniques can provide a compelling solution for visualizing and analyzing very large data sets. This demonstrates how these visual analytic techniques can complement traditional semantic analysis of RDF data.

References

1. Manola, F., Miller, E.: RDF primer. W3C recommendation (2004)
2. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C candidate recommendation (2006)
3. Simile, <http://simile.mit.edu/welkin>
4. Splendiani, A.: Semantic browsing of pathway ontologies and biological networks with RDFScope. In: Managing and Mining Genome Information: Frontiers in Bioinformatics. Dagstuhl Seminar Proceedings (2006)
5. IsaViz, <http://www.w3.org/2001/11/IsaViz>
6. RDFGravity, <http://semweb.salzburgresearch.at/apps/rdf-gravity>

OntoRevision: A Plug-in System for Ontology Revision in Protégé

Nathan Cobby¹, Kewen Wang^{1,*}, Zhe Wang², and Marco Sotomayor¹

¹ Griffith University, Australia

² Oxford University, UK

k.wang@griffith.edu.au

Abstract. Ontologies have been widely used in advanced information systems. However, it has been a challenging issue in ontology engineering to efficiently revise ontologies as new information becomes available. A novel method of revising ontologies has been proposed recently by Wang et al. However, related algorithms have not been implemented yet. In this article we describe an implementation of these algorithms called OntoRevision and report some experimental results. Our system is a plug-in for revising general ontologies in Protégé and thus can be used by Protégé users to revise ontologies automatically.

1 Introduction

In knowledge engineering, an *ontology* is a formal model of some domain knowledge of the world [6], by providing a shared *vocabulary* relevant to the domain, specification of the *meaning* (semantics) of the terms, and a formalized specification of the conceptualization. Ontologies have been applied in a wide range of practical domains such as e-Science, e-Commerce, medical informatics, bio-informatics, and the Semantic Web.

As with all knowledge formalizing structures, ontologies are not static, but may evolve over time. In particular, ontologies may need to be extended and sometimes revised. Although the operation of incorporating an ontology into another existing ontology is supported by Protégé[1], it does not provide any machinery to assure the validity or usefulness of such incorporation. Firstly, classes with the same name in different ontologies are, by default, considered to be distinct. When incorporating two ontologies, classes with the same name co-exist in the resulting ontology. For instance, suppose we have two ontologies both with a class called *Student*. When merging the two ontologies, two classes both named *Student* will occur in the result. The two classes can only be distinguished when we refer to their respective URI inherited from their source ontologies. Secondly, suppose we can change the URI of the two classes *Student* to unify them, another problem occurs when the knowledge in the two ontologies contradicts to each other. In such case, Protégé simply combine the two ontologies leaving the result inconsistent. Although Protégé can detect such inconsistency, no solution is provided to resolve the inconsistency.

* Corresponding author.

¹ <http://protege.stanford.edu>

Recently, a novel framework for revising ontologies in DL-Lite is introduced in [7]. The DL-Lite [21], which forms the basis of OWL 2 QL [3], is a family of lightweight DLs with efficient ontology reasoning and query answering algorithms. However, Wang et al's algorithm for ontology algorithms has not been implemented yet. In this article we describe a reasoning system for ontology revision called *OntoRevision*². This system is an implementation of Wang et al's original revision algorithm and an improved algorithm. Our system is a plug-in for revising general ontologies in Protégé and thus can be used by Protégé users to revise ontologies automatically. We also report some preliminary experimental results.

2 Feature-Based Revision

In this section we briefly recall some basics of ontology revision introduced in [7]. The revision operator is based on a new semantic characterization called *features*. So we first introduce the definition of features.

2.1 An Alternative Semantics for DL-Lite

A *signature* is a finite set $\mathcal{S} = \mathcal{S}_C \cup \mathcal{S}_R \cup \mathcal{S}_I \cup \mathcal{S}_N$ where \mathcal{S}_C is the set of atomic concepts, \mathcal{S}_R is the set of atomic roles, \mathcal{S}_I is the set of individual names and \mathcal{S}_N is the set of natural numbers in \mathcal{S} . We assume 1 is always in \mathcal{S}_N . \top and \perp will not be considered as atomic concepts or atomic roles. Formally, given a signature \mathcal{S} , a DL-Lite ^{\mathcal{R}, \mathcal{N}} _{bool} language has the following syntax:

$$\begin{array}{ll} R \leftarrow P \mid P^- & S \leftarrow P \mid \neg P \\ B \leftarrow \top \mid A \mid \geq n R & C \leftarrow B \mid \neg C \mid C_1 \sqcap C_2 \end{array}$$

where $n \in \mathcal{S}_N$, $A \in \mathcal{S}_C$ and $P \in \mathcal{S}_R$. B is called a *basic concept* and C is called a *general concept*. We write \perp as a shorthand for $\neg\top$, $\exists R$ for $\geq 1 R$, $\leq n R$ for $\neg(\geq n + 1 R)$, and $C_1 \sqcup C_2$ for $\neg(\neg C_1 \sqcap \neg C_2)$. Let $R^+ = P$, where $P \in \mathcal{S}_R$, whenever $R = P$ or $R = P^-$.

A *TBox* \mathcal{T} is a finite set of *concept inclusions* of the form $C_1 \sqsubseteq C_2$ with C_1 and C_2 being general concepts, and *role inclusions* of the form $R_1 \sqsubseteq R_2$. An *ABox* \mathcal{A} is a finite set of membership *assertions* of the form $C(a)$ or $S(a, b)$, where a, b are individual names. We call $C(a)$ a concept assertion and $S(a, b)$ a role assertion. A knowledge base (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$. In this paper, a DL ontology is represented as a DL KB. We will use ontology and KB alternatively.

Features for DL-Lite ^{\mathcal{N}} _{bool} are based on the notion of *types* defined in [5]. An \mathcal{S} -type τ is a set of basic concepts over \mathcal{S} such that $\top \in \tau$, and for any $m, n \in \mathcal{S}_N$ with $m < n$, $\geq n R \in \tau$ implies $\geq m R \in \tau$. When the signature \mathcal{S} is clear from context, we will simply call an \mathcal{S} -type a *type*. As $\top \in \tau$ for any type τ , we omit it in examples for simplicity. For example, let $\mathcal{S}_C = \{A, B\}$, $\mathcal{S}_R = \{P\}$, and $\mathcal{S}_N = \{1, 3\}$. Then $\tau = \{A, \exists P, \geq 3 P, \exists P^-\}$ is a type.

² <http://www.ict.griffith.edu.au/~kewen/OntoRevision/>

Define a type τ *satisfying* a concept in the following way: τ satisfies basic concept B if $B \in \tau$, τ satisfies $\neg C$ if τ does not satisfy C , and τ satisfies $C \sqcap D$ if τ satisfies both C and D .

We can also define a type τ *satisfies* concept inclusion $C \sqsubseteq D$ if τ satisfies concept $\neg C \sqcup D$. Type τ *satisfies* a TBox \mathcal{T} if it satisfies every inclusion in \mathcal{T} .

Types are sufficient to capture the semantics of TBoxes, but as they do not refer to individuals, they are insufficient to capture the semantics of ABoxes. We need to extend the notion of types with individuals and thus define *Herbrand sets* in DL-Lite.

Definition 2.1. An \mathcal{S} -Herbrand set (or simply Herbrand set) \mathcal{H} is a finite set of assertions of the form $B(a)$ or $P(a, b)$, where $a, b \in \mathcal{S}_I$, $P \in \mathcal{S}_R$ and B is a basic concept over \mathcal{S} , satisfying the following conditions

1. For each $a \in \mathcal{S}_I$, $\top(a) \in \mathcal{H}$, and $\geq n R(a) \in \mathcal{H}$ implies $\geq m R(a) \in \mathcal{H}$ for $m, n \in \mathcal{S}_N$ with $m < n$.
2. For each $P \in \mathcal{S}_R$, $P(a, b_i) \in \mathcal{H}$ ($i = 1, \dots, n$) implies $\geq m P(a) \in \mathcal{H}$ for any $m \in \mathcal{S}_N$ such that $m \leq n$.
3. For each $P \in \mathcal{S}_R$, $P(b_i, a) \in \mathcal{H}$ ($i = 1, \dots, n$) implies $\geq m P^-(a) \in \mathcal{H}$ for any $m \in \mathcal{S}_N$ such that $m \leq n$.

We use \mathcal{H}_R to denote the set of all role assertions in \mathcal{H} . Given a Herbrand set \mathcal{H} for a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an individual a , $\tau(a, \mathcal{H}) = \{C \mid C(a) \in \mathcal{H}\}$ is a type, called the *type of a in \mathcal{H}* .

We define a Herbrand set \mathcal{H} *satisfies* concept assertion $C(a)$ if $\tau(a, \mathcal{H})$ satisfies concept C . Herbrand set \mathcal{H} *satisfies* role assertion $P(a, b)$ if $P(a, b)$ is in \mathcal{H} , and $\neg P(a, b)$ if $P(a, b)$ is not in \mathcal{H} . Herbrand set \mathcal{H} *satisfies* an ABox \mathcal{A} if \mathcal{H} satisfies every assertion in \mathcal{A} . The concept of *features* is defined as follows.

Definition 2.2 (Features). Given a signature \mathcal{S} , an \mathcal{S} -feature (or simply feature) is defined as a pair $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$, where Ξ is a non-empty set of \mathcal{S} -types and \mathcal{H} a \mathcal{S} -Herbrand set, satisfying the following conditions:

1. $\exists P \in \bigcup \Xi$ iff $\exists P^- \in \bigcup \Xi$, for each $P \in \mathcal{S}_R$.
2. $\tau(a, \mathcal{H}) \in \Xi$, for each $a \in \mathcal{S}_I$.

Example 2.1. Consider the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where

$$\begin{aligned} \mathcal{T} &= \{ A \sqsubseteq \exists P, B \sqsubseteq \exists P, \exists P^- \sqsubseteq B, A \sqcap B \sqsubseteq \perp, \geq 2 P^- \sqsubseteq \perp \} \\ \mathcal{A} &= \{ A(a), P(a, b) \}. \end{aligned}$$

Take $\mathcal{S} = \text{sig}(\mathcal{K}) = \{A, B, P, 1, 2, a, b\}$. Then $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$ is a (finite) model feature of \mathcal{K} , where

$$\begin{aligned} \Xi &= \{ \tau_1, \tau_2 \} \text{ with } \tau_1 = \{A, \exists P\} \text{ and } \tau_2 = \{B, \exists P, \exists P^-\}, \text{ and} \\ \mathcal{H} &= \{ A(a), \exists P(a), B(b), \exists P(b), \exists P^-(b), P(a, b) \}. \end{aligned}$$

Definition 2.3. Given a feature $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$, we say \mathcal{F} satisfies

- a concept C if there is a type in Ξ satisfying C .
- an inclusion $C \sqsubseteq D$ if τ satisfies $C \sqsubseteq D$ for all $\tau \in \Xi$.
- an assertion $C(a)$ or $S(a, b)$ if \mathcal{H} satisfies it.
- \mathcal{F} is a model feature of KB \mathcal{K} if \mathcal{F} satisfies every concept inclusion and every membership assertion in \mathcal{K} . $M_{\mathcal{F}}(\mathcal{K})$ denotes the set of all model features of \mathcal{K} .

It has been shown in [7] that the semantics defined in terms of features characterize the standard semantics of DL-Lite in terms of all major reasoning forms for DL-Lite ontologies.

A KB \mathcal{K} is said to be a *maximal approximation* of \mathbb{M} over \mathcal{S} if (1) $\text{sig}(\mathcal{K}) \subseteq \mathcal{S}$ and $\mathbb{M} \subseteq \text{mod}(\mathcal{K})$, and (2) there exists no KB \mathcal{K}' satisfying (1) such that $\text{mod}(\mathcal{K}') \subset \text{mod}(\mathcal{K})$. It is shown in [4] that maximal approximation may not exist for some DLs. However, as shown in [7], maximal approximations always exist in DL-Lite $_{bool}^N$.

2.2 Ontology Revision

Given two \mathcal{S} -features $\mathcal{F}_1 = \langle \Xi_1, \mathcal{H}_1 \rangle$ and $\mathcal{F}_2 = \langle \Xi_2, \mathcal{H}_2 \rangle$, the *distance* between \mathcal{F}_1 and \mathcal{F}_2 , denoted $\mathcal{F}_1 \Delta \mathcal{F}_2$, is a pair $\langle \Xi_1 \Delta \Xi_2, \mathcal{H}_1 \Delta \mathcal{H}_2 \rangle$. Recall that $X \Delta Y$ is the symmetric difference for any two sets X and Y .

To compare two distances, we define $\mathcal{F}_1 \Delta \mathcal{F}_2 \subseteq \mathcal{F}_3 \Delta \mathcal{F}_4$ if $\Xi_1 \Delta \Xi_2 \subseteq \Xi_3 \Delta \Xi_4$ and $\mathcal{H}_1 \Delta \mathcal{H}_2 \subseteq \mathcal{H}_3 \Delta \mathcal{H}_4$; and $\mathcal{F}_1 \Delta \mathcal{F}_2 \subset \mathcal{F}_3 \Delta \mathcal{F}_4$ if $\mathcal{F}_1 \Delta \mathcal{F}_2 \subseteq \mathcal{F}_3 \Delta \mathcal{F}_4$ and $\mathcal{F}_3 \Delta \mathcal{F}_4 \not\subseteq \mathcal{F}_1 \Delta \mathcal{F}_2$.

Definition 2.4 (F-Revision). Let $\mathcal{K}, \mathcal{K}'$ be two DL-Lite $_{bool}^N$ KBs and $\mathcal{S} = \text{sig}(\mathcal{K} \cup \mathcal{K}')$. Define the f -revision of \mathcal{K} by \mathcal{K}' , denoted $\mathcal{K} \circ_f \mathcal{K}'$, such that $M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}') = M_{\mathcal{F}}(\mathcal{K}')$ if $M_{\mathcal{F}}(\mathcal{K}) = \emptyset$, and otherwise

$$M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}') = \{ \langle \Xi', \mathcal{H}' \rangle \in M_{\mathcal{F}}(\mathcal{K}') \mid \exists \langle \Xi, \mathcal{H} \rangle \in M_{\mathcal{F}}(\mathcal{K}) \text{ s.t.} \\ \mathcal{H} \Delta \mathcal{H}' \in d_H(\mathcal{K}, \mathcal{K}') \text{ and } \langle \Xi \Delta \Xi', \mathcal{H} \Delta \mathcal{H}' \rangle \in d_F(\mathcal{K}, \mathcal{K}') \}.$$

where

$$d_H(\mathcal{K}_1, \mathcal{K}_2) = \min_{\subseteq} (\{ \mathcal{H}_1 \Delta \mathcal{H}_2 \mid \exists \langle \Xi_1, \mathcal{H}_1 \rangle \in M_{\mathcal{F}}(\mathcal{K}_1), \exists \langle \Xi_2, \mathcal{H}_2 \rangle \in M_{\mathcal{F}}(\mathcal{K}_2) \}), \\ d_F(\mathcal{K}_1, \mathcal{K}_2) = \min_{\subseteq} (\{ \mathcal{F}_1 \Delta \mathcal{F}_2 \mid \exists \mathcal{F}_1 \in M_{\mathcal{F}}(\mathcal{K}_1), \exists \mathcal{F}_2 \in M_{\mathcal{F}}(\mathcal{K}_2) \})$$

Example 2.2. Consider the following knowledge base,

$$\mathcal{K} = \langle \{ \text{PhDStudent} \sqsubseteq \text{Student} \sqcap \text{Postgrad}, \\ \text{Student} \sqsubseteq \neg \exists \text{teaches}, \exists \text{teaches}^- \sqsubseteq \text{Course}, \\ \text{Student} \sqcap \text{Course} \sqsubseteq \perp \}, \{ \text{PhDStudent}(\text{Tom}) \} \rangle.$$

The TBox of \mathcal{K} specifies that PhD students are postgraduate students, and students are not allowed to teach any courses, while the ABox states that *Tom* is a PhD student. Suppose PhD students are actually allowed to teach, and we want to revise \mathcal{K} with

$$\mathcal{K}' = \langle \{ \text{PhDStudent} \sqsubseteq \exists \text{teaches} \}, \emptyset \rangle.$$

Then, $\mathcal{K} \circ_f \mathcal{K}'$ is

$$\langle \{ \text{PhDStudent} \sqsubseteq \text{Student} \sqcap \text{Postgrad}, \text{PhDStudent} \sqsubseteq \exists \text{teaches}, \\ \text{Student} \sqcap \exists \text{teaches} \sqsubseteq \text{PhDStudent}, \exists \text{teaches}^- \sqsubseteq \text{Course}, \\ \text{Student} \sqcap \text{Course} \sqsubseteq \perp \}, \{ \text{Student}(\text{Tom}), \text{Postgrad}(\text{Tom}) \} \rangle.$$

2.3 Algorithms for Ontology Revision

In this section, we introduce an algorithm for computing the maximal approximation of revision syntactically and briefly explain how it can be improved.

Given a \mathcal{S} -type τ , we denote the concept $C_\tau = \prod_{B \in \tau} B \sqcap \prod_{B \notin \tau} \neg B$, where B is a basic concept over \mathcal{S} . In what follows, we present an algorithm for DL-Lite $_{bool}^N$ KB revision (ref. Figure 1).

Algorithm 1

Input: Two DL-Lite $_{bool}^N$ KBs \mathcal{K} and \mathcal{K}' , $\mathcal{S} = \text{sig}(\mathcal{K} \cup \mathcal{K}')$.

Output: $\mathcal{K} \circ_f \mathcal{K}'$.

Method: Initially, let $\mathcal{T} = \emptyset$ and $\mathcal{A} = \emptyset$.

Step 1. Compute $M_{\mathcal{F}}(\mathcal{K})$ and $M_{\mathcal{F}}(\mathcal{K}')$.

Step 2. Obtain $M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}')$ from $M_{\mathcal{F}}(\mathcal{K})$ and $M_{\mathcal{F}}(\mathcal{K}')$ by Definition 2.4

Step 3. For each \mathcal{S} -type τ not occurring in any type set in $M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}')$, add inclusion $C_\tau \sqsubseteq \perp$ into \mathcal{T} .

Step 4. For each individual $a \in \mathcal{S}_I$, add concept assertion $(\bigsqcup_{\tau \in \Xi_a} C_\tau)(a)$ into \mathcal{A} , where $\Xi_a = \{ \tau \mid \exists \langle \Xi, \mathcal{H} \rangle \in M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}') \text{ s.t. } \tau \text{ is the type of } a \text{ in } \mathcal{H} \}$.

Step 5. For each role assertion $P(a, b)$ occurring in every Herbrand set in $M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}')$, add $P(a, b)$ into \mathcal{A} .

Step 6. Return $\langle \mathcal{T}, \mathcal{A} \rangle$ as $\mathcal{K} \circ_f \mathcal{K}'$.

Fig. 1. Compute f-revision

In general, it is inefficient to compute the set of features for an ontology. For this reason, we have developed an improved algorithm. In particular, we only need to consider subsets of $M_{\mathcal{F}}(\mathcal{K})$ and $M_{\mathcal{F}}(\mathcal{K}')$ when selecting the model features of the revision.

The optimisation is based on the following observations when selecting model features $\mathcal{F} = \langle \Xi, \mathcal{H} \rangle$ for the revision. Firstly, we can compare the Herbrand sets \mathcal{H} independently from the type sets Ξ , and eliminate those features whose Herbrand sets do not have a minimal distance. Secondly, we do not need to consider all the Herbrand sets, but only those containing only role assertions explicitly appearing in the ABoxes \mathcal{A} and \mathcal{A}' . Thirdly, when the Herbrand sets \mathcal{H} are fixed, then the corresponding type sets Ξ can be constructed based on \mathcal{H} .

3 Implementation Details

OntoRevision is implemented in Java as a plug-in of Protégé. The system has been tested for Protégé version 4.1.0 (Build 213). To install OntoRevision, we need only

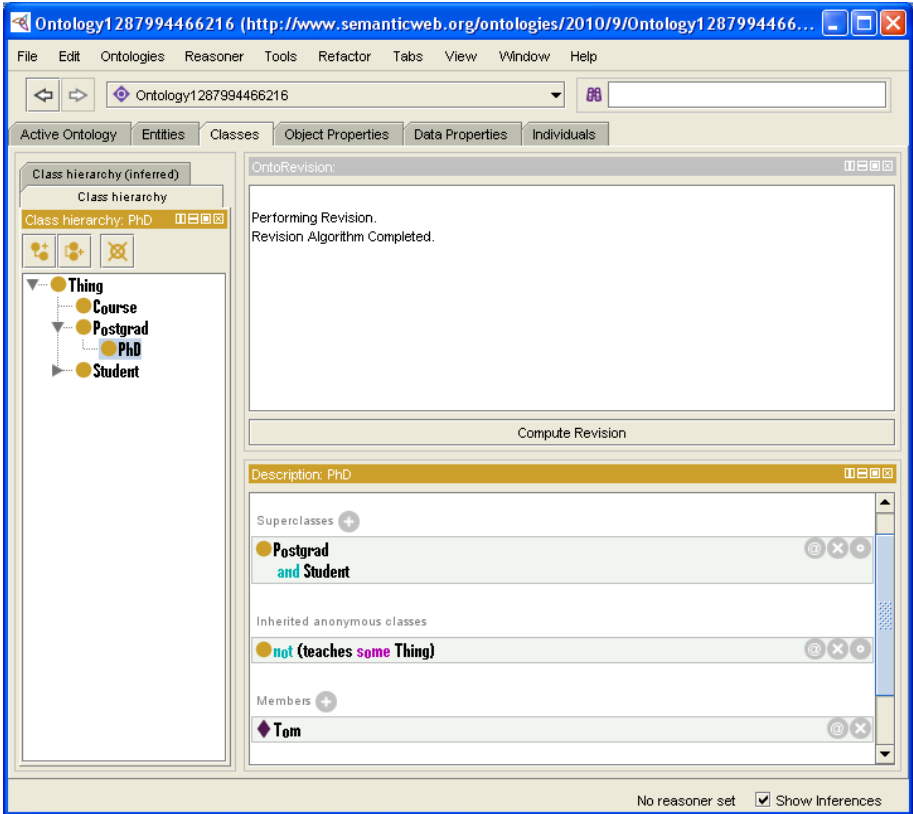


Fig. 2. A Screen Shot of OntoRevision

to copy the file `OntoRevision.jar` into Protégé's plug-in directory. Once the plug-in is installed, it can be displayed within Protégé by selecting the `OntoRevision` menu item under `View` \rightarrow `Ontology View` and placing it within the tab *user interface*. Protégé uses the Manchester OWL syntax for editing ontologies.

Besides necessary preprocessing and postprocessing, `OntoRevision` has four major modules: (1) Feature Constructor (for computing the set of features for a given KB); (2) Distance Calculator (for calculating the distance between two features); (3) Feature Selector (for picking out features with minimal distances); and (4) KB Constructor (for constructing a KB from a set of features). An input of `OntoRevision` is a pair $(\mathcal{K}, \mathcal{K}')$ of DL-Lite KBs. The system first computes the sets $M_{\mathcal{F}}(\mathcal{K})$ and $M_{\mathcal{F}}(\mathcal{K}')$ of features for \mathcal{K} and \mathcal{K}' , respectively. Then $M_{\mathcal{F}}(\mathcal{K} \circ_f \mathcal{K}')$ is obtained by the module Feature Selector, which uses the Distance Calculator. Finally, the revision result (maximal approximation) is obtained by KB Constructor. A screen shot for the completion of revision operation in Protégé is shown in Figure 3.

Some preliminary experiments have been performed on a desktop computer (Intel Pentium 4 CPU 3.4 GHz, 2 GB RAM). We compared the performance of the original

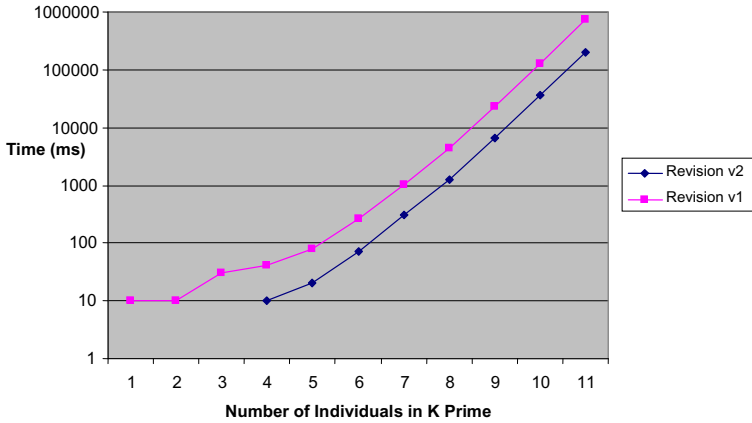


Fig. 3. Number of Individuals vs Time

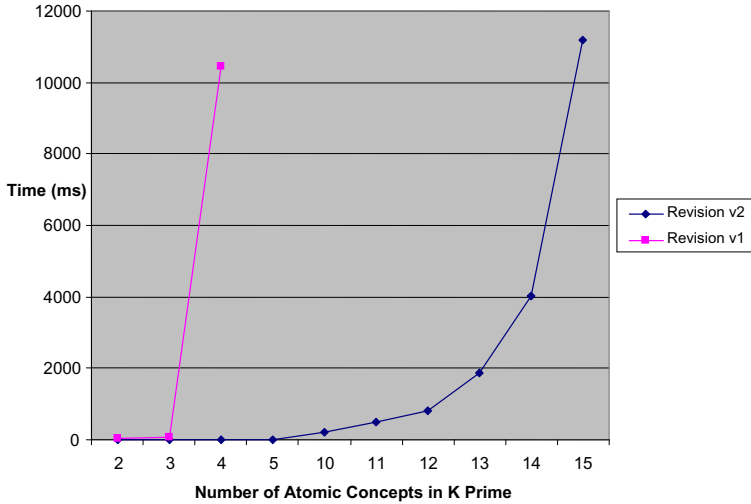


Fig. 4. Number of Atomic Concepts vs Time

algorithm for ontology revision (v1) in [7] and an (improved) version (v2). In the first example we tested the performance of two algorithms when the number of individuals in \mathcal{K}' is increased. The example used is $\mathcal{K} = (\{A \sqsubseteq \neg B\}, \{\})$ and $\mathcal{K}'_k = \langle \{A \sqsubseteq B\}, \{A(a_1), A(a_2), \dots, A(a_k)\} \rangle$ with $k > 0$. The experimental results are shown in Figure 3. It can be seen that the improved algorithm performs better than the original one but the improvement is not radical.

We also tested the performance of the two algorithms when the number of concepts in \mathcal{K}' is increased. The example used is $\mathcal{K} = \langle \{A \sqsubseteq \neg B_1\}, \{A(a)\} \rangle$ and $\mathcal{K}'_k = \langle \{A \sqsubseteq B_1, B_1 \sqsubseteq B_2, \dots, B_k \sqsubseteq B_{k+1}\}, \{\} \rangle$ with $k > 0$. The results show that the improved algorithm is significantly faster than the original one (ref. Figure 3).

4 Conclusion

We have implemented a prototype system for revising DL-Lite ontologies, called *OntoRevision*. It is able to revise general DL-Lite knowledge bases (i. e. containing both TBoxes and ABoxes). The system is implemented as a plug-in for the ontology editor *Protégé*. Some experimental results have also been reported in the paper. However, the scalability of *OntoRevision* is still a challenge. Currently, we are working on developing more efficient algorithms for DL-Lite revision.

Acknowledgments. We would like to thank all anonymous reviewers for their comments. This work was supported by the Australia Research Council (ARC) Discovery Projects DP110101042 and DP1093652.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyashev, M.: DL-Lite in the light of first-order logic. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007), pp. 361–366 (2007)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning* 39(3), 385–429 (2007)
3. Dean, M., Connolly, D., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: Owl web ontology language reference (February 10, 2004) 3C Recommendation, <http://www.w3.org/tr/2004/rec-owl-ref-20040210/>
4. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On the approximation of instance level update and erasure in description logics. In: Proceedings of the 22th National Conference on Artificial Intelligence (AAAI 2007), pp. 403–408 (2007)
5. Kontchakov, R., Wolter, F., Zakharyashev, M.: Can you tell the difference between DL-Lite ontologies? In: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 285–295. AAAI Press (2008)
6. Staab, S., Studer, R. (eds.): Handbook on Ontologies, 2nd edn. Springer, Berlin (2009)
7. Wang, Z., Wang, K., Topor, R.W.: A new approach to knowledge base revision in dl-lite. In: Proc. of 24th AAAI, pp. 369–374 (2010)

An Efficient Approach to Debugging Ontologies Based on Patterns^{*}

Qiu Ji¹, Zhiqiang Gao^{1,**}, Zhisheng Huang², and Man Zhu¹

¹ School of Computer Science and Engineering, Southeast University,
Nanjing, China

{jqiu,zqgao,mzhu}@seu.edu.cn

² Department of Mathematics and Computer Science,
Vrije University Amsterdam

huang@cs.vu.nl

Abstract. Ontology debugging helps users to understand the unsatisfiability of a concept in an ontology by finding minimal unsatisfiability-preserving sub-ontologies (MUPS) of the ontology for the concept. Although existing approaches have shown good performance for some real life ontologies, they are still inefficient to handle ontologies that have many MUPS for an unsatisfiable concept. In this paper, we propose an efficient approach to debugging ontologies based on a set of patterns. Patterns provide general information to explain unsatisfiability but are not dependent on a specific ontology. In this approach, we make use of a set of heuristic strategies and construct a directed graph w.r.t. the hierarchies where the depth-first search strategy can be used to search paths. The experiments show that our approach has gained a significant improvement over the state of the art and can find considerable number of MUPS.

1 Introduction

Ontology debugging is one of the key tasks in the Semantic Web, which helps users to understand the unsatisfiability of a named concept in an ontology. Here, a concept is unsatisfiable if it is interpreted as an empty set. Currently, various debugging approaches have been proposed. The first approach is originally proposed in [12], which is a tableaux based approach and is restricted to unfoldable \mathcal{ALC} TBoxes. In [8], the authors developed a glass-box approach based on description logic tableaux reasoner - Pellet and a black-box approach which is independent of any reasoner. As these methods are based on the entire ontologies which may contain a large number of axioms, the module extraction methods are used to improve the efficiency by extracting a relatively smaller module for an unsatisfiable concept. The work in [13] is an example.

Also, there are some works to debug ontologies by using the patterns. Patterns here provide general information to explain unsatisfiability but are not dependent

^{*} This paper is sponsored by NSFC 60873153, 60803061 and 61170165.

^{**} Corresponding author.

on a specific ontology. In [14], a heuristic strategy is proposed to detect various patterns which are identified by the experience. Although this method is efficient, for each unsatisfiable concept it only computes one explanation which is not always a MUPS. Thus, it is not enough for ontology repair and cannot meet the users' needs if more than one explanation for a concept is preferred. In [3], a set of logical inconsistent patterns (i.e. debugging patterns) is given. Based on these patterns, an ontology is debugged manually with the help of an ontology editor. So this method becomes very inefficient if many axioms are involved to explain the unsatisfiability of a concept.

Although quite a few methods have been developed, they are still inefficient to deal with those ontologies with large number of MUPS. Therefore, we propose a pattern-based approach to finding a set of MUPS efficiently. Take a commonly used ontology *bt_km* (see Section 4) with 5000 axioms as an example. The concept *hardware_system* has more than 16000 MUPS. After applying a model extraction method given in [5] (i.e., we choose "top of bottom"), the number of axioms in the extracted module becomes 124. Then the glass-box approach proposed in [7] is applied to compute all MUPS. It cannot finish the process within 30 minutes and only no more than 100 MUPS are found. But our approach only takes no more than 1000 seconds to finish the debugging process and 16719 MUPS are returned. Although our approach cannot ensure to find all MUPS, it can be regarded as a heuristic strategy of the existing approaches to computing all MUPS.

Specifically, to improve the efficiency of finding MUPS, we use a set of heuristic strategies instead of invoking a classical DL reasoner frequently which is quite time-consuming. In this approach, the ontologies need to be normalized and a directed graph is constructed for applying the efficient search strategies. The experimental results demonstrate an improvement of several orders of magnitude in efficiency by comparing with a representative existing approach. At the same time, considerable number of MUPS have been found.

The main contributions include the following: We propose a novel debugging approach by using a set of heuristic strategies based on patterns. Through experiments, it shows that our approach has gained a significant improvement over the state of the art approaches and can find considerable number of MUPS.

2 Preliminaries

We assume the readers are familiar with description logics (DLs) and refer to [1] for more details. In the followings, we give several key notations in DL ontologies.

Definition 1. (Unsatisfiable Concept)[4] *A named concept C in an ontology O is unsatisfiable iff for each model \mathcal{I} of O , $C^{\mathcal{I}} = \emptyset$. Otherwise C is satisfiable.*

Definition 2. (Incoherent Ontology)[4] *An ontology O is incoherent iff there exists at least one unsatisfiable named concept in O . Otherwise O is coherent.*

Ontology debugging deals with the problem of computing a set of minimal incoherence-preserving sub-ontologies for a specific unsatisfiable concept.

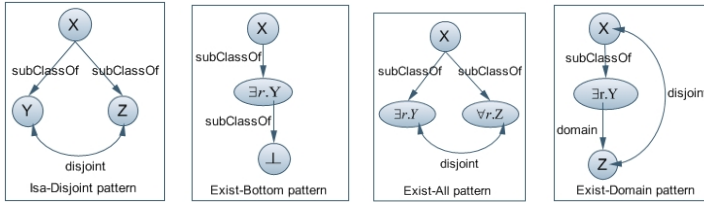


Fig. 1. A set of debugging patterns

Definition 3. (MUPS) [72] Assume C is an unsatisfiable concept in a TBox \mathcal{T} . A set $\mathcal{T}' \subseteq \mathcal{T}$ is a minimal unsatisfiability-preserving sub-TBox (MUPS) of \mathcal{T} if C is unsatisfiable in \mathcal{T}' , and C is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$.

In this paper, the computation of MUPS based on a set of debugging patterns is proposed. So far, various debugging patterns have been given in [14, 3] (e.g. the patterns in Figure 1). Each pattern can be represented as a directed graph G_P . In G_P , a vertex indicates a (possibly complex) concept and is labeled with the concept name. An arc or directed edge from vertex A to vertex B indicates a relation between A and B which can be represented by a pair of concepts. A relation indicates a logical axiom. For example, $\langle X, \exists r.Y \rangle$ indicates the edge from X to $\exists r.Y$ and it corresponds to the axiom $X \sqsubseteq \exists r.Y$. $\langle Y, \neg Z \rangle$ indicates the edge “ Y disjoint Z ” and corresponds to the axiom $Y \sqsubseteq \neg Z$. These patterns are illustrative to introduce our approach and thus they are not exhaustive.

It should be noted that a MUPS may contain several different patterns and also the same pattern can appear in a MUPS for many times. For example, in the MUPS $\{A \sqsubseteq \exists P.B, B \sqsubseteq C, B \sqsubseteq D, C \sqsubseteq \neg D\}$, the Isa-Disjoint pattern and the Exist-Bottom pattern are involved. In the MUPS $\{A \sqsubseteq \exists P_1.B, B \sqsubseteq \exists P_2.C, C \sqsubseteq \perp\}$, the Exist-Bottom pattern has appeared twice.

3 Approach to Debugging Ontologies Based on Patterns

Based on various patterns, how to make use of these patterns to find MUPS is an extremely important problem. In this section, we propose a novel debugging approach to automatically computing a set of MUPS for an unsatisfiable concept by applying a set of heuristic strategies based on the patterns. Before presenting our approach, the task of data preprocessing needs to be performed.

3.1 Data Preprocessing

Data preprocessing consists of the following steps: normalizing an ontology and constructing a directed graph of the concept and property hierarchies.

Normalizing an ontology is to remove those nested descriptions and make the axioms as small and flat as possible, which makes the axioms in an ontology suitable to apply our heuristic strategies. To normalize an ontology, we borrow

the idea of structural transformation used in [6] and originally defined in [9]. The normalization can be done by introducing new concepts. Specifically, for the axioms in the forms of $C \sqsubseteq D$, $C \sqsubseteq \exists P.D$, $C \sqsubseteq \forall P.D$, $C \sqsubseteq \geq nP.D$, $C \sqsubseteq \leq nP.D$, the normalization step is to make them contain only one atomic concept in the right side of an axiom. For the axiom in the form of $C \sqsubseteq D_1 \sqcup D_2$, it makes the axiom contain only two atomic concepts in the right side.

Note that we differentiate the unsatisfiable concepts in the original ontology O with those that are newly added concepts when normalizing O . The former concepts are found by applying a standard DL reasoner and the latter ones are computed by using some heuristic strategies. For instance, a novel concept is unsatisfiable if it has an existential restriction whose filler is unsatisfiable.

Constructing a directed graph is to obtain those implicit relations which will be used when detecting and expanding a pattern. Such a graph consists of two sub-graphs for the concept and property hierarchies respectively. To construct the sub-graph w.r.t. the concept hierarchy, we only consider those subsumed relations between pairs of atomic concepts which are explicitly included in a normalized ontology O_N . In this sub-graph, an edge from vertex A to vertex B indicates that B is a direct super concept of A . Similarly, we can construct a sub-graph according to the property hierarchy by considering those explicitly declared subsumed relations between pairs of atomic properties.

Example 1. Assume we have an ontology O consisting of the following axioms:

$$A \sqsubseteq \exists P1.(B \sqcap \exists P2.C), \quad C \sqsubseteq \exists P3.D \sqcap \forall P3.E, \quad E \sqsubseteq F, \quad D \sqsubseteq \neg F$$

In O , A and C are unsatisfiable. After normalizing this ontology, we can obtain the normalized ontology O_N :

$$\begin{array}{llll} A \sqsubseteq \exists P1.N, & N \sqsubseteq \exists P2.C, & N \sqsubseteq B, & C \sqsubseteq \exists P3.D \\ C \sqsubseteq \forall P3.E, & E \sqsubseteq F, & D \sqsubseteq \neg F. & \end{array}$$

Here, N is a freshly added concept. According to our heuristic strategy, the novel concept N is also unsatisfiable as we have $N \sqsubseteq \exists P2.C$ where C is unsatisfiable. As for the mapping \mathcal{M}_N , we take the following example. $N \sqsubseteq \exists P2.C$ in O_N can be represented as $\langle N, \exists P2.C \rangle$ and mapped to $A \sqsubseteq \exists P1.(B \sqcap \exists P2.C)$ in O .

3.2 Pattern-Based Debugging Approach

In Algorithm 1, vc is a global variable which is initialized by an empty set. To compute MUPS of C , findMUPS may be invoked for several times. Thus we use vc to avoid the case that this method takes a concept as input more than once as this will cause a dead circle. findMUPS invokes each concrete method like $\text{findMUPS_Isa-Disjoint}(C)$ to instantiate a pattern. We only provide the algorithm to detect the Exist-All pattern as others can be instantiated similarly.

Algorithm 2 is the algorithm to instantiate the Exist-All pattern. In the algorithm, \mathcal{M}_{exists_C} (resp. \mathcal{M}_{all_C}) is a mapping which maps an ancestor of C (a concept can be an ancestor of itself) to the set of its direct super conditions, each of which is defined with an existential (resp. universal) restriction.

Algorithm 1: findMUPS(C)**Data:** An unsatisfiable concept C in an ontology O **Result:** A set of MUPS MU of C

```

1 begin
2   if  $C \in vc$  then
3     return null;
4    $vc \leftarrow vc \cup \{C\}$ ;
5    $MU \leftarrow \text{findMUPS\_Isa-Disjoint}(C)$ ;
6    $MU \leftarrow MU \cup \text{findMUPS\_Exist-Bot}(C)$ ;
7    $MU \leftarrow MU \cup \text{findMUPS\_Exist-All}(C)$ ;
8    $MU \leftarrow MU \cup \text{findMUPS\_Exist-Domain}(C)$ ;
9   return  $MU$ ;
10 end

```

Algorithm 2: findMUPS_Exist-All(C)**Data:** An unsatisfiable concept C in an ontology O **Result:** A set of MUPS MU of C based on the Exist-All pattern

```

1 begin
2    $MU \leftarrow \emptyset$ ;
3   for  $\langle C_1, cond_1 \rangle \in \mathcal{M}_{exist_C}$  do
4     for  $\langle C_2, cond_2 \rangle \in \mathcal{M}_{all_C}$  do
5       if property in  $cond_1 \neq$  property in  $cond_2$  then
6         continue;
7        $S_{disj} \leftarrow \text{findDisjRelations}(filler\ in\ cond_1, filler\ in\ cond_2)$ ;
8       if  $S_{disj} = \emptyset$  then
9         continue;
10      for  $\langle A, \neg B \rangle \in S_{disj}$  do
11         $\mathcal{P}_{expand} \leftarrow \{\langle filler\ in\ cond_1, A \rangle, \langle filler\ in\ cond_2, B \rangle, \langle C, C_1 \rangle, \langle C, C_2 \rangle\}$ ;
12         $combPaths \leftarrow \emptyset$ ;
13        for  $\langle D_1, D_2 \rangle \in \mathcal{P}_{expand}$  do
14           $paths \leftarrow \text{findPaths}(D_1, D_2)$ ;
15           $combPaths \leftarrow \text{combine}(combPaths, paths)$ ;
16         $MU_t \leftarrow \text{combine}(combPaths, \{\{\langle C_1, cond_1 \rangle, \langle C_2, cond_2 \rangle, \langle A, \neg B \rangle\}\})$ ;
17         $MU \leftarrow MU \cup \text{minimize}(t(MU_t))$ ;
18   return  $MU$ ;
19 end

```

Algorithm 2 iterates on \mathcal{M}_{exist_C} and \mathcal{M}_{all_C} to find each pair of restrictions which share the same property. If found, we then compute the disjoint relations between the two fillers in the two restrictions by using the method `findDisjRelations` (see below). If we fail to find such kind of disjoint relations, it shows that this pair of restrictions cannot form any instantiated pattern. In such case, another pair of restrictions will be checked. For each found disjoint relation like $\langle A, \neg B \rangle$, several pairs (see Line 11) of atomic concepts like $\langle C, C_1 \rangle$ need to be further instantiated in the directed graph G of the hierarchies. It is because, for such a pair to be expanded, we only know the second concept in the pair is a super concept of the first one (possibly the two concepts are same), we do not know the specific reason. To instantiate each such kind of subsumed relation, the method `findPaths` will be invoked to search a set of paths in the graph G (see below). As different sets of paths belong to different parts of an instantiated pattern, they should be combined with each other (see lines 12-16). Then each found set of axioms by translating the set of paths with the function `t` (see below) needs to be minimized to form a set of MUPS.

$\text{findDisjRelations}(A, B)$ is to find disjoint relations by iterating on the ancestors of A and those of B . If one ancestor of A is disjoint with the other ancestor of B , then we know A is also disjoint with B . In this way, we can find those disjoint relations between A and B . $\text{findPaths}(D_1, D_2)$ is to find the paths from a starting vertex D_1 to an ending vertex D_2 by applying the depth-first search strategy in a directed graph of hierarchies. $\text{combine}(\mathcal{S}_1, \mathcal{S}_2)$ combines two sets of sets. Namely, $\{S_1 \cup S_2 \mid S_1 \in \mathcal{S}_1 \text{ and } S_2 \in \mathcal{S}_2\}$. If \mathcal{S} is a set of sets of (possibly complex) concept pairs, $\text{t}(\mathcal{S})$ is to translate each set in \mathcal{S} to a set of logical axioms according to the mapping \mathcal{M}_N . If \mathcal{S} is a set of sets of axioms, the method $\text{minimize}(\mathcal{S})$ means to check each axiom in $S \in \mathcal{S}$ one by one. If removing an axiom does not influence the unsatisfiability of the input concept C in S , then remove it from S . In this way, we can find MUPS of C .

Example 2. For A in Example 1, the Exist-Bottom pattern is detected in $\text{findMUPS_Exist-Bot}$ as $A \sqsubseteq \exists P1.N$ where N is unsatisfiable in O_N . To explain why N is unsatisfiable, findMUPS needs to be invoked again. Then the Exist-Bottom pattern is detected again for N as $N \sqsubseteq \exists P2.C$ where C is unsatisfiable.

For C , the Exist-All pattern is detected. It is because C has two super conditions $\exists P3.D$ and $\forall P3.E$ which share the same property $P3$ and there is a disjoint relation $\langle D, \neg F \rangle$ between D and E (see Line 7 in Alg. 2). For this relation, we have $\mathcal{P}_{\text{expand}} = \{\langle D, D \rangle, \langle E, F \rangle, \langle C, C \rangle\}$. After expanding the pairs, we obtain $\text{combPaths} = \{\{\langle E, F \rangle\}\}$. Then we have $\text{t}(MU_t) = \{\{C \sqsubseteq \exists P3.D \sqcap \forall P3.E, E \sqsubseteq F, D \sqsubseteq \neg F\}\}$. After minimizing, we know MU_t is a MUPS of C .

When going back to the method $\text{findMUPS_Exist-Bot}$, the axiom $A \sqsubseteq \exists P1.(B \sqcap \exists P2.C)$ will be added to each found MUPS by $\text{findMUPS_Exist-All}(C)$. Finally, we obtain one MUPS for A which contains all axioms in O .

Property 1. Our debugging approach findMUPS is sound but not complete.

It is straightforward to prove this property based on our method findMUPS .

4 Experimental Evaluation

Our approach 1 was implemented with OWL API 3.1.0 and the standard reasoning tasks are performed using Pellet 2.2.2. The experiments were performed on a computer with 2.99 GHz Intel(R) Core(TM)2 Duo CPU and 2.00 GB of RAM using Windows XP. Sun's Java 1.6.0 was used for Java-based tools and the maximum heap space was set to 1GB. The maximal time limit is 30 minutes.

4.1 Data Set

The ontologies in the first dataset are learned by applying the ontology learning framework Text2Onto 2 on a text corpus which consists of the abstracts from the "knowledge management" information space of the BT Digital Library. To obtain processable data, we choose two sub-ontologies (marked as `bt_km-3000`

¹ <http://research.aturstudio.com/Reasoning/debugPatterns.zip>

and `bt_km-5000`) for our test which are obtained by randomly choose 3000 and 5000 axioms from the original ontology containing 12040 axioms respectively.

The second dataset includes two biomedical ontologies: the galen medical knowledge base (Galen) and the gene ontology (Go). To make them incoherent, for each of them about 500 disjoint relations are added between pairs of sibling concepts. Two concepts are siblings if they share a direct super concept. We mark them as `Galen-inco` (contains 4569 axioms) and `Go-inco` (contains 29375 axioms) respectively. Both of them are $\mathcal{EL}++$ ontologies.

4.2 Evaluation Results

Our approach is evaluated w.r.t. the efficiency and completeness by comparing with a representative debugging approach (i.e. the glass-box approach in [7]). To make the glass-box approach achieve better performance, we extract modules by applying the “top of bottom” approach² proposed in [11] (marked as `Module_top-bot`). Our approach is based on the entire ontologies. The efficiency is measured by the time to finish the process to find MUPS which includes the time to output each found MUPS. The completeness is measured by the ratio of the number of MUPS found by our approach to the number of MUPS found by the glass-box approach. For each ontology, four unsatisfiable concepts are randomly selected: The extracted modules of the selected concepts contain no more than 400, 60, 45 and 125 axioms for `Galen-inco`, `Go-inco`, `bt_km-3000` and `bt_km-5000` respectively.

UC	<code>bt_km-3000</code>	<code>bt_km-5000</code>	<code>Galen-inco</code>	<code>Go-inco</code>
C1	<code>application_design</code>	<code>application_design</code>	<code>AreaOfAtrophicGastritis</code>	<code>GO_0006588</code>
C2	<code>management</code>	<code>hardware_system</code>	<code>InflammationOfStomach</code>	<code>GO_0042421</code>
C3	<code>network_technology</code>	<code>management</code>	<code>KneeJointCavity</code>	<code>GO_0042427</code>
C4	<code>term</code>	<code>term</code>	<code>RenalAbscess</code>	<code>GO_0042429</code>

Efficiency: For each selected concept in all ontologies except `bt_km-5000`, our approach can finish the debugging process within 1 second. Even for a concept in `bt_km-5000` which contains quite a lot MUPS (e.g. at least 16719 MUPS for `C2`), it takes about 1000 seconds. As for `Module_top-bot`, it cannot finish the process within the limited time for 10 out of 16 concepts. Overall, comparing `Module_top-bot` with our approach, the efficiency has been improved at most 1446 times for `C3` in `bt_km-3000` and about 319 times in average over all selected concepts except those which cannot be processed by `Module_top-bot`. The experiment shows the merit of adopting patterns to guide the process of computing MUPS.

Completeness: For `bt_km-3000`, all MUPS can be found by both approaches. For the concepts `C2` and `C4` in `Galen-inco`, `Module_top-bot` can find all MUPS and our approach can find more than 84% MUPS. As for other selected concepts in ontologies `Galen-inco`, `Go-inco` and `bt_km-5000`, `Module_top-bot` cannot finish the process to compute all MUPS within the limited time, but our approach can

² The implementation of this approach is available in OWL API.

find quite a lot MUPS within a short time. For example, for C2 in the ontology `bt_km-5000`, 16719 MUPS are found within 971 seconds. In a word, this reflects that our approach returns MUPS with high completeness and is suitable to deal with those ontologies that contain many MUPS.

5 Conclusion and Future Work

To improve the efficiency to debug an ontology, we proposed a pattern-based approach by using a set of heuristic strategies instead of invoking a classical DL reasoner frequently. Comparing with the glass-box approach based on modules, the efficiency has been improved at most 1446 times and about 319 times in average. Besides, at least 84% MUPS have been found for each selected concept. It shows the advantage of adopting the heuristic strategies based on patterns.

In the future, we will use the hitting set tree algorithm [10] to find all MUPS of a concept based on those found by our approach. Besides, more patterns will be integrated into our approach.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press (2003)
2. Cimiano, P., Völker, J.: *Text2onto – A Framework for Ontology Learning and Data-Driven Change Discovery*. In: Montoyo, A., Muñoz, R., Métails, E. (eds.) *NLDB 2005*. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
3. Corcho, Ó., Roussey, C., Blázquez, L.M.V., Pérez, I.: *Pattern-based owl ontology debugging guidelines*. In: *WOP (2009)*
4. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: *Inconsistencies, negations and changes in ontologies*. In: *AAAI*, Boston, Massachusetts, pp. 1295–1300 (2006)
5. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: *Just the right amount: extracting modules from ontologies*. In: *WWW*, pp. 717–726 (2007)
6. Horridge, M., Parsia, B., Sattler, U.: *Laconic and Precise Justifications in Owl*. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
7. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: *Finding all Justifications of OWL DL Entailments*. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
8. Parsia, B., Sirin, E., Kalyanpur, A.: *Debugging owl ontologies*. In: *WWW*, pp. 633–640 (2005)
9. Plaisted, D.A., Greenbaum, S.: *A structure-preserving clause form translation*. *Journal of Symbolic Computation* (1986)
10. Reiter, R.: *A theory of diagnosis from first principles*. *Artificial Intelligence* 32(1), 57–95 (1987)

11. Sattler, U., Schneider, T., Zakharyashev, M.: Which kind of module should i extract? In: *Description Logics* (2009)
12. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: *IJCAI*, pp. 355–362 (2003)
13. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P.: A Modularization-Based Approach to Finding All Justifications for OWL DL Entailments. In: Domingue, J., Anurariya, C. (eds.) *ASWC 2008*. LNCS, vol. 5367, pp. 1–15. Springer, Heidelberg (2008)
14. Wang, H., Horridge, M., Rector, A.L., Drummond, N., Seidenberg, J.: Debugging OWL-DL Ontologies: A Heuristic Approach. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) *ISWC 2005*. LNCS, vol. 3729, pp. 745–757. Springer, Heidelberg (2005)

Author Index

Abel, Fabian 160
Ashraf, Jamshaid 376

Bai, Xi 318
Bai, Yin 385
Beauregard, Bill 402
Bertault, Francois 411

Celino, Irene 128
Chao, Jiansong 268
Chen, Huajun 258
Chen, Jianfeng 350
Cheng, Gong 226
Chute, Christopher G. 342
Cobby, Nathan 417

d'Aquin, Mathieu 284
Davis, Hugh C. 210
Dell'Aglio, Daniele 128
Du, Jianfeng 144, 394

Fang, Jun 1
Feng, Wendy 411
Finin, Tim 334
Fionda, Valeria 64

Gao, Qi 160
Gao, Zhiqiang 425
Ge, Weiyi 226
Gibbins, Nicholas 242
Groza, Tudor 300
Gu, Peiqin 258

Hadzic, Maja 376
Haller, Armin 300
Han, Lushan 334
Hihara, Keisuke 96
Horne, Ross 242
Houben, Geert-Jan 160
Hu, Wei 48, 350
Hu, Yong 144
Huang, Zhisheng 1, 425

Ichise, Ryutaro 112

Ji, Feng 226
Ji, Qiu 425
Joshi, Anupam 334

Kim, Hong-Gee 358
Kim, Hyoung-Joo 33
Kim, Kisung 33
Klein, Ewan 318
Kozaki, Kouji 96
Krustins, Austris 411

Lee, Kevin 17
Li, Juanzi 80
Li, Zonghui 385
Luo, Shengmei 226

Ma, Yuanchao 385
Mizoguchi, Riiciro 96
Moon, Bongki 33
Motta, Enrico 284
Mueller, Ralf 402
Musen, Mark A. 342

Namgoong, Hyun 358
Nikolov, Andriy 284
Noy, Natalya F. 342

Pan, Jeff Z. 17, 80, 144, 394
Pirr , Giuseppe 64
Prater, Jean 402

Qi, Guilin 144, 394
Qiu, Che 394
Qu, Yuzhong 48, 226, 350

Ren, Yuan 17
Robertson, Dave 318
Rosenberg, Florian 300

Sassone, Vladimiro 242
Shah, Nigam H. 342
Shen, Yidong 366
Solbrig, Harold R. 342
Sotomayor, Marco 366, 417

- Tao, Cui 342
Thornton, John 366
Tian, Yuan 176
Tiropanis, Thanassis 210
- Umbrich, Jürgen 176
- van Harmelen, Frank 1
Verza, Arturs 411
- Wang, Haofen 268
Wang, Kewen 366, 417
Wang, Shuai 144, 394
Wang, Xin 210
Wang, Zhe 417
Wang, Zhichun 80
- Wang, Zhigang 80
Wu, Gang 192
- Xu, Bin 385
- Yang, Mengdong 192
Yang, Sungkwon 358
Yi, Liangrong 411
Yu, Yong 176, 268
- Zhang, Hang 48, 350
Zhang, Weinan 268
Zhao, Lihua 112
Zhou, Wenlei 268
Zhu, Man 425
Zong, Nansu 358