

**Philipp Cimiano Miriam Fernández
Vanessa Lopez Stefan Schlobach
Johanna Völker (Eds.)**

LNCS 7955

The Semantic Web: ESWC 2013 Satellite Events

**ESWC 2013 Satellite Events
Montpellier, France, May 2013
Revised Selected Papers**



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

Philipp Cimiano Miriam Fernández
Vanessa Lopez Stefan Schlobach
Johanna Völker (Eds.)

The Semantic Web: ESWC 2013 Satellite Events

ESWC 2013 Satellite Events
Montpellier, France, May 26-30, 2013
Revised Selected Papers



Springer

Volume Editors

Philipp Cimiano
University of Bielefeld, CITEC, Germany
E-mail: cimiano@cit-ec.uni-bielefeld.de

Miriam Fernández
Open University, Milton Keynes, UK
E-mail: m.fernandez@open.ac.uk

Vanessa Lopez
IBM Research - Ireland, Dublin, Ireland
E-mail: vanlopez@ie.ibm.com

Stefan Schlobach
VU University Amsterdam, The Netherlands
E-mail: k.s.schlobach@vu.nl

Johanna Völker
University of Mannheim, Germany
E-mail: voelker@informatik.uni-mannheim.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-41241-7 e-ISBN 978-3-642-41242-4
DOI 10.1007/978-3-642-41242-4
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013949029

CR Subject Classification (1998): H.3, H.4, I.2, H.5, H.2, D.2

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

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, 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.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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

This volume contains the post-proceedings of the ESWC 2013 conference, which took place in Montpellier, France, from during 26h–30th, 2013.

The conference was without doubt a great success and attracted a total number of 300 participants. Besides the presentations given in the research tracks ¹, the conference featured a rich and broad tutorial and workshop program consisting of 11 workshops and 7 tutorials.

A short paper by the tutorial and workshop chairs providing an overview of the tutorial workshop program is included in this volume.

Further, 20 posters and 24 demonstrations were presented at the poster and demonstration sessions, respectively. This volume contains the 44 papers describing the posters and demonstrations. While demonstration papers are up to 5 pages long, poster papers are shorter with 2 pages maximum. A short overview of the demonstration and poster session is also included in this volume.

Further, the volume also contains the 10 best workshop papers, which were selected by the workshop chair and the general chair on the basis of their reviews and nominations by the workshop chairs. The authors of these workshop papers also received the opportunity to do a lightning presentation on their work at the last day of the conference.

Finally, the volume contains four papers of the AI Mashup Challenge ². They were peer-reviewed after the corresponding systems had been presented at the conference in the AI Mashup Challenge track.

We hope you enjoy the volume!

Philipp Cimiano
Miriam Fernández
Vanessa Lopez
Stefan Schlobach
Johanna Völker

¹ For those interested in the lectures of the research tracks, recordings can be found at http://videlectures.net/eswc2013_montpellier/.

² <http://aimashup.org/aimashup13/doku.php>

Organization

Organizing Committee

General Chair

Philipp Cimiano Bielefeld University, Germany

Program Chairs

Oscar Corcho UPM, Spain
Valentina Presutti Institute of Cognitive Sciences and
 Technologies, Italy

Local Chairs

Clement Jonquet LIRMM, Université Montpellier 2, France
François Scharffe LIRMM, Université Montpellier 2, France

Poster and Demo Chairs

Vanessa Lopez IBM, Ireland
Miriam Fernandez Open University, UK

Workshops Chair

Johanna Völker University of Mannheim, Germany

Tutorials Chair

Stefan Schlobach VU University of Amsterdam, The Netherlands

PhD Symposium Chairs

Laura Hollink VU University of Amsterdam, The Netherlands
Sebastian Rudolph Karlsruhe Institute of Technology, Germany

Semantic Technologies Coordinators

Dieter Fensel STI Innsbruck, Austria
Birgit Leiter STI Innsbruck, Austria
Alex Oberhauser STI Innsbruck, Austria
Cord Wiljes Bielefeld University, Germany

Project Networking Session Chair

Achim Rettinger Karlsruhe Institute of Technology, Germany

Sponsorship Chair

Axel Ngonga University of Leipzig, Germany

Publicity Chair

Fabien Gandon Inria, France

Panel Chair

Marko Grobelnik Jozef Stefan Institute Ljubljana, Slovenia

Proceedings Chair

Katja Temnow Bielefeld University, Germany

Treasurer

Alexander Wahler STI, Germany

Local Organization and Conference Administration

STI International Austria

Webmaster

STI International Austria

Program Committee

Program Chairs

Oscar Corcho UPM, Spain
Valentina Presutti Institute of Cognitive Sciences and
Technologies, Italy

Track Chairs

Ontologies Track

Eva Blomqvist Linköping University, Sweden
Aldo Gangemi LIPN-Paris 13-Sorbonne Cité, France and
ISTC-CNR, Italy

Reasoning Track

Pascal Hitzler	Wright State University Dayton at Ohio, USA
Luciano Serafini	FBK, Italy

Semantic Data Management Track

María Esther Vidal	Universidad Simón Bolívar, Venezuela
Axel Polleres	Siemens AG, Austria

Linked Open Data Track

Jun Zhao	University of Oxford, UK
Jens Lehmann	University of Leipzig, Germany

Social Web and Web Science Track

Marta Sabou	MODUL University Vienna, Austria
Andreas Hotho	University of Würzburg, Germany

Natural Language Processing and Information Retrieval Track

Alfio Gliozzo	IBM Watson Research Center, USA
Malvina Nissim	University of Bologna, Italy

Mobile Web, Sensors and Semantic Streams Track

Josiane Parreira	DERI, Ireland
Payam Barnaghi	University of Surrey, UK

Machine Learning Track

Claudia d'Amato	University of Bari, Italy
Dunja Mladenic	Jozef Stefan Institute Ljubljana, Slovenia

Special Track: Cognition and Semantic Web

Krzysztof Janowicz	University of California at Santa Barbara, USA
Kai-Uwe Kühnberger	University of Osnabrück, Germany

In-Use and Industrial Track

Sofia Angeletou	BBC, UK
José Manuel Gomez-Pérez	iSOCO, Spain

Workshop Program Committee

Chris Bizer	University of Mannheim, Germany
Dieter Fensel	University of Innsbruck, Austria
Aldo Gangemi	ISTC-CNR, Italy
Asuncion Gomez-Perez	Universidad Politécnica de Madrid, Spain
Frank van Harmelen	VU University Amsterdam, The Netherlands
Pascal Hitzler	Wright State University, Dayton, USA
Enrico Motta	The Open University, UK

Tutorial Program Committee

Avi Bernstein	University of Zurich, Switzerland
Carole Goble	University of Manchester, UK
Christophe Gueret	DANS, The Netherlands
Craig Knoblock	University of Southern California, USA
Pascal Hitzler	Wright State University, Dayton, USA
Sren Auer	University of Leipzig, Germany
Tom Heath	Talis, UK

AI Mashup Challenge Program Committee

Christoph Lange	University of Birmingham, UK
Jevon Wright	Massey University, New Zealand
Heiko Paulheim	University of Mannheim, Germany
Olexiy Chudnovskyy	Chemnitz University of Technology, Germany
Michalis Stefanidakis	Ionian University, Greece
Thorsten Liebig	University of Ulm & derivo GmbH, Germany
Ikuya Yamada	Studio Ousia Inc., Japan
Emilian Pascalau	CNAM, France
Emanuele Della Valle	Politecnico di Milano, Italy
Charalampos Bratsas	OKFN, Greece
Felix Burkhardt	Telekom Innovation, Germany
Deborah Dahl	World Wide Web Consortium, USA
Bill Scholz	NewSpeech, USA
Luis M. Vilches Blázquez	Universidad Politécnica de Madrid, Spain

Demonstration and Poster Program Committee

Carlos Pedrinaci	The Open University, UK
Aba-Sah Dadzie	The University of Sheffield, UK
Andrés Garcia-Silva	Universidad Politécnica de Madrid, Spain
Andriy Nikolov	The Open University, UK
Basil Ell	Institute AIFB, Germany
Carlo Alloca	FORTH-ICS, Greece
Claudia Wagner	Joanneum Research, Austria
David Vallet	Universidad Autónoma de Madrid, Spain
Denny Vrandecic	Wikimedia Deutschland, Germany
Elena Cabrio	Inria, France
Elisabeth Cano	The Open University, UK
Fadi Maali	Digital Enterprise Research Institute, Ireland
Fouad Zablith	American University of Beirut, Lebanon
Hassan Saif	The Open University, UK
Ivn Cantador	Universidad Autónoma de Madrid, Spain
Jorge Gracia	Universidad Politécnica de Madrid, Spain
Elena Montiel-Posada	Universidad Politécnica de Madrid, Spain
Liliana Cabral	CSIRO, Australia
Marco Luca Sbodio	IBM Research, Ireland
Mari Carmen Suarez-Figueroa	Universidad Politécnica de Madrid, Spain
Maria Maleshkova	Karlsruhe Institute of Technology, Germany
Marta Sabou	MODUL University Vienna, Austria
Matthew Rowe	Lancaster University, UK
Raul Palma	Poznan Supercomputing and Networking Center, Poland
Sam Chapman	Knowledge Now Limited, UK
Sofia Angeletou	BBC, UK
Stefan Dietze	L3S Research Center, Germany
Stuart Wrigley	University of Sheffield, UK
Tania Tudorache	Stanford University, USA
Victoria Uren	The University of Aston, UK
Ziqi Zhang	University of Sheffield, UK

Steering Committee

Chair

John Domingue

The Open University, UK and STI
International, Austria

Members

Grigoris Antoniou

Forth, Greece

Lora Aroyo

VU University of Amsterdam, The Netherlands

Fabio Ciravegna

University of Sheffield, UK

Marko Grobelnik

JSI, Slovenia

Eero Hyvnen

Aalto University, Finland

Axel Polleres

Siemens AG, Austria

Elena Simperl

University of Southampton, UK

Paolo Traverso

IRST, Italy

Sponsoring Institutions





ONTOLOGOS CORP.
WWW.ONTOLOGOS-CORP.COM



SIFR project





VPH-Share

YAHOO! LABS

Local Sponsors



Table of Contents

Best Workshop Papers

A Summary of the Workshop and Tutorial Program at ESWC 2013	1
<i>Johanna Völker and Stefan Schlobach</i>	
ParlBench: A SPARQL Benchmark for Electronic Publishing Applications	5
<i>Tatiana Tarasova and Maarten Marx</i>	
I See a Car Crash: Real-Time Detection of Small Scale Incidents in Microblogs	22
<i>Axel Schulz, Petar Ristoski, and Heiko Paulheim</i>	
Ontology Adaptation upon Updates	34
<i>Alessandro Solimando and Giovanna Guerrini</i>	
Caching and Prefetching Strategies for SPARQL Queries	46
<i>Johannes Lorey and Felix Naumann</i>	
Characterising Citations in Scholarly Documents: The CiTalO Framework	66
<i>Angelo Di Iorio, Andrea Giovanni Nuzzolese, and Silvio Peroni</i>	
YASGUI: Not Just Another SPARQL Client	78
<i>Laurens Rietveld and Rinke Hoekstra</i>	
A Case-Study of Ontology-Driven Semantic Mediation of Flower-Visiting Data from Heterogeneous Data-Stores in Three South African Natural History Collections	87
<i>Willem Coetzer, Deshendran Moodley, and Aurore Gerber</i>	
Mapping Keywords to Linked Data Resources for Automatic Query Expansion	101
<i>Isabelle Augenstein, Anna Lisa Gentile, Barry Norton, Ziqi Zhang, and Fabio Ciravegna</i>	
Finding Fault: Detecting Issues in a Versioned Ontology	113
<i>Maria Copeland, Rafael S. Gonçalves, Bijan Parsia, Uli Sattler, and Robert Stevens</i>	

Optique: Towards OBDA Systems for Industry	125
<i>Eugeny Kharlamov, Ernesto Jiménez-Ruiz, Dmitriy Zheleznyakov,</i>	
<i>Dimitris Bilidas, Martin Giese, Peter Haase, Ian Horrocks,</i>	
<i>Herald Kllapi, Manolis Koubarakis, Özgür Özçep,</i>	
<i>Mariano Rodríguez-Muro, Riccardo Rosati, Michael Schmidt,</i>	
<i>Rudolf Schlatte, Ahmet Soylu, and Arild Waaler</i>	

Demonstration Session

Summary of the Demonstration and Poster Track	141
<i>Miriam Fernández and Vanessa López</i>	

LODatio: A Schema-Based Retrieval System for Linked Open Data at Web-Scale	142
<i>Thomas Gottron, Ansgar Scherp, Bastian Kraye, and Arne Peters</i>	

<i>Payola</i> : Collaborative Linked Data Analysis and Visualization Framework	147
<i>Jakub Klímek, Jiří Helmich, and Martin Nečaský</i>	

A System for Aligning Taxonomies and Debugging Taxonomies and Their Alignments	152
<i>Valentina Ivanova and Patrick Lambrix</i>	

ALASKA for Ontology Based Data Access	157
<i>Jean-François Baget, Madalina Croitoru, and</i>	
<i>Bruno Paiva Lima da Silva</i>	

Multilingual MoKi: How to Manage Multilingual Ontologies in a Wiki	162
<i>Mauro Dragoni, Chiara Ghidini, and Alessio Bosca</i>	

SAIM – One Step Closer to Zero-Configuration Link Discovery	167
<i>Klaus Lyko, Konrad Höffner, René Speck,</i>	
<i>Axel-Cyrille Ngonga Ngomo, and Jens Lehmann</i>	

Linked Data Query Wizard: A Tabular Interface for the Semantic Web	173
<i>Patrick Hoefler, Michael Granitzer, Vedran Sabol, and</i>	
<i>Stefanie Lindstaedt</i>	

Facilitating Music Information Research with Shared Open Vocabularies	178
<i>Alo Allik, György Fazekas, Simon Dixon, and Mark Sandler</i>	

Exploratory Search on the Top of DBpedia Chapters with the Discovery Hub Application	184
<i>Nicolas Marie, Fabien Gandon, Damien Legrand, and</i>	
<i>Myriam Ribière</i>	

Applying SPARQL-DQP for Federated SPARQL Querying over Google Fusion Tables	189
<i>Freddy Priyatna, Carlos Buil Aranda, and Oscar Corcho</i>	
Querying Multilingual DBpedia with QAKiS	194
<i>Elena Cabrio, Julien Cojan, Fabien Gandon, and Amine Hallili</i>	
LDtogo: A Data Querying and Mapping Framework for Linked Data Applications	199
<i>Niels Ockeloen, Victor de Boer, and Lora Aroyo</i>	
Exploring the Linked University Data with Visualization Tools	204
<i>Miika Alonen, Tomi Kauppinen, Osmo Suominen, and Eero Hyvönen</i>	
Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data	209
<i>Charalampos Nikolaou, Kallirroï Dogani, Kostis Kyzirakos, and Manolis Koubarakis</i>	
A Distributional Semantic Search Infrastructure for Linked Dataspaces	214
<i>André Freitas, Seán O’Riain, and Edward Curry</i>	
XSPARQL-Viz: A Mashup-Based Visual Query Editor for XSPARQL	219
<i>Syed Zeeshan Haider Gillani, Muhammad Intizar Ali, and Alessandra Mileo</i>	
LinDA: A Service Infrastructure for Linked Data Analysis and Provision of Data Statistics	225
<i>Nicolas Beck, Stefan Scheglmann, and Thomas Gottron</i>	
Identifying Functions of Citations with CiTalO	231
<i>Angelo Di Iorio, Andrea Giovanni Nuzzolese, and Silvio Peroni</i>	
Graphia: Extracting Contextual Relation Graphs from Text	236
<i>Daniilo S. Carvalho, André Freitas, and João C.P. da Silva</i>	
Cross-Lingual Querying and Comparison of Linked Financial and Business Data	242
<i>Seán O’Riain, Barry Coughlan, Paul Buitelaar, Thierry Declerk, Uli Krieger, and Susan Marie-Thomas</i>	
R2RML by Assertion: A Semi-automatic Tool for Generating Customised R2RML Mappings	248
<i>Luís Eufrazio T. Neto, Vânia Maria P. Vidal, Marco A. Casanova, and José Maria Monteiro</i>	

Tipalo: A Tool for Automatic Typing of DBpedia Entities 253
*Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti,
 Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini*

Tracking and Analyzing The 2013 Italian Election 258
*Vuk Milicic, José Luis Redondo García, Giuseppe Rizzo, and
 Raphaël Troncy*

FRED: From Natural Language Text to RDF and OWL in One
 Click 263
*Francesco Draicchio, Aldo Gangemi, Valentina Presutti, and
 Andrea Giovanni Nuzzolese*

Poster Session

Trusted Facts: Triplifying Primary Research Data Enriched
 with Provenance Information 268
*Kai Schlegel, Sebastian Bayerl, Stefan Zwicklbauer,
 Florian Stegmaier, Christin Seifert, Michael Granitzer, and
 Harald Kosch*

Semantic Hyperlocal Search for Parlance Mobile Spoken Dialogue
 System 271
*Panos Alexopoulos, Marie-Aude Aufaure, Nesrine Ben-Mustapha,
 Hugues Bouchard, José Manuel Gomez-Pérez, James Henderson,
 Beibei Hu, Joel Lang, Peter Mika, and Yves Vanrompay*

Representation of Complex Expressions in RDF 273
Sébastien Ferré

Representing and Querying Negative Knowledge in RDF 275
Fariz Darari

The Semantic Evolution of General and Specific Communities 277
Matthew Rowe and Claudia Wagner

Experiments Varying Semantic Similarity Measures and Reference
 Ontologies for Ontology Alignment 279
Valerie Cross, Pramit Silwal, and Xi Chen

Market-Based SPARQL Brokerage: Towards Economic Incentives
 for Linked Data Growth 282
Mengia Zollinger, Cosmin Basca, and Abraham Bernstein

A Shared Vocabulary for Audio Features 285
Alo Allik, György Fazekas, Simon Dixon, and Mark Sandler

Exploratory Search on the Top of DBpedia Chapters with the Discovery Hub Application	287
<i>Nicolas Marie, Fabien Gandon, and Myriam Ribière</i>	
The Finnish Law as a Linked Data Service	289
<i>Matias Frosterus, Jouni Tuominen, Mika Wahlroos, and Eero Hyvönen</i>	
A Distributed Entity Directory	291
<i>Fausto Giunchiglia and Alethia Hume</i>	
Optique: OBDA Solution for Big Data	293
<i>D. Calvanese, Martin Giese, Peter Haase, Ian Horrocks, T. Hubauer, Y. Ioannidis, Ernesto Jiménez-Ruiz, E. Kharlamov, H. Kllapi, J. Klüwer, Manolis Koubarakis, S. Lamparter, R. Möller, C. Neuenstadt, T. Nordtveit, Ö. Özcep, M. Rodriguez-Muro, M. Roshchin, F. Savo, Michael Schmidt, Ahmet Soylu, Arild Waaler, and Dmitriy Zheleznyakov</i>	
Linked Open Ontology Cloud KOKO—Managing a System of Cross-Domain Lightweight Ontologies	296
<i>Matias Frosterus, Jouni Tuominen, Sini Pessala, Katri Seppälä, and Eero Hyvönen</i>	
A Semantic-Enabled Engine for Mobile Social Networks	298
<i>Ronald Chenu-Abente, Ilya Zaihrayeu, and Fausto Giunchiglia</i>	
The Birds of the World Ontology AVIO	300
<i>Jouni Tuominen, Nina Laurence, Mikko Koho, and Eero Hyvönen</i>	
Exploring the Dynamics of Linked Data	302
<i>Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O’Byrne, and Aidan Hogan</i>	
MAD SWAN: A Semantic Web Service Composition System	304
<i>George Markou and Ioannis Refanidis</i>	
Longitudinal Queries over Linked Census Data	306
<i>Albert Meroño-Peñuela, Rinke Hoekstra, Andrea Scharnhorst, Christophe Guéret, and Ashkan Ashkpour</i>	
Deciphering Location Context – A Semantic Web Approach	308
<i>Zhenning Shangguan and Deborah L. McGuinness</i>	
Comparative Classifier Evaluation for Web-Scale Taxonomies Using Power Law	310
<i>Rohit Babbar, Ioannis Partalas, Cornelia Metzigg, Eric Gaussier, and Massih-reza Amini</i>	

Mashup Challenge

NERITS - A Machine Translation Mashup System Using Wikimeta and DBpedia	312
<i>Kamel Nebhi, Luka Nerima, and Eric Wehrli</i>	
SNARC - An Approach for Aggregating and Recommending Contextualized Social Content	319
<i>Ahmad Assaf, Aline Senart, and Raphael Troncy</i>	
Twindex Fuorisalone: Social Listening of Milano during Fuorisalone 2013	327
<i>Marco Balduini, Emanuele Della Valle, Daniele Dell'Aglio, Mikalai Tsytsarau, Themis Palpanas, and Cristian Confalonieri</i>	
DataConf: A Full Client-Side Web Mashup for Scientific Conferences	337
<i>Lionel Médini, Florian Bacle, Benoit Durant de la Pastellière, Fiona Le Peutrec, and Nicolas Armando</i>	
Author Index	345

A Summary of the Workshop and Tutorial Program at ESWC 2013

Johanna Völker¹ and Stefan Schlobach²

¹ Data & Web Science Research Group
Universität Mannheim

² Department of Computer Science
Vrije Universiteit Amsterdam

Abstract. ESWC 2013 was held in Montpellier in May 2013 and the main program was preceded by two days of workshops and tutorials. This short report gives an overview of the 11 workshops and 7 tutorials that were attended by over 170 researchers before the start of the main conference, and describes the procedure adopted to select the proposals.

1 Introduction

The pre-conference program of ESWC 2013 was held on 26th and 27th of May, the two days preceding the main conference program. In the beautiful surroundings of the École Polytechnique of the University of Montpellier 2, more than 170 registered participants spent two days on learning and discussing hot and upcoming topics relevant to the Semantic Web. There were 11 workshops and 7 tutorials, as well as the PhD symposium, the papers of which have appeared already in the main conference proceedings.

Acceptance of workshops and tutorials was decided on the basis of quality and relevance to ESWC. This selection resulted in a broad and varied program that covered many recent and emerging topics in Semantic Web research. There were two tutorials on newly proposed **standards**; one on *the W3C Provenance standard*, the other on the *R2RML and Direct Mapping Standards*. Three events focused on **infrastructure**, with the tutorial on *Semantic Data Management in Graph Databases* and the two workshops on *Benchmarking RDF Systems (BeRSys)* as well as another one on *Services and Applications over Linked APIs and Data (SALAD)*. For quite some time up to now, analysing, mining and visualising have become core topics of interest in the community, and this year's preconference program had a highly visited tutorial on *Analyzing and Visualizing Linked Data with R (LODR2013)* as well as two workshops: *Knowledge Discovery and Data Mining meets Linked Open Data (Know@LOD)* and *Artificial Intelligence meets the Web of Data (AImWD)*. On the crossroad of Natural Language processing and data analysis there was a tutorial on *Cross-language text mining*.

The topics of **ontologies and reasoning** have traditionally had a strong presence at ESWC conferences, and 2013 has been no exception in this respect. With the *OWL: Experiences and Directions (OWLED)* a major two-day

event was organised, in addition to the tutorial on *OWL plus Rules=.. ?* and a shorter workshop on *Debugging Ontologies and Ontology Mappings*. A newer addition to the conference program is work on **the Social Semantic Web** which contributed the workshop *Semantic Web Collaborative Spaces* and the tutorial *Crowdsourcing for the Semantic Web* to the program.

A great feature of this year's program was the upcoming of new **application domains** and work on **applications** of Semantic Technologies, nicely balanced with 4 workshops on *Usage Analysis and the Web of Data (USEWOD)*, *Biodiversity, Social Media and Linked Data for Emergency Response (SMILE)* and *Semantic Publications (SePublica)*.

2 Workshops

The members of the workshop program committee helped us to select the best out of 16 submitted proposals, and to create a rich and diverse workshop program. Five half-day workshops, four full-day workshops were held, as well as OWLED, which was held during both pre-conference days. Together, they offered participants the possibility to learn about state-of-the-art research, engage in lively discussions, and get in contact with colleagues from other institutions:

- AIImWD.** *Artificial Intelligence meets the Web of Data* (Antonis Bikakis, Christophe Guéret, Dino Ienco, Francois Scharffe, Robert Tolksdorf and Serena Villata)
- BeRSys.** *Benchmarking RDF Systems* (Irina Fundulaki, Ioana Manolescu and Ioan Toma)
- BioDiv.** *Semantics for Biodiversity* (Pierre Larmande, Isabelle Mougenot, Clement Jonquet and Therese Libourel)
- OWLED.** *OWL Experiences and Directions* (Mariano Rodríguez-Muro, Simon Jupp and Kavitha Srinivas)
- Know@LOD.** *Knowledge Discovery and Data Mining meets Linked Open Data* (Jens Lehmann, Mathias Niepert, Heiko Paulheim, Harald Sack and Johanna Völker)
- SALAD.** *Services and Applications over Linked APIs and Data* (Maria Maleshkova, Craig Knoblock, Ruben Verborgh, and Steffen Stadtmüller)
- SePublica.** *Semantic Publications* (Alexander Garcia, Christoph Lange, Robert Stevens and Phillip Lord)
- SMILE.** *Social Media and Linked Data for Emergency Response* (Vitaveska Lanfranchi, Suvodeep Mazumdar, Eva Blomqvist and Christopher Brewster)
- SWCS.** *Semantic Web Collaborative Spaces* (Pascal Molli, John Breslin, Hideaki Takeda and Sebastian Schaffert)
- USEWOD.** *Usage Analysis and the Web of Data* (Bettina Berendt, Laura Hollink, Markus Luczak-Rösch, Knud Möller and David Valet)
- WoDOOM.** *Debugging Ontologies and Ontology Mappings* (Patrick Lambrix, Guilin Qi, Matthew Horridge and Bijan Parsia)

The high quality of the individual workshop programs is reflected by the selection of contributions in this volume. From a total of 19 papers proposed as best paper candidates by the workshop organizers, we selected 10 for being included in this ESWC post-proceedings volume. In addition, the authors of these papers received the opportunity to present their work in a lightning presentation at the main conference. The corresponding session was well attended and allowed people who had not been able to attend the pre-conference events to catch a glimpse on what they missed during the first two days.

We thank the members of the committee for their great support (Chris Bizer, Dieter Fensel, Aldo Gangemi, Asuncion Gomez Perez, Frank van Harmelen, Pascal Hitzler and Enrico Motta).

3 Tutorials

The tutorial program committee consisted of 8 senior researchers from different institutions and research areas. Based on recommendations of the program committee, the best proposals were selected. Five half-day and two full-day tutorials were held during both pre-conference days, offering a diverse program centered around some core technologies and methods for Semantic Web research and usage:

OWL Plus Rules = .. ? Organised by David Carral Martinez and Adila Alfa Krisnadhi (Kno.e.sis Center, Wright State University) and Matthias Knorr (CENTRIA, Universidade Nova de Lisboa), this tutorial introduced new developments to combine rule-based and ontology-based paradigms for modeling knowledge on the Semantic Web.

Crowdsourcing for the Semantic Web. This tutorial, presented by Elena Simperl (University of Southampton), Gianluca Demartini (University of Fribourg) and Maribel Acosta (AIFB Karlsruhe) described a selection of state-of-the-art methods in crowdsourcing research and their implementation in various platforms. It also analyzed the motivation and incentive mechanisms that are used in crowdsourcing-driven projects, and introduced guidelines and best practices.

Semantic Data Management in Graph Database. This tutorial, organised by Maria Esther Vidal, Edna Ruckhaus (Universidad Simon Bolivar, Venezuela), Maribel Acosta (Institute AIFB, Karlsruhe Institute of Technology) and Cosmin Basca (University of Zurich, Switzerland), described existing approaches to model graph databases and different techniques implemented in RDF and database engines. It introduced specific graph algorithms and discussed their applicability in the context of the Semantic Web as well as solutions that have been proposed in the context of the Semantic Web to manage large graphs.

RDB2RDF. This tutorial on the new R2RML and Direct Mapping Standards, presented by Juan F. Sequeda (University of Texas at Austin), Barry Norton (The University of Sheffield), Daniel P. Miranker (University of Texas) and

Maria Maleshkova (Karlsruhe Institute of Technology) introduced these. In particular, the suitability of each approach was characterized by both architectural concerns of the larger system, and the skill sets that exist in the supporting organizations.

Cross-Language Text. This tutorial, organised by Marta Ruiz Costa-Jussa (Universitat Politècnica de Catalunya (UPC, Barcelona)) and Patrik Lambert (Le Mans University), covered translation techniques used in several cross-language topics such as cross-language knowledge extraction or opinion mining, focusing on the use of semantic resources or contextual information for machine translation.

Getting to Know PROV - The W3C Provenance Specifications. This tutorial, presented by Paul Groth (VU University Amsterdam), Jun Zhao (University of Oxford) and Olaf Hartig (University of Waterloo), provided an in-depth dive into the new specifications including hands-on information on how to publish, query and access provenance information, and model provenance data using the PROV data model and ontology. It also showed how to produce provenance information that enables integrity checking and inferences, as well as how to expose and acquire provenance information using PROV access mechanisms and services.

LODR2013. This tutorial on Analyzing and Visualizing Linked Data with R 2013 was organised by Tomi Kauppinen (Aalto University) and Willem Robert van Hage (VU University Amsterdam) and introduced the idea and concepts about Linked Science, and showed via illustrative examples how to practically query and analyze Linked Data from within an R environment for visual and statistical analysis.

We thank the members of the program committee: Sören Auer, Avi Bernstein, Carole Goble, Marko Grobelnik, Christophe Guéret, Tom Heath, Pascal Hitzler and Craig Knoblock.

ParlBench: A SPARQL Benchmark for Electronic Publishing Applications*

Tatiana Tarasova and Maarten Marx

ISLA, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands
{t.tarasova,maartenmarx}@uva.nl

Abstract. ParlBench is an RDF benchmark modelling a large scale electronic publishing scenario. The benchmark offers large collections of the Dutch parliamentary proceedings together with information about members of the parliament and political parties. The data is real, but free of intellectual property rights issues. On top of the benchmark data sets several application benchmarks as well as targeted micro benchmarks can be developed. This paper describes the benchmark data sets and 19 analytical queries covering a wide range of SPARQL constructs. The potential use of ParlBench is demonstrated by executing the queries for 8 different scaling of the benchmark data sets on Virtuoso RDF store. Measured on a standard laptop, data loading times varied from 43 seconds (for 1% of the data set) to 48 minutes (for the complete data set), and execution of the complete set of queries (570 queries in total) varied from 9 minutes to 13 hours.

Keywords: SPARQL, RDF benchmark, parliamentary proceedings.

1 Introduction

RDF stores are the backbones of RDF data driven applications. There is a wide range of RDF store systems available¹ together with various benchmark systems² to assess performances of the RDF stores.

As discussed in the Benchmark Handbook [1], different applications impose different requirements to a system, and the performance of the system may vary from one application domain to another. This creates the need for domain specific benchmarks. The existing application benchmarks for RDF store systems often employ techniques developed by the Transaction Processing Performance Council (TPC)³ for relational databases and use synthetically generated data sets for their workloads. However, performance characteristics for loading and querying such data may differ from those that were measured on real life data sets, as it was shown by the DBpedia benchmark [2] on DBpedia [6]. To the best of our knowledge, among the existing benchmarks for RDF store systems, only the DBpedia benchmark provides a real data set.

* This work was supported by the EU- FP7 (FP7/2007-2013) project ENVRI (grant number 283465).

¹ <http://www.w3.org/wiki/LargeTripleStores>, last accessed: July 9, 2013.

² <http://www.w3.org/wiki/RdfStoreBenchmarking>, last accessed: July 9, 2013.

³ <http://www.tpc.org/>, last accessed: July 10, 2013.

With this work we propose the ParlBench application benchmark that closely mimics a real-life scenario: large scale electronic publishing with OLAP-type queries. ParlBench consists of (1) real life data and (2) a set of analytical queries developed on top of these data.

The benchmark data sets include the Dutch parliamentary proceedings, political parties and politicians. The ParlBench data fit very well the desiderata of Gerhard Weikum’s recent Sigmod blog⁴: it is open, big, real, useful, linked to other data sources, mixing data-values and free text, and comes with a number of real-life workloads.

The queries in the benchmark can be viewed as coming from one of two use cases: create a report or perform a scientific research. As an example of the latter, consider the question whether the performance of males and females differs in parliament, and how that has changed over the years. To enable more comprehensive analysis of the RDF stores’ performances, we grouped the benchmark queries into four micro benchmarks [3] with respect to their analytical aims *Average*, *Count*, *Factual* and *Top 10*.

The paper is organized as follows. Section 2 gives an overview of the related work. Section 3 describes the benchmark data sets. In Section 4 we define the benchmark queries and present the micro benchmarks. An experimental run of ParlBench on the Virtuoso RDF store is discussed in Section 5.

2 Related Work

There is a number of RDF store benchmarks available. The most relevant benchmarks to our work are discussed further. The Berlin SPARQL Benchmark (BSBM) [4] implements an e-commerce application scenario. Similarly to ParlBench, BSBM employed the techniques developed by the Transaction Processing Performance Council (TPC)⁵, such as query permutations (for the Business Intelligence use case) and system ramp-up.

The SPARQL Performance Benchmark (SP²Bench) [5] is settled in the DBLP scenario. SP²Bench queries are carefully designed to test the behavior of RDF stores in relation to common SPARQL constructs, different operator constellations and RDF access patterns. SP²Bench measures query response time in cold runs settings, i.e., when query execution time is measured immediately after the server was started.

Both the Berlin and SP²Bench use synthetically generated data sets, whereas, the DBpedia SPARQL Benchmark (DBPSB) [2] uses a real data set, DBpedia [6]. In addition to using a real data set, the DBPSB benchmark uses real queries that were issued by humans and applications against DBpedia. These queries cover most of the SPARQL features and enable comprehensive analysis of RDF stores’ performances on a single feature as well as combinations of features. The main difference between the ParlBench and DBPSB benchmarks is that the latter is not developed with a particular application in mind. Thus, it is more useful for a general assessment of the performance of different RDF stores’ implementations, while ParlBench is particularly targeted on developers of e-publishing applications and can support them in choosing systems that are more suitable for analytical query processing.

⁴ <http://wp.sigmod.org/?p=786>, last accessed: July 1, 2013.

⁵ <http://www.tpc.org/>, last accessed: July 1, 2013.

3 Benchmark Data Sets

The benchmark consists of five conceptually separate data sets summarized in Table 1:

Members : describes political players of the Dutch parliament.

Parties : describes Dutch political parties.

Proceedings : describes the structure of the Dutch parliamentary proceedings.

Paragraphs : contains triples linking paragraphs to their content.

Tagged entities : contains triples linking paragraphs to DBpedia entities indicating that these entities were discussed in the paragraphs.

Table 1. Statistics of the benchmark data sets

data set	# of triples	size	# of files
members	33,885	14M	3,583
parties	510	612K	151
proceedings	36,503,688	4.15G	51,233
paragraphs	11,250,295	5.77G	51,233
tagged entities	34,449,033	2.57G	34,755
TOTAL:	82,237,411	~13G	140,955

The data model of the benchmark data sets is described in Appendix A.

3.1 Scaling of the Benchmark Data Sets

The size of the ParlBench data sets can be changed in different ways. The data set can be scaled by the number of included proceedings. All proceeding files are ordered chronologically by year. In order to construct a scaled proceeding data set, we randomly sampled from each year of proceedings an equal number of proceedings. This was done to ensure uniform and sufficient presence of proceedings from each year. Optionally, one can include *Tagged entities* and/or *Paragraphs* data sets to the test collection. In this case *Paragraphs* and *Tagged Entities* are scaled accordingly to the included proceedings, i.e., only paragraphs and/or tags pointing to id's in the chosen proceedings are included.

4 Benchmark Queries

ParlBench provides 19 SPARQL queries. The queries were grouped into four micro benchmarks:

Average: 3 queries, numbered from A0 to A2, retrieve aggregated information.

Count: 5 queries, numbered from C0 to C4, count entities that satisfy certain filtering conditions.

Factual: 6 queries, numbered from F0 to F5, retrieve instances of a particular class that satisfy certain filtering conditions.

Top 10: 5 queries, numbered from T0 to T4, retrieve the top 10 instances of a particular class that satisfy certain filtering conditions.

All the queries are listed in Appendix B. Their SPARQL representations can be seen in Appendix C. The benchmark queries cover a wide range of the SPARQL language constructs. Table 2 shows the usage of SPARQL features by individual query and distribution of the features across micro benchmarks.

Table 2. SPARQL characteristics of the benchmark queries

micro benchmark	Average			Count				Factual					Top 10						
	A0	A1	A2	C0	C1	C2	C3	C4	F0	F1	F2	F3	F4	F5	T0	T1	T2	T3	T4
FILTER																			
UNION	+	+	+						+	+	+	+	+	+				+	+
LIMIT										+			+		+	+	+	+	+
ORDER BY										+					+	+	+	+	+
GROUP BY	+	+	+					+	+	+	+	+	+	+	+	+	+	+	+
COUNT	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DISTINCT									+			+	+	+					
AVG	+	+	+																
negation															+				
OPTIONAL									+						+				
subquery	+	+	+					+	+	+	+	+	+	+					
blank node scoping				+	+	+	+	+	+	+	+	+	+	+					
# of triple patterns	10	9	12	5	5	5	6	13	8	16	6	6	2	4	2	4	9	3	11

5 Experimental Run of the Benchmark

In this section we demonstrate the application of our benchmark on the Virtuoso RDF native store (OSE, v.06.01.3127)⁶. Tested on the Berlin benchmark, Virtuoso showed one of the best performance results among other systems [4].

5.1 Experimental Setup

Test Environment. For the benchmark experiment we used a personal laptop Apple MacBook Pro with Intel i7 CPU (2x2 cores) running at 2.8 GHz and 8GB memory. See Appendix D for a more detailed specification of the test environment.

Evaluation Metrics

Loading Time. The loading time is the time for loading RDF data into an RDF store. The benchmark data sets are in RDF/XML format. The time is measured in seconds. Loading of data into Virtuoso was done one data set at a time. For the loading of *Parties* and *Members* we used the Virtuoso RDF bulk load procedure. For *Proceedings* we used the Virtuoso function `DB.DBA.RDF_LOAD_RDFXML.MT` to load large RDF/XML text.

Query Response Time. The query response time is the time it takes to execute a SPARQL query. To run the queries programmatically, we used `isql`, the Virtuoso interactive SQL utility. The execution time of a single query was taken as the real time returned by the `bash /usr/bin/time` command. 10 permutations of the benchmark queries were created, each containing 19 SPARQL queries.

Before starting measuring the query response time, we *warmed-up* Virtuoso by running 5 times 10 different permutations of all 19 queries of the benchmark. In total, 950 queries were executed in the *warm-up* phase, and each query was run 50 times. After that, we run the same permutations 3 more times and measured the execution time of each query. The query response time was computed as the mean response time of executing each query 30 times.

Test Collections. Experiments are run on 8 test collections. Each collection includes the *Parties* and *Members* data sets and a scaled *Proceedings* data set ranging from 1 to 100%. Table 3 gives an overview of the size of each test collection.

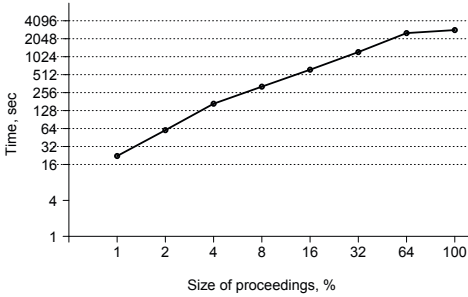
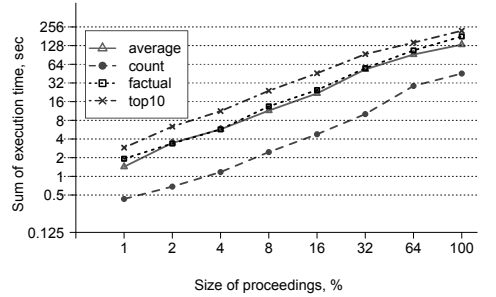
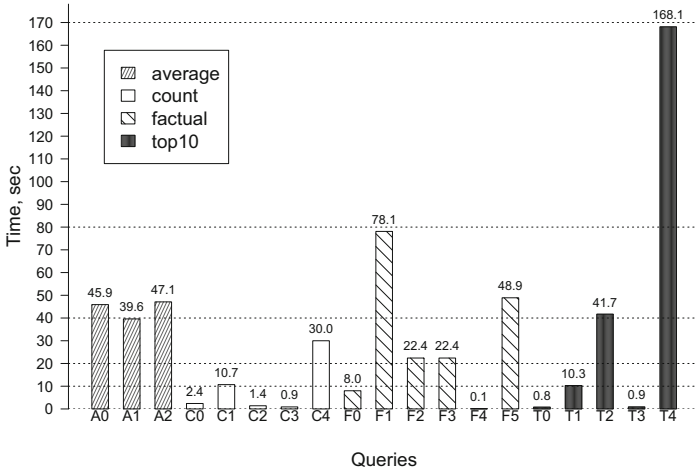
⁶ <http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/>, last accessed: July 1, 2013.

Table 3. Sizes of the test collections for different scaling of *Proceedings*

Scaling Factor	1%	2%	4%	8%	16%	32%	64%	100%
# of triples	494,875	1,027,395	1,906,880	3,851,642	7,554,304	15,129,621	23,341,602	36,542,431

5.2 Results

We report on three experiments, relating database size to execution time: (1) time needed to load the test collections (see fig. 1), (2) total time needed to execute all the queries in micro benchmarks⁷ (see fig. 2), and (3) query execution time of all the queries on the largest collection (see fig. 3).


Fig. 1. Loading Time in sec of the benchmark collections

Fig. 2. Query Execution Time in sec of micro benchmarks on the test collections

Fig. 3. Query Execution Time in sec of the benchmark queries on the largest test collection

The y-axes on fig. 1 and fig. 2 are presented in a log scale, and the numbers represent the loading and query response time in seconds. To make the results reproducible, we publish the benchmark data sets, queries and scripts at <http://data.politicalmashup.nl/RDF/>.

⁷ For each group we summed the execution time of each query in the group.

6 Conclusion

ParlBench has the proper characteristics of an RDF benchmark: it can be scaled easily and it has a set of intuitive queries which measure different aspects of the SPARQL engine.

We believe that ParlBench is a good proxy for a realistic large scale digital publishing application. ParlBench provides real data that encompass major characteristics shared by most of the e-publishing use cases including rich metadata and hierarichal organization of the content into text chunks.

The data set is large enough to perform non-trivial experiments. In addition to the analytical scenario presented, one can think of several other application scenarios that can be developed on the same data sets. Due to the many and strong connections of the benchmark to the Linked Open Data Cloud through the DBpedia links, natural Linked Data integration scenarios can be developed from ParlBench. The ParlBench data is also freely available in XML format [7], enabling cross-platform comparisons of the same workload.

As a future work we will consider the execution of the benchmark on other RDF stores and comparison of the results with the ones achieved on Virtuoso.

Another interesting direction for future work could be to extend the set of queries. Currently, there are only two queries with the `OPTIONAL` operator and one query with `negation`. Queries that use these and other features of SPARQL 1.1 could be a good addition to the benchmark. ParlBench has many queries that extensively use the `UNION` to construct various paths based on the `hasPart` property. We could re-write these queries through the SPARQL 1.1. path expressions and exploit the transitive property of `hasPart` to test the reasoning capabilities of RDF store systems.

References

1. Gray, J.: The Benchmark Handbook for Database and Transaction Systems, 2nd edn. Morgan Kaufmann (1993)
2. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
3. Afanasiev, L., Manolescu, I., Michiels, P.: MemBeR: A Micro-benchmark Repository for XQuery. In: Bressan, S., Ceri, S., Hunt, E., Ives, Z.G., Bellahsène, Z., Rys, M., Unland, R. (eds.) XSym 2005. LNCS, vol. 3671, pp. 144–161. Springer, Heidelberg (2005)
4. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. *International Journal on Semantic Web and Information Systems* 5(2), 1–24 (2009)
5. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP²Bench: A SPARQL Performance Benchmark. In: ICDE, pp. 222–233. IEEE (2009)
6. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. *Journal of Web Semantics (JWS)* 7, 154–165 (2009)
7. Maarten, M.: Advanced Information Access to Parliamentary Debates. *Journal of Digital Information (JoDI)* 10(6) (2009)
8. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)* 34(3), 16:1–16:45 (2009)

A Data Model

Figure 4 gives an overview of the data model. The data model consists of the following classes:

- **PoliticalParty** represents political parties. Parties are linked to their equivalent resources in DBpedia.
- **ParliamentMember** represents politicians, i.e., members of the parliament. Members are linked to the DBpedia resources that correspond to the same politicians. Additionally, members have biographical information such as gender, birthday and the place of birth, and the death date and place if applicable.
- **ParliamentaryProceedings** are written records of parliamentary meetings.
- **Topic** represents a different point on the agenda in the proceedings.
- **Scene** and **StageDirection** are important structural elements of the Dutch parliamentary proceedings. They contain information about current speakers and their interrupters.
- **Speech** is a constituent of a *Topic*, a *Scene* or *Stage Direction*. *Speech* represents a beginning of a speech of a new speaker.
- **Paragraph** is a container for all spoken text; can be part of any structural element of the proceedings described above.

The structural elements of the proceedings are connected to **Proceedings** through the `dcterms:hasPart` property. The speaker of the speech and the affiliated party of the speaker are attached to **Speech** via the `refMember` and `refParty` properties correspondingly.

Vocabularies. Relevant existing vocabularies and ontologies to represent documents of parliamentary proceedings fall into two categories. In the first category we identified vocabularies that are too generic, such as the SALT Document Ontology⁸ or the DoCo, the Document Components Ontology⁹. They do not provide means to represent such specific concepts as *stage direction* or *scene*. Vocabularies in the second category are too specific. For example, the Semantic Web Conference Ontology¹⁰ or the Semantic Web Portal Ontology¹¹ define classes and properties to model conference proceedings.

⁸ <http://salt.semanticauthoring.org/ontologies/sdo#>

⁹ <http://purl.org/spar/doco/Paragraph>

¹⁰ http://data.semanticweb.org/ns/swc/swc_2009-05-09.html#

¹¹ <http://sw-portal.deri.org/ontologies/swportal#>

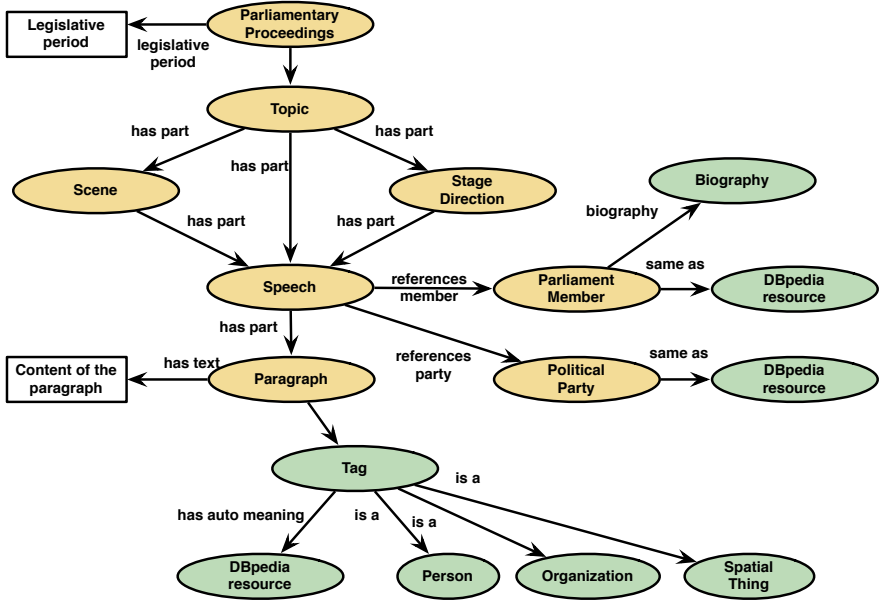


Fig. 4. Data model of the benchmark data sets

We defined our own RDFS vocabulary to model parliamentary proceedings, the Parliamentary Proceedings vocabulary¹², and integrated it with other existing vocabularies. To represent biographical information of politicians, we used BIO: A vocabulary for biographical information¹³ together with the Friend of a Friend Vocabulary (FOAF)¹⁴ and the DBpedia Ontology¹⁵. The Modular Unified Tagging Ontology (MUTO)¹⁶ was used to represent information about tagged entities of paragraphs. The Dublin Core Metadata Terms¹⁷ was used to encode metadata information.

¹² <http://purl.org/vocab/parlipro#>

¹³ <http://vocab.org/bio>

¹⁴ <http://xmlns.com/foaf/0.1/>

¹⁵ <http://dbpedia.org/ontology/>

¹⁶ <http://muto.socialtagging.org/>

¹⁷ <http://purl.org/dc/terms/> and <http://purl.org/dc/elements/1.1/>

B ParlBench Queries

Table 4. List of the benchmark queries

average		
1.	A0	Retrieve average number of people spoke per topic.
2.	A1	Retrieve average number of speeches per topic.
3.	A2	Retrieve average number of speeches per day.
count		
4.	C0	Count speeches of females.
5.	C1	Count speeches of males.
6.	C2	Count speeches of speakers who were born after 1960.
7.	C3	Count speeches of male speakers who were born after 1960.
8.	C4	Count speeches of a female speaker from the topic where only one female spoke.
factual		
9.	F0	What members were born after 1950, their parties and dates of death if exist?
10.	F1	What is the gender of politicians who spoke most between 1995 and 2005?
11.	F2	What is the percentage of male speakers?
12.	F3	What is the percentage of female speakers?
13.	F4	What politician has most number of Wikipedia pages in different languages?
14.	F5	What speeches are made by politicians without Wikipedia pages?
top 10		
15.	T0	Retrieve top 10 members with the most number of speeches.
16.	T1	Retrieve top 10 topics when most of the people spoke.
17.	T2	Retrieve top 10 topics with the most number of speeches.
18.	T3	Retrieve top 10 days with the most number of topics.
19.	T4	Retrieve top 10 longest topics (i.e., topics with the most number of paragraphs).

C ParlBench SPARQL Queries

Table 5. Prefixes used in the SPARQL queries

rdf:	<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
parlipro:	<http://purl.org/vocab/parlipro#>
dcterms:	<http://purl.org/dc/terms/>
dc:	<http://purl.org/dc/elements/1.1/>
bio:	<http://purl.org/vocab/bio/0.1/>
foaf:	<http://xmlns.com/foaf/0.1/>
dbpedia:	<http://dbpedia.org/resource/>
owl:	<http://www.w3.org/2002/07/owl#>

Table 6. SPARQL representation of the benchmark queries

<p>A0: Retrieve average number of people spoke per topic.</p> <pre> SELECT AVG(?numOfMembers) as ?avgNumOfMembersPerTopic WHERE {{ SELECT COUNT(?member) AS ?numOfMembers WHERE { ?topic rdf:type parlipro:Topic . ?speech rdf:type parlipro:Speech . ?speech parlipro:refMember ?member . {?topic dcterms:hasPart ?speech .} UNION{ ?topic dcterms:hasPart ?sd . ?sd rdf:type parlipro:StageDirection . ?sd dcterms:hasPart ?speech .} UNION{ ?topic dcterms:hasPart ?scene . ?scene rdf:type parlipro:Scene . ?scene dcterms:hasPart ?speech .}} GROUP BY ?topic}}</pre>
<p>A1: Retrieve average number of speeches per topic.</p> <pre> SELECT AVG(?numOfSpeeches) as ?avgNumOfSpeechesPerTopic WHERE {{ SELECT COUNT(?speech) AS ?numOfSpeeches WHERE { ?topic rdf:type parlipro:Topic . ?speech rdf:type parlipro:Speech . {?topic dcterms:hasPart ?speech .} UNION{ ?topic dcterms:hasPart ?sd . ?sd rdf:type parlipro:StageDirection . ?sd dcterms:hasPart ?speech .} UNION{ ?topic dcterms:hasPart ?scene . ?scene rdf:type parlipro:Scene . ?scene dcterms:hasPart ?speech .}} GROUP BY ?topic}}</pre>

A2: Retrieve average number of speeches per day.

```
SELECT AVG(?numOfSpeeches) as ?avgNumOfSpeechesPerDay
WHERE {{
SELECT ?date COUNT(?speech) AS ?numOfSpeeches
WHERE {
  ?proc dcterms:hasPart ?topic .
  ?proc rdf:type parlipro:ParliamentaryProceedings .
  ?proc dc:date ?date .
  ?speech rdf:type parlipro:Speech .
  ?topic rdf:type parlipro:Topic .
  {?topic dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?sd .
  ?sd rdf:type parlipro:StageDirection .
  ?sd dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?scene .
  ?scene rdf:type parlipro:Scene .
  ?scene dcterms:hasPart ?speech .}}
GROUP BY ?date}}
```

C0: Count speeches of females.

```
SELECT COUNT(?speech)
WHERE {
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:gender dbpedia:Female .}
```

C1: Count speeches of males.

```
SELECT COUNT(?speech)
WHERE {
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:gender dbpedia:Male .}
```

C2: Count speeches of speakers who were born after 1960.

```
SELECT COUNT(?speech)
WHERE {
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:birthday ?birthday .
  FILTER (year(?birthday) > 1960)}
```

C3: Count speeches of male speakers who were born after 1960.

```
SELECT COUNT(?speech)
WHERE {
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:gender dbpedia:Male .
  _:bio foaf:birthday ?birthday .
  FILTER (year(?birthday) > 1960)}
```

C4: Count speeches of a female speaker from the topic where only one female spoke.

```
SELECT ?topic ?member COUNT(?speech) as ?numOfSpeeches
WHERE {{
  SELECT ?topic ?member COUNT(?member) AS ?numOfFemales ?speech
  WHERE {
    ?topic rdf:type parlipro:Topic .
    ?speech rdf:type parlipro:Speech .
    ?speech parlipro:refMember ?member .
    ?member bio:biography _:bio .
    _:bio rdf:type bio:Biography .
    _:bio foaf:gender dbpedia:Female .
    {?topic dcterms:hasPart ?speech .}
  }
  UNION{
    ?topic dcterms:hasPart ?sd .
    ?sd rdf:type parlipro:StageDirection .
    ?sd dcterms:hasPart ?speech .}
  UNION{
    ?topic dcterms:hasPart ?scene .
    ?scene rdf:type parlipro:Scene .
    ?scene dcterms:hasPart ?speech .}}
GROUP BY ?topic ?member ?speech}
FILTER (?numOfFemales = 1 )}
GROUP BY ?topic ?member
```

F0: What members were born after 1950, their parties and dates of death if exist?

```
SELECT DISTINCT ?member ?party ?birthday
WHERE {
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?speech parlipro:refParty ?party .
  ?member rdf:type parlipro:ParliamentMember .
  ?member bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:birthday ?birthday .
  FILTER (year(?birthday) > 1960)}
FILTER (year(?birthday) > 1950)
OPTIONAL{_:bio dbpedia-ont:deathDate ?deathDate .}}
```

F1: What is the gender of politicians who spoke most between 1995 and 2005?

```

SELECT ?gender COUNT(?member) AS ?numOfMembers
WHERE {
  ?proc rdf:type parlipro:ParliamentaryProceedings .
  ?proc dc:date ?date .
  ?proc dcterms:hasPart ?topic .
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member rdf:type parlipro:ParliamentMember .
  ?member bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:gender ?gender .
  {?topic dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?sd .
  ?sd rdf:type parlipro:StageDirection .
  ?sd dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?scene .
  ?scene rdf:type parlipro:Scene .
  ?scene dcterms:hasPart ?speech .}
FILTER (year(?date) > 1995 AND year(?date) < 2005)}
GROUP BY ?gender
ORDER BY DESC(?numOfMembers)
LIMIT 1

```

F2: What is the percentage of male speakers?

```

SELECT (?numOfMaleMembers*100)/?numOfMembers
WHERE{{
SELECT COUNT(DISTINCT ?memberMale) as ?numOfMaleMembers
      COUNT(DISTINCT ?member) as ?numOfMembers
WHERE {{
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member rdf:type parlipro:ParliamentMember .}
UNION{
  ?memberMale bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:gender dbpedia:Male .
FILTER (sameTerm(?member,?memberMale))}}}}

```

F3: What is the percentage of female speakers?

```
SELECT (?numOfFemaleMembers*100)/?numOfMembers
WHERE{{
SELECT COUNT(DISTINCT ?memberFemale) as ?numOfFemaleMembers
      COUNT(DISTINCT ?member) as ?numOfMembers
WHERE {{
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member rdf:type parlipro:ParliamentMember .}
UNION{
  ?memberFemale bio:biography _:bio .
  _:bio rdf:type bio:Biography .
  _:bio foaf:gender dbpedia:Female .
FILTER (sameTerm(?member,?memberFemale))}}}}
```

F4: What politician has most number of Wikipedia pages in different languages?

```
SELECT ?member ?numOfPages
WHERE {{
SELECT DISTINCT ?member COUNT(?dbpediaMember) AS ?numOfPages
WHERE {
  ?member rdf:type parlipro:ParliamentMember .
  ?member owl:sameAs ?dbpediaMember .}
GROUP BY ?member}}
ORDER BY DESC(?numOfPages)
LIMIT 1
```

F5: What speeches are made by politicians without Wikipedia pages?

```
SELECT DISTINCT ?speech ?member
WHERE {
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .
  ?member rdf:type parlipro:ParliamentMember .
OPTIONAL {?member owl:sameAs ?dbpediaMember .}
FILTER (!bound(?dbpediaMember))}
```

T0: Retrieve top 10 members with the most number of speeches.

```
SELECT ?member COUNT(?speech) as ?numOfSpeeches
WHERE {
  ?member rdf:type parlipro:ParliamentMember .
  ?speech parlipro:refMember ?member .}
GROUP BY ?member
ORDER BY DESC(?numOfSpeeches)
LIMIT 10
```

T1: Retrieve top 10 topics when most of the people spoke.

```
SELECT ?topic COUNT(?member) as ?numOfMembersSpokeInTopic
WHERE {
  ?topic rdf:type parlipro:Topic .
  ?topic dcterms:hasPart ?speech .
  ?speech rdf:type parlipro:Speech .
  ?speech parlipro:refMember ?member .}
GROUP BY ?topic
ORDER BY DESC(?numOfMembersSpokeInTopic)
LIMIT 10
```

T2: Retrieve top 10 topics with the most number of speeches.

```
SELECT ?topic
COUNT(?speech) as ?numOfSpeeches
WHERE {
  ?topic rdf:type parlipro:Topic .
  ?speech rdf:type parlipro:Speech .
  {?topic dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?sd .
  ?sd rdf:type parlipro:StageDirection .
  ?sd dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?scene .
  ?scene rdf:type parlipro:Scene .
  ?scene dcterms:hasPart ?speech .}}
GROUP BY ?topic
ORDER BY DESC(?numOfSpeeches)
LIMIT 10
```

T3: Retrieve top 10 days with the most number of topics.

```
SELECT ?date COUNT(?topic) as ?numOfTopics
WHERE {
  ?proc rdf:type parlipro:ParliamentaryProceedings .
  ?proc dcterms:hasPart ?topic .
  ?proc dc:date ?date .}
GROUP BY ?date
ORDER BY DESC(?numOfTopics)
LIMIT 10
```

T4: Retrieve top 10 longest topics (i.e., topics with the most number of paragraphs).

```

SELECT ?topic COUNT(?par) as ?numOfPars
WHERE {
  ?topic rdf:type parlipro:Topic .
  ?speech rdf:type parlipro:Speech .
  ?speech dcterms:hasPart ?par .
  ?par rdf:type parlipro:Paragraph .
  {?topic dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?sd .
  ?sd rdf:type parlipro:StageDirection .
  ?sd dcterms:hasPart ?speech .}
UNION{
  ?topic dcterms:hasPart ?scene .
  ?scene rdf:type parlipro:Scene .
  ?scene dcterms:hasPart ?speech .}}
GROUP BY ?topic
ORDER BY DESC(?numOfPars)
LIMIT 10

```

D Test Machine Specification

For the benchmark evaluation we used a personal laptop Apple MacBook Pro. The operating system running is Mac OS X Lion 10.7.5 x64. The specification of the machine is the following:

Hardware

- CPUs: 2.8 GHz Intel Core i7 (2x2 cores)
- Memory: 8 GB 1333 MHz DDR3
- Hard Disk: 750GB

Software

- OpenLink Virtuoso: Open Source Edition v.06.01.3127 compiled from source for OS X
- MySQL Community Server (GPL) v. 5.5.15
- Scripts (bash 3.2, Python 2.7.3) to scale and upload RDF data sets, to create permutations of queries and run them on Virtuoso. The scripts are available for downloading at <http://data.politicalmashup.nl/RDF/scripts/>.

Virtuoso Configuration We configured the Virtuoso Server to handle load of large data sets¹⁸.

¹⁸ The configuration was performed following the guidelines at <http://www.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtRDFPerformanceTuning>

NumberOfBuffers = 680000

MaxDirtyBuffers = 500000

The RDF Index Scheme remained as it was supplied with the default Virtuoso installation. Namely, the scheme consists of the following indexes:

- PSOG - primary key.
- POGS - bitmap index for lookups on object value.
- SP - partial index for cases where only S is specified.
- OP - partial index for cases where only O is specified.
- GS - partial index for cases where only G is specified.

I See a Car Crash: Real-Time Detection of Small Scale Incidents in Microblogs

Axel Schulz^{1,2}, Petar Ristoski¹, and Heiko Paulheim³

¹ SAP Research

² Technische Universität Darmstadt
Telecooperation Lab.

{axel.schulz,petar.ristoski}@sap.com

³ University of Mannheim

Data and Web Science Group

heiko@informatik.uni-mannheim.de

Abstract. Microblogs are increasingly gaining attention as an important information source in emergency management. Nevertheless, it is still difficult to reuse this information source during emergency situations, because of the sheer amount of unstructured data. Especially for detecting small scale events like car crashes, there are only small bits of information, thus complicating the detection of relevant information.

We present a solution for a real-time identification of small scale incidents using microblogs, thereby allowing to increase the situational awareness by harvesting additional information about incidents. Our approach is a machine learning algorithm combining text classification and semantic enrichment of microblogs. An evaluation based shows that our solution enables the identification of small scale incidents with an accuracy of 89% as well as the detection of all incidents published in real-time Linked Open Government Data.

1 Introduction

Social media platforms are widely used for sharing information about incidents. Ushahidi, a social platform used for crowd-based filtering of information [12], was heavily used during the Haitian earthquake for labeling crisis related information, as well as incidents such as the Oklahoma grass fires and the Red River floods in April 2009 [19] or the terrorist attacks on Mumbai [3]. All these examples show that citizens already act as observers in crisis situations and provide potentially valuable information on different social media platforms.

Current approaches for using social media in emergency management largely focus on large scale incidents like earthquakes. Large-scale incidents are characterized by a large number of social media messages as well as a wide geographic and/or temporal coverage. In contrast, small-scale incidents, such as car crashes or fires, usually have a small number of social media messages and only narrow geographic and temporal coverage. This imposes certain challenges on approaches for detecting small scale incidents: large incidents, like earthquakes,

with thousands of social media postings, are much easier to detect than smaller incidents with only a dozen of postings, and further processing steps, such as the extraction of factual information, can work on a larger set of texts. For large scale incidents, one can optimize systems for precision, whereas recall is not problematic due to the sheer amount of information on the incident. In contrast, detecting small scale incidents imposes much stricter demands on *both* precision and recall.

The approach discussed in this paper combines information from the social and the semantic web. While the social web consists of text, images, and videos, all of which cannot be processed by intelligent agents easily, the Linked Open Data (LOD) cloud contains semantically annotated, formally captured information. Linked Open Data can be used to enhance Web 2.0 contents by semantically enriching it, as shown, e.g., in [4] and [6]. Such enrichments may also be used as features for machine learning [5]. In our approach, we leverage machine learning and semantic web technologies for detecting user-generated content that is related to a small scale incident with high accuracy. Furthermore, we refined current approaches for spatial and temporal filtering allowing us to detect space and time information of a small scale incident more precisely. We further show how Linked Open Government Data can be used to evaluate classifications.

This paper contributes an approach that leverages information provided in the social web for detection of small scale incidents. The proposed method consists of two steps: (1) automatic classification of user-generated content related to small scale incidents, and (2) prefiltering of irrelevant content, based on spatial, temporal, and thematic aspects.

The rest of this paper is structured as follows: In section 2, we review related approaches. Section 3 provides a detailed description of our process, followed by an evaluation in section 4. In section 5, we show an example application. We conclude in Section 6 with a short summary and an outlook on future work.

2 Related Work

Event and incident detection in social media gained increased attention the last years. In this case, various machine learning approaches have been proposed for detecting large scale incidents in microblogs, e.g., in [7],[10],[14].

All those approaches focus on large scale incidents. On the other side, only few state-of-the-art approaches focus on the detection of small scale incidents in microblogs. Agarwal et al. [2] focus on detecting events related to a fire in a factory. They rely on standard NLP-based features like Named Entity Recognition (NER) and part-of-speech tagging. Furthermore, they employ a spatial dictionary to geolocalize tweets on city level. They report a precision of 80% using Naïve Bayes classifier.

Twitcident [1] is a mashup for filtering, searching, and analyzing social media information about small scale incidents. The system uses information about incidents published in an emergency network in the Netherlands for constructing an initial query to crawl relevant tweets. The collected messages are further

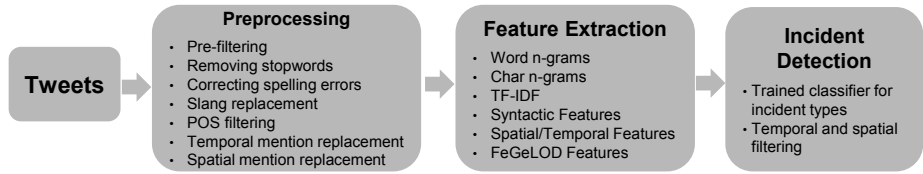


Fig. 1. Pipeline

processed by the semantic enrichment module which includes NER using *DBpedia Spotlight* [11]. The extracted concepts are used as attribute-value pairs, e.g., '(location, `dbpedia:Austin_Texas`)'. In this case, those pairs are used to create a weighting of important concepts for different types of incidents. Furthermore, referenced web pages in the tweet message are provided as external information. A classification of tweets is done using manually created rules based on the attribute-value pairs and keywords. Though they show an advantage of using semantic features, they do not provide any evaluation results for detecting small scale incidents.

Wanichayapong et al. [20] focus on extracting traffic information in microblogs from Thailand. Compared to other approaches, they use an approach which detects tweets that contain place mentions as well as traffic related information. The evaluation of the approach was made on 1249 manually labeled tweets and showed an accuracy of 91.7%, precision of 91.39%, and recall of 87.53%. Though the results are quite promising, they restricted their initial test set to tweets containing manually defined traffic related keywords, thus, the number of relevant tweets is significantly higher than in a random stream.

Li et al. [8] introduce a system for searching and visualization of tweets related to small scale incidents, based on keyword, spatial, and temporal filtering. Compared to other approaches, they iteratively refine a keyword-based search for retrieving a higher number of incident related tweets. Based on these tweets a classifier is built upon Twitter specific features, such as hashtags, @-mentions, URLs, and spatial and temporal characteristics. Furthermore, events are geolocalized on city scale. They report an accuracy of 80% for detecting incident related tweets, although they do not provide any information about their evaluation approach.

In summary, only few of the mentioned approaches make use of semantic web technologies for detecting small scale incidents. Furthermore, all of the mentioned approaches do not compare with events published in governmental data sources, thus the detection of incidents in real-time remains unevaluated.

3 Approach

Figure 1 shows our pipeline for detecting incident related tweets. The pipeline is divided into two phases. First, a classifier for detecting incident related tweets is trained. In this case, we crawl and preprocess all incoming tweets. Based on

the preprocessed tweets, several features are extracted and used for classification. Second, for real-time incident detection, new tweets are first filtered based on spatial and temporal filtering. Then the classifier is applied to detect incident related information. For optimizing our classifier, we evaluate the results on incident reports published in Linked Open Government Data. As a result, the optimized pipeline can be used to present valuable information for decision makers in emergency management systems.

3.1 Classification

Crawling and Preprocessing. We continuously collect tweets using the Twitter Search API¹. In this case, we restrict our search to English tweets of certain cities. Every collected tweet is then preprocessed. First, we remove all retweets as these are just duplicates of other tweets and do not provide additional information. Second, @-mentions in the tweet message are removed as we assume that they are not relevant for detection of incident related tweets. Third, very frequent words like stop words are removed as they are not valuable as features for a machine learning algorithm. Fourth, abbreviations are resolved using a dictionary compiled from www.noslang.com. Furthermore, as tweets contain spelling errors, we apply the Google Spellchecking API² to identify and replace them if possible.

We use our temporal detection approach (see below) to detect temporal expressions and replace them with two annotations @DATE and @TIME, so we prevent overfitting the classification model for temporal values from the training dataset. Likewise, we use our spatial detection approach (see below) to detect place and location mentions and replace them with two annotations @LOC and @PLC.

Before extracting features, we normalize the words using the Stanford lemmatization function³. Furthermore, we apply the Stanford POS tagger⁴. This enables us to filter out some word categories, which are not useful for our approach. E.g., during our evaluation we found out that using only nouns and proper nouns for classification improve the accuracy of the classification (cf. Section 4.3)

Feature Extraction. After finishing the initial preprocessing steps, we extract several features from the tweets that will be used for training a classifier:

Word unigram extraction: A tweet is represented as a set of words. We use two approaches: a vector with the frequency of words, and a vector with the occurrence of words (as binary values).

Character n-grams: A string of three respective four consecutive characters in a tweet message is used as a feature. For example, if a tweet is: “Today is so hot. I feel tired” then the following trigrams are extracted: “tod”, “oda”,

¹ <https://dev.twitter.com/docs/api/1.1/get/search/tweets>

² <https://code.google.com/p/google-api-spelling-java/>

³ <http://nlp.stanford.edu/software/corenlp.shtml>

⁴ <http://nlp.stanford.edu/software/tagger.shtml>

“day”, “ay ”, “y I”, etc.. To construct the trigram respective fourgram list, all the special characters, which are not a letter, a space character, or a number, are removed.

TF-IDF: For every document we calculate an accumulated tf-idf score [9]. It measures the overall deviation of a tweet from all the positive tweets in the training set by summing the tf-idf scores of that tweet, where the idf is calculated based on all the positive examples in the training set.

Syntactic features: Along with the features directly extracted from the tweet, several syntactic features are expected to improve the performances of our approach. People might tend to use a lot of punctuations, such as explanation mark and question mark, or a lot of capitalized letter when they are reporting some incident. In this case, we extract the following features: the number of “!” and “?” in a tweet and the number of capitalized characters.

Spatial and Temporal unigram features: As spatial and temporal mentions are replaced with corresponding annotations, they appear as word unigrams or character n-grams in our model and can therefore be regarded as additional features.

Linked Open Data features: *FeGeLOD* [13] is a framework which extracts features for an entity from Linked Open Data. For example, for the instance `dbpedia:Ford_Mustang`, features that can be extracted include the instance’s direct types (such as `Automobile`), as well as the categories of the corresponding Wikipedia page (such as `Road_transport`). We use the transitive closures of types and categories with respect to `rdfs:subClassOf` and `skos:broader`, respectively. The rationale is that those features can be extracted on a training set and on a test set, even if the actual instance has never been seen in the test set: the types and categories mentioned above could also be generated for a tweet talking about a `dbpedia:Volkswagen_Passat`, for example. This allows for a semantic abstraction from the concrete instances a tweet talks about. In order to generate features from tweets with FeGeLOD, we first preprocess the tweets using *DBpedia Spotlight* in order to identify the instances a tweet talks about. Unlike the other feature extractions, the extraction of Linked Open Data features is performed on the original tweet, not the preprocessed one.

Classification. The different features are combined and evaluated using three classifiers. For classification, the machine learning library Weka [21] is used. We compare a Naïve Bayes Binary Model (NBB), the Ripper rule learner (JRip), and a classifier based on a Support Vector Machine (SVM).

3.2 Real-Time Incident Detection

For real-time incident detection, we first apply spatial and temporal filtering before applying the classifier.

Temporal Extraction. Using the creation date of a tweet is not always sufficient for detecting events, as people also report on incidents that occurred in the

past or events that will happen in the future. During our evaluations we found out, that around 18% of all incident related tweets contain temporal information. Thus, a mechanism can be applied to filter out tweets that are not temporally related to a specific event.

For identifying what the temporal relation of a tweet is, we adapted the HeidelbergTime [18] framework for temporal extraction. HeidelbergTime is a rule-based approach mainly using regular expressions for the extraction of temporal expressions in texts. As the system was developed for large text documents, we adapted it to work on microblogs. Based on our adaptation, we are able to extract time mentions in microblogs like 'yesterday' and use them to calculate the time of the event to which a tweet refers to. E.g., the tweet 'I still remember that car accident from Tuesday', created on FR 15.02.2013 14:33, can now be readjusted to reference an accident on TU 12.02.2013 14:33.

As discussed above, we also use our adaptation to replace time mentions in microblogs with the annotations @DATE and @TIME to use temporal mentions as additional features. Furthermore, compared to other approaches we are able to retrieve precise temporal information about a small scale incident, which enables the real-time detection of current incidents.

Spatial Extraction. Besides a temporal filtering, a spatial filtering is also applied. As only 1% of all tweets retrieved from the Twitter Search API are geotagged, location mentions in tweet messages or the user's profile information have to be identified.

For location extraction, we use a threefold approach. First, location mentions are identified using Stanford NER⁵. As we are only interested in recognizing location mentions, which includes streets, highways, landmarks, blocks, or zones, we retrained the Stanford NER model on a set of tweets, using a set of 1250 manually labeled tweets. We use only two classes for named entities: LOCATION and PLACE. All location mentions, for which we can extract accurate geo coordinates, such as cities, streets and landmarks, are labeled with LOCATION. The words of the tweets that are used as to abstractly describe where the event took place, such as "home", "office", "school" etc., are labeled with PLACE. The customized NER model has precision of 95.5% and 91.29% recall. As discussed above, we also use our adaptation to annotate the text so that a spatial mention can be used as an additional feature. The following tweet may be the result of this approach: "Several people are injured in car crash on <LOC>5th Ave</LOC> <LOC>Seattle</LOC>"

Second, to relate the location mention to a point where the event happened, we geocode the location strings. In this case, we create a set of word unigrams, bigrams, and trigrams. These are sent to the geographical database GeoNames⁶ to identify city names in each of the n-grams and to extract geocoordinates. As city names are ambiguous around the world, we choose the longest n-gram as the most probable city. If there is no city mention in the tweet, we try to extract the city from the location field in the user profile (see [15] for details).

⁵ <http://nlp.stanford.edu/software/CRF-NER.shtml>

⁶ <http://www.geonames.org>

Third, for fine-grained geolocalization, on street or building level, we are using the MapQuest Nominatim API⁷, which is based on OpenStreetMap data. Using this approach we are able to extract precise location information for 87% of the tweets. Compared to other approaches, which rely on city level precision, we are able to precisely geolocalize small scale events on street level.

Classification. After temporal and spatial filtering, we apply our classifier to identify incident related tweets. For further refinement and the verification that our approach enables the real-time detection of tweets, we compare our results with the official incident information published in Linked Open Government Data like the data that can be retrieved from data.seattle.gov. Finally, the detected relevant information can be presented to decision makers in incident management systems (cf. Section 5).

4 Evaluation

We conduct an evaluation of our method on the publicly available Twitter feed. First, we evaluate the performance of the FeGeLOD features. Second, we measure the performance using all features and third, we compare our results to real-time incident reports.

4.1 Datasets

Training Dataset. For building a training dataset, we collected 6 million public tweets using the Twitter Search API from November 19th, 2012 to December 19th, 2012 in a 15km radius around the city centers of Seattle, WA and Memphis, TN. For labeling the tweets, we first extracted tweets containing incident related keywords. We retrieved all incident types using the “Seattle Real Time Fire 911 Calls” dataset from seattle.data.gov and defined one general keyword set with keywords that are used in all types of incidents like “incident”, “injury”, “police” etc. For each incident type we further identified specific keywords, e.g., for the incident type “Motor Vehicle Accident Freeway” we use the keywords “vehicle”, “accident”, and “road”. Based on these words, we use WordNet⁸ to extend this set by adding the direct hyponyms. For instance, the keyword “accident” was extended with “collision”, “crash”, “wreck”, “injury”, “fatal accident”, and “casualty”.

For building our training set for identifying car crashes, we used the general and the specific keyword set for the incident types “Motor Vehicle Accident”, “Motor Vehicle Accident Freeway”, “Car Fire”, and “Car Fire Freeway” to extract tweets that might be related to car crashes. We randomly selected 10k tweets from this set to manually label the tweets in two classes “car accident related” and “not car incident related”. The tweets were labeled by scientific members of our departments. The final training set consists of 993 car accident related and 993 not car accident related tweets.

⁷ <http://developer.mapquest.com/web/products/open/nominatim>

⁸ <http://wordnet.princeton.edu>

Test Dataset. To show that the resulting model using the training dataset is not overfitted to the events in the period when the training data was collected, we collected an additional 1.5 million tweets in the period from February 1st, 2013 to February 7th, 2013 from the same cities. We also used the keyword extraction approach and manually labeled test dataset, resulting in 320 car accident related tweets and 320 not car accident related tweets.

Socrata Dataset. For evaluating our approach with real-time Linked Open Government Data, we use the “Seattle Real Time Fire 911 Calls” dataset from seattle.data.gov. In the period from February 5th, 2013 to February 7th, 2013, we collected information about 15 incidents related to car crashes and 830 related to other incident types.

4.2 Metrics

Our classification results are calculated using stratified 10-fold cross validation on the training set, and evaluating a model trained on the training dataset on the test dataset. To measure the performance of the classification approaches, we report the following metrics:

- Accuracy (Acc): Number of the correctly classified tweets divided by total number of tweets.
- Averaged Precision (Prec): Calculated based on the Precision of each class (how many of our predictions for a class are correct).
- Averaged Recall (Rec): Calculated based on the Recall of each class (how many tweets of a class are correctly classified as this class).
- F-Measure (F): Weighted average of the precision and recall.

4.3 Results

Evaluation of FeGeLOD Features. We used DBpedia Spotlight to detect named entities in the tweet messages⁹. These annotations were used to generate additional features with FeGeLOD, as discussed above. Table 1 shows the classification accuracy achieved using only features generated with FeGeLOD from the training dataset. We used FeGeLOD to create three models with different features. The first one contains only types of the extracted entities, the second one contains only Wikipedia categories of the extracted entities, and the third one contains both categories and types of the extracted entities from the tweets. The best results with accuracy of 67.1% are achieved when using categories and types. Furthermore, analyzing the results from JRip, we get rules using categories as “Accidents”, “Injuries”, or “Road_infrastructure” and Types as “Road104096066” or “AdministrativeArea”. This shows that the features generated by FeGeLOD are actually meaningful.

⁹ The parameters used were: Confidence=0.2; Contextual score= 0.9; Support = 20; Disambiguator = Document; Spotter=LingPipeSpotter; Only best candidate.

Table 1. Classification results for FeGeLOD features.

Features	Method	Training Set			
		Acc	Prec	Rec	F
Types + categories	SVM	67.1%	71.3%	67.1%	65.4%
	NB	64.78%	68.7%	64.8%	62.8%
	JRip	62.42%	62.7%	62.4%	62.2%
Categories	SVM	64.53%	72.3%	64.5%	61.1%
	NB	63.63%	71.9%	63.6%	59.8%
	JRip	61.87%	72.6%	61.9%	65.7%
Types	SVM	61.92%	63.6%	61.9%	60.7%
	NB	58.85%	62.6%	58.9%	55.5%
	JRip	60.51%	60.6%	60.5%	60.4%

Table 2. Classification results for all features

Features	Method	Training Set				Test Set			
		Acc	Prec	Rec	F	Acc	Prec	Rec	F
Word-n-grams+POS filtering, TF-IDF, syntactic, FeGeLOD	SVM	88.43%	88.5%	88.4%	88.4%	89.06%	89.1%	89.1%	89.1%
	JRip	86.46%	87.0%	86.5%	86.4%	84.21%	84.3%	84.2%	84.2%
	NB	85.81%	86.8%	85.8%	85.7%	79.21%	85.1%	79.2%	78.3%
Word-n-grams w/o POS fil- tering, TF-IDF, syntactic	SVM	90.24%	90.4%	90.2%	90.2%	85.93%	86.0%	85.9%	85.9%
	JRip	84.75%	85.5%	84.8%	84.7%	85.93%	86.8%	85.9%	85.9%
	NB	85.86%	86.9%	85.9%	85.8%	86.25%	87.3%	86.3%	86.2%

Evaluation of all Features. In order to evaluate which machine learning features contribute the most to the accuracy of the classifier, we built different classification models for all the combinations of the features that we described in Section 3.1. We first evaluated the models on the training dataset, then we reevaluated the models on the test dataset. Table 2 shows the best classification results achieved using different combinations of machine learning features.

The tests showed that using word n-grams without POS filtering, TF-IDF accumulate score, and syntactic features provide the best classification results for the training dataset. But re-evaluating the same model on the test set that contains data from a different time period, the results dropped significantly. That leads to conclusion that the model is overfitted to the training dataset. Furthermore, the model contains a large number of features and requires a lot of processing performance to be used for predictions in real-time.

Adding the features generated by FeGeLOD did not affect the classification accuracy on the training dataset, but improved the classification accuracy on the test dataset. This shows that using semantic features generated from LOD helps to prevent overfitting. Additionally, we used POS filtering to filter out some word categories from the tweet to see which word categories contribute to the classification performance. The tests showed that using only nouns and proper nouns when generating the word n-grams improve the results on the test set. This approach significantly reduced the number of attributes in the model, making it applicable for real-time predictions.

Evaluation on Real-World Incident Reports. To evaluate how many incidents our system can detect compared to governmental emergency systems, we

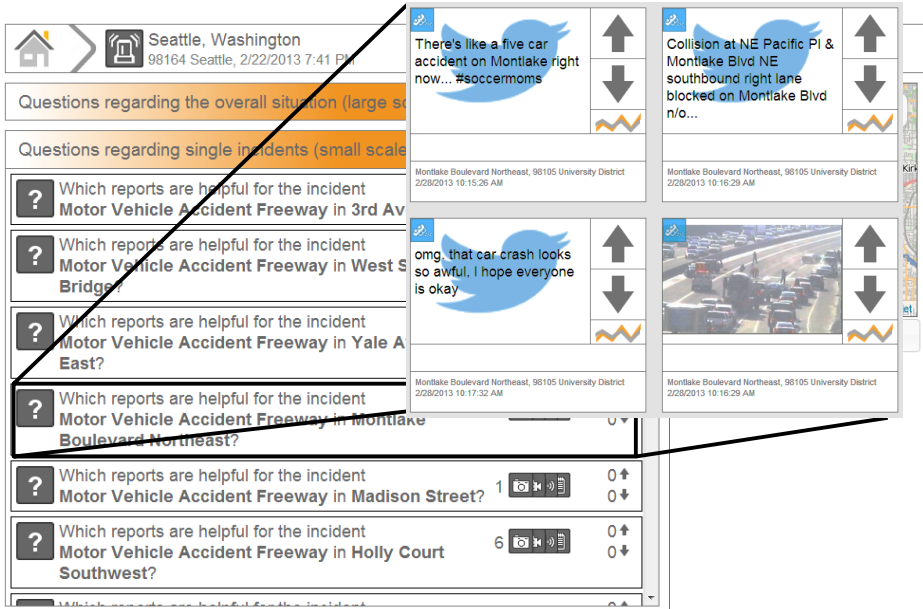


Fig. 2. Integration of classified microblogs into the Incident Classifier application

evaluated our predictions with the data from the Socrata test set. We correlated each of the 15 car accidents with the incidents from our system, if a spatial (150m) and temporal (+/-20min) matching applies. Using this approach we were able to detect all of the Socrata incidents with our approach. As the average number of tweets identified by our system for each car accident was around ten (with a minimum of only three tweets for one accident), this shows that our approach is capable of detecting incidents with only very few social media posts.

Performance. For our experiments, we have crawled the Twitter API as discussed above. In the scenario using data from Seattle and Memphis, there were around 100 Tweets per minute. Processing and classifying a bulk of 500 tweets using our trained SVM takes around seven seconds, which is about 14 milliseconds per Tweet.

5 Example Application

In [17], we introduced the idea of an information cockpit called Incident Classifier as a central access point for a decision maker to use different types of user-generated content for increasing the understanding of the situation at hand. Based on an aggregation algorithm we introduced in [16], we integrate the classified microblogs in the Incident Classifier. In this case, we apply spatio-temporal filtering and aggregation based on the incident type, e.g., the “car crash“ type.

In Figure 2, the aggregation of different incident related information to a small scale incident, including images extracted from referenced web pages in tweets, is shown. E.g., in this case, a picture of the incident as well as the number of involved cars are shown, thus, enabling a decision maker to get additional information about an incident that might be relevant.

6 Conclusion and Future Work

This paper contributes an approach that leverages information provided in microblogs for detection of small scale incidents. We showed how machine learning and semantic web technologies can be combined to identify incident related microblogs. With 89% detection accuracy, we outperform state-of-the-art approaches. Furthermore, our approach is able to precisely localize microblogs in space and time, thus, enabling the real-time detection of incidents.

With the presented approach, we are able to detect valuable information during crisis situations in the huge amount of information published in microblogs. In this case, additional and previously unknown information can be retrieved that could contribute to enhance situational awareness for decision making in daily crisis management. In the future, we aim at refining our approach, e.g., to use more sophisticated NLP techniques, exploring the capability of our approach to detect other types of events, as well as including larger sets of open government data in the evaluation.

Acknowledgements. This work has been partly funded by the German Federal Ministry for Education and Research (BMBF, 13N10712, 01jS12024), by the German Science Foundation (DFG, FU 580/2), and by EIT ICT Labs under the activities 12113 Emergent Social Mobility and 13064 CityCrowdSource of the Business Plan 2012.

References

1. Abel, F., Hauff, C., Houben, G.J., Stronkman, R., Tao, K.: Twitcident: Fighting Fire with Information from Social Web Stream. In: Proceedings of International Conference on Hypertext and Social Media (2012)
2. Agarwal, P., Vaithyanathan, R., Sharma, S., Shroff, G.: Catching the long-tail: Extracting local news events from twitter. In: Proceedings of the Sixth International Conference on Weblogs and Social Media, ICWSM 2012, Dublin, Ireland (2012)
3. Goolsby, R.: Lifting Elephants: Twitter and Blogging in Global Perspective. In: Social Computing and Behavioral Modeling, pp. 1–6. Springer, Heidelberg (2009)
4. Heim, P., Thom, D.: SemSor: Combining Social and Semantic Web to Support the Analysis of Emergency Situations. In: Proceedings of the 2nd Workshop on Semantic Models for Adaptive Interactive Systems SEMAIS. Springer, Heidelberg (2011)
5. Hienert, D., Wegener, D., Paulheim, H.: Automatic classification and relationship extraction for multi-lingual and multi-granular events from wikipedia. In: Detection, Representation, and Exploitation of Events in the Semantic Web. CEUR-WS, vol. 902, pp. 1–10 (2012)

6. Jadhav, A., Wang, W., Mutharaju, R., Anantharam, P.: Twitris: Socially Influenced Browsing. In: Demo at 8th International Semantic Web Conference on Semantic Web Challenge 2009, Washington, DC, USA (2009)
7. Krstajic, M., Rohrdantz, C., Hund, M., Weiler, A.: Getting There First: Real-Time Detection of Real-World Incidents on Twitter. In: Proceedings of 2nd IEEE Workshop on Interactive Visual Text Analytics (2012)
8. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.C.: Tedas: A twitter-based event detection and analysis system. In: 2011 11th International Conference on ITS Telecommunications (ITST), pp. 1273–1276. IEEE Computer Society (2012)
9. Manning, C.D., Raghavan, P., Schütze, H.: An Introduction to Information Retrieval, pp. 117–120. Cambridge University Press (2009)
10. Marcus, A., Bernstein, M.S., Badar, O., Karger, D.R., Madden, S., Miller, R.C.: Twitinfo: aggregating and visualizing microblogs for event exploration. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2011, pp. 227–236. ACM, New York (2011)
11. Mendes, P.N., Jakob, M., García-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. ACM (2011)
12. Okolloh, O.: Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information. *Participatory Learning and Action* 59, 65–70 (2008)
13. Paulheim, H., Frnkranz, J.: Unsupervised Feature Generation from Linked Open Data. In: International Conference on Web Intelligence, Mining, and Semantics, WIMS 2012 (2012)
14. Sakaki, T., Okazaki, M.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 851–860 (2010)
15. Schulz, A., Hadjakos, A., Paulheim, H., Nachtwey, J., Mühlhäuser, M.: A multi-indicator approach for geolocation of tweets. In: Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM (2013)
16. Schulz, A., Ortmann, J., Probst, F.: Getting user-generated content structured: Overcoming information overload in emergency management. In: Proceedings of 2012 IEEE Global Humanitarian Technology Conference (GHTC 2012), pp. 1–10 (2012)
17. Schulz, A., Paulheim, H., Probst, F.: Crisis Information Management in the Web 3.0 Age. In: Proceedings of the Information Systems for Crisis Response and Management Conference (ISCRAM 2012), pp. 1–6 (2012)
18. Strötgen, J., Gertz, M.: Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation* (2012)
19. Vieweg, S., Hughes, A.L., Starbird, K., Palen, L.: Microblogging during two natural hazards events: what twitter contribute to situational awareness. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2010, pp. 1079–1088. ACM, New York (2010)
20. Wanichayapong, N., Pruthipunyaskul, W., Pattara-Atikom, W., Chaovalit, P.: Social-based traffic information extraction and classification. In: 11th International Conference on ITS Telecommunications (ITST), pp. 107–112 (2011)
21. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques. Elsevier, Morgan Kaufman, Amsterdam, Netherlands (2005)

Ontology Adaptation upon Updates

Alessandro Solimando and Giovanna Guerrini

Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
Università di Genova, Italy
name.surname@unige.it

Abstract. Ontologies, like any other model, change over time due to modifications in the modeled domain, deeper understanding of the domain by the modeler, error corrections, simple refactoring or shift of modeling granularity level. Local changes usually impact the remainder of the ontology as well as any other data and metadata defined over it. The massive size of ontologies and their possible fast update rate requires automatic adaptation methods for relieving ontology engineers from a manual intervention, in order to allow them to focus mainly on high-level inspection. This paper, in spirit of the *Principle of minimal change*, proposes a fully automatic ontology adaptation approach that reacts to ontology updates and computes sound reformulations of ontological axioms triggered by the presence of certain preconditions. The rule-based adaptation algorithm covers up to *SRQL* DL.

1 Introduction and Motivations

Ontologies, like any other model, change over time and a revalidation of all data and metadata defined on top of the modified ontology is needed upon updates. Massive ontology size and fast update rate¹ call for automated support and adaptation algorithms. Despite the great attention devoted in the last ten years to ontology evolution [2,8], to the best of our knowledge there are no proposals in the literature coping with ontology adaptation upon updates. With similar motivations, an adaptation algorithm for a subset of *SPARQL* queries (with expressivity equivalent to union of Conjunctive Queries) in response to ontology updates is proposed in [7]. *Protégé*², one of the most complete ontology frameworks, does not support any kind of adaptation w.r.t. ontology updates: when a concept or a role is deleted, all the axioms referring it are removed as well. Even if there are cases in which this behavior is acceptable (*e.g.*, error corrections), there are others for which it is detrimental, for instance a modification of the modeling granularity of the ontology. In this scenario, a sound reformulation of axioms by means of super/sub concepts or roles is not only desirable but usually manually performed by the modeler. Additionally, in Artificial Intelligence (*Belief Revision*), knowledge deletion usually follows the *Principle of Minimal*

¹ An example is the *Gene Ontology* (<http://www.geneontology.org/>), with $\sim 416K$ axioms and $\sim 40K$ entities, daily updated (statistics for data-version 2013-02-22).

² Available here: <http://protege.stanford.edu/>

Change [6], which suggests that the amount of lost information should be as minimal as possible. Given that ontologies do not necessarily (explicitly) include all their logical consequences, also the implicit knowledge should be taken into account, as well as explicit one (that is, ontology axioms).

While a set of basic ontology changes can be easily defined, it is impossible to identify a set of complex changes without fixing the granularity level, *i.e.*, updates expressed as arbitrarily complex graph patterns (see [11], Section 3.2.1). In this proposal we consider the basic updates proposed by [4]: addition, deletion and update of entities (concepts and roles). Given that adding or updating entities do not reduce knowledge, and that ontology consistency can be tested using ontology reasoners, our adaptation algorithm focuses only on entity deletions.

In this paper, we propose an algorithm that, given an ontology and an entity (concept or role) to delete, scans for an equivalent, a super and a sub-entity and tries to reformulate the axioms involving the entity in question, with a rule-based approach. Our reformulated axioms are a fraction of the *implicit knowledge* of the ontology under update that would be lost by deleting all of the axioms involving the removed entity. An alternative would be to compute the closure (that is, complete inference of implicit knowledge) for the ontology prior to entity deletion. Due to its high computational cost and possible non-finiteness of the result, a suboptimal but less expensive approach is preferable for our target scenario, that is interactive modeling.

Even if the adaptation algorithm is completely automatic, it may not always be aligned with the modeler’s intention. For this reason, the present proposal has to be intended as an optional feature. When activated, it provides a preview of the changes to show the automatic adaptation effects. On this basis, the modeler can accept or ignore the proposed changes. In addition, a straightforward extension could be the possibility, for the modeler, to select the equivalent (resp. sub/super) entity for the reformulation, when different alternatives are available.

The contribution of the present paper can be summarized as follows: an automatic adaptation algorithm supporting up to *SRONTQ* expressivity, its correctness proof, and temporal complexity analysis (Section 3), an experimental evaluation of the percentage of adaptable entities and axioms on a dataset of real ontologies (Section 4). First, DL basics are introduced (Section 2), and the paper concludes discussing future work (Section 5).

2 Preliminaries

Our proposal covers up to *SRONTQ* Description Logic (DL), on top of which the *Ontology Web Language* (OWL2) [12] is defined. The notations and definitions used in this section are borrowed from [5]. An ontology is defined by a set of axioms and a set of entity names (signature), composed by three disjoint subsets: $N_{\mathcal{R}}$ for role names, $N_{\mathcal{I}}$ for individual names, $N_{\mathcal{C}}$ for concept names. These entities are defined by means of expressions. We have *Role expressions* $\mathbf{R} ::= U \mid N_{\mathcal{R}} \mid N_{\mathcal{R}}^-$, and *Concept expressions* $\mathbf{C} ::= N_{\mathcal{C}} \mid (\mathbf{C} \sqcup \mathbf{C}) \mid (\mathbf{C} \sqcap \mathbf{C}) \mid \neg \mathbf{C} \mid \top \mid \perp \mid \exists \mathbf{R}.\mathbf{C} \mid \forall \mathbf{R}.\mathbf{C} \mid \geq_n \mathbf{R}.\mathbf{C} \mid \leq_n \mathbf{R}.\mathbf{C} \mid \exists \mathbf{R}.\textit{Self} \mid \{N_{\mathcal{I}}\}$, with $n \geq 0$. For the

Table 1. Adaptation rules for concept deletion $DEL(C)$, where $B, C, C', D, E, F \in N_C$, $R \in N_R$ and $a \in N_I$

	Precondition	Rule
a.1	$C \equiv C'$, $C \in \text{signature}(\text{axiom})$	$\text{axiom} \rightarrow \text{axiom}[C/C']$
a.2	$C \sqsubseteq D$	$E \equiv \exists R.C \rightarrow E \sqsubseteq \exists R.D$
a.3	$C \sqsubseteq D$	$E \equiv \geq_n R.C \rightarrow E \sqsubseteq \geq_n R.D$
a.4	$C \sqsubseteq D$	$E \equiv C \sqcup F \rightarrow E \sqsubseteq D \sqcup F$
a.5	$C \sqsubseteq D$	$E \equiv C \sqcap F \rightarrow E \sqsubseteq D \sqcap F$
a.6	$C \sqsubseteq D$	$E \equiv \neg C \rightarrow \neg D \sqsubseteq E$
a.7	$C \sqsubseteq D$	$C(a) \rightarrow D(a)$
a.8	$C \sqsubseteq D$	$E \equiv \forall R.C \rightarrow E \sqsubseteq \forall R.D$
a.9	$B \sqsubseteq C$	$E \equiv \leq_n R.C \rightarrow E \sqsubseteq \leq_n R.B$
a.10	$B \sqsubseteq C$	$E \equiv C \sqcup F \rightarrow B \sqcup F \sqsubseteq E$
a.11	$B \sqsubseteq C$	$E \equiv C \sqcap F \rightarrow B \sqcap F \sqsubseteq E$
a.12	$B \sqsubseteq C$	$E \equiv \neg C \rightarrow E \sqsubseteq \neg B$
a.13	$B \sqsubseteq C$	$C \sqsubseteq E \rightarrow B \sqsubseteq E$

semantics associated with nominals, role and concept expressions the reader may refer to [5]. The set of axioms of an ontology, denoted with *Axioms*, is defined as $\text{Axiom} ::= \text{ABox} \cup \text{RBox} \cup \text{TBox}$. The reader may refer to [5] also for a detailed description of the different available axioms for *SRQIQ* DL, and to [10] for the definitions of ontology *interpretation* and ontology *satisfiability*. W.l.o.g. in the paper we will consider normalized ontologies in *Negation Normal Form* (NNF), with an application of *Structural Reduction* (SR), as shown in [10] (Subsection 5.3). SR introduces fresh concept names for (complex) concept expressions, thus letting us to easily refer to each concept expression by means of its associated concept name. Neither the SR nor the NNF are required for the application of our method. NNF, however, may increase the ratio of adapted axioms.

3 Algorithm

This section introduces the adaptation rules (Section 3.1), the rule-based adaptation algorithm (Section 3.2), the correctness proof for the given rules (Section 3.3), and the temporal complexity of the algorithm (Section 3.4).

3.1 Adaptation Rules

The adaptation rules are presented in Table 1 (rules for concepts) and Table 2 (rules for roles). We denote by $\text{axiom}[A/B]$ the alpha renaming of an axiom of entity A by entity B . A rule r is composed by a left hand side, $LHS(r)$, a right hand side, $RHS(r)$, and a precondition $prec(r)$. A rule is defined **applicable** iff $prec(r)$ is satisfied by at least one concept (resp. role). Given an ontology o and an entity e to delete, the LHS of a rule r is said to be **matching** iff an axiom in o exists that is equal to $LHS(r)$ modulo alpha renaming of C (resp. R)

Table 2. Adaptation rules for role deletion $DEL(R)$, where $E, C \in N_C$, $Q, R, R', S, T, T_i, T'_j \in N_{\mathcal{R}}$, with $m, n, p, q \geq 0$, and $a, b \in N_{\mathcal{I}}$

	Precondition	Rule
b.1	$R \equiv R'$, $R \in \text{signature}(\text{axiom})$	$\text{axiom} \rightarrow \text{axiom}[R/R']$
b.2	$Q \sqsubseteq R$	$T_0 \circ \dots \circ T_m \circ R \circ T'_0 \circ \dots \circ T'_p \sqsubseteq T$ $\rightarrow T_0 \circ \dots \circ T_m \circ Q \circ T'_0 \circ \dots \circ T'_p \sqsubseteq T$
b.3	$Q \sqsubseteq R$	$E \equiv \forall R.C \rightarrow E \sqsubseteq \forall Q.C$
b.4	$Q \sqsubseteq R$	$E \equiv \leq_n R.C \rightarrow E \sqsubseteq \leq_n Q.C$
b.5	$Q \sqsubseteq R$	$T \equiv R^- \rightarrow Q^- \sqsubseteq T$
b.6	$Q \sqsubseteq R$	$\text{Disjoint}(R, T) \rightarrow \text{Disjoint}(Q, T)$
b.7	$R \sqsubseteq S$	$R(a, b) \rightarrow S(a, b)$
b.8	$R \sqsubseteq S$	$E \equiv \exists R.C \rightarrow E \sqsubseteq \exists S.C$
b.9	$R \sqsubseteq S$	$E \equiv \exists R.\text{Self} \rightarrow E \sqsubseteq \exists S.\text{Self}$
b.10	$R \sqsubseteq S$	$E \equiv \geq_n R.C \rightarrow E \sqsubseteq \geq_n S.C$
b.11	$R \sqsubseteq S$	$T \equiv R^- \rightarrow T \sqsubseteq S^-$
b.12	$R \sqsubseteq S$	$T_0 \circ \dots \circ T_q \sqsubseteq R \rightarrow T_0 \circ \dots \circ T_q \sqsubseteq S$

with e , denoted with $LHS(r)[e]$. The **application** of an applicable rule r w.r.t. o and e rewrites any axiom of o matching $LHS(r)[e]$ into $RHS(r)[e']$, where e' is the selected entity for reformulation. It is worth noting that if a DL less expressive than \mathcal{SROIQ} is adapted, only a subset of the rules will be applicable, depending on the axioms and constructors available. For instance, for basic \mathcal{ALC} with *General Concept Inclusion* (i.e., $C \sqsubseteq D$), rules a.3, a.9, b.2, b.4, b.5, b.9, b.10, b.11, b.12 are not applicable.

3.2 Adaptation Algorithm

Algorithm 1 presents the adaptation algorithm for ontology updates. It takes as input the entity e to be deleted and the ontology o it belongs to. By means of function *computePrec*, the set of axioms related to e is computed, as well as a triple p consisting of a (nondeterministically chosen) equivalent, a sub and a super entity, if any (line 3). For each axiom a having e in its signature (line 4), it tests if the axiom matches the left hand side of the rule (line 5). At this point, function *satisfies* (line 6) checks if the current axiom is compatible with rule r and if the required element in p is not null. The reformulated axiom is inserted in o (line 7). Finally, all the axioms involving entity e are removed from o (line 8). Even if a preliminary classification phase is not required, it may increase the algorithm effectiveness. In what follows we give a toy example of ontology update, comparing the result of adaptation to classical deletion approach.

Example 1. Consider an ontology o consisting of these axioms and the obvious associated signature: $\text{Human} \equiv \exists \text{eats.Food}$, $\text{Food}(\text{cheese})$, $\text{Eater} \equiv \forall \text{eats.Food}$, $\perp \equiv \text{Plastic} \sqcap \text{Food}$, $\text{Uneatable} \equiv \neg \text{Eatable}$, $\text{Pizza} \sqsubseteq \text{Food}$, $\text{Food} \sqsubseteq \text{Eatable}$. Deleting Food concept from o with adaptation we obtain: $\text{Human} \sqsubseteq \exists \text{eats.Eatable}$, $\text{Eatable}(\text{cheese})$, $\text{Eater} \sqsubseteq \forall \text{eats.Eatable}$, $\text{Plastic} \sqcap$

Algorithm 1. Ontology Update Adaptation

```

1: function ONTOUPDATEADAPT(Entity  $e$ , Ontology  $o$ )
2:    $axioms = \emptyset$ 
3:    $p := \langle eq, sub, sup \rangle \leftarrow computePrec(e, axioms, o)$ 
4:   for  $a \in axioms$  do
5:     for  $r \in Rules$  .  $a = LHS(r)[e]$  do
6:       if  $satisfies(\langle a, e, e' \rangle, prec(r))$ ,  $e' \in \{eq, sub, sup\}$  then
7:          $Axioms(o) \leftarrow Axioms(o) \cup \{RHS(r)[e']\}$ 
8:       end if
9:     end for
10:  end for
11:   $Axioms(o) \leftarrow Axioms(o) \setminus axioms$ 
12: end function
13: function COMPUTEPREC(Entity  $e$ , Set  $axioms$ , Ontology  $o$ )
14:   $eq, sub, sup \leftarrow \epsilon$ 
15:  for  $a \in Axioms(o)$  .  $e \in signature(a)$  do
16:     $axioms \leftarrow axioms \cup \{a\}$ 
17:    if  $eq, sub, sup \neq \epsilon$  then
18:      break
19:    end if
20:    if  $a = e \equiv e'$  or  $a = e' \equiv e$  then
21:       $eq \leftarrow e'$ 
22:    else if  $a = e \sqsubseteq e'$  then
23:       $sup \leftarrow e'$ 
24:    else if  $a = e \supseteq e'$  then
25:       $sub \leftarrow e'$ 
26:    end if
27:  end for
28:  return  $\langle eq, sub, sup \rangle$ 
29: end function

```

$Pizza \sqsubseteq \perp$, $Pizza \sqsubseteq Eatable$, $Uneatable \equiv \neg Eatable$ (using rule a.2, a.7, a.8, a.11 and a.13, respectively). Without adaptation, instead, only the last axiom would be present in o after concept deletion.

3.3 Rules Correctness Proof

Before stating the proposition about the correctness of the adaptation rules we introduce some definitions and lemmata. For sake of brevity we will interchangeably refer to the axioms and their semantics, according to [5].

Definition 1. An axiom A_1 **entails** an axiom A_2 iff, for any interpretation I , $I \models A_2 \implies I \models A_1$, that is $A_2^I \subseteq A_1^I$.

Definition 2. An adaptation rule r is **sound** iff $\{LHS(r), prec(r)\}$ entails $RHS(r)$.

Lemma 1. $\forall C, D, F \in N_C$. $C \sqsubseteq D \implies C \sqcup F \sqsubseteq D \sqcup F$.

Proof. By considering the associated semantics the Lemma can be restated as $C^I \subseteq D^I \implies \underbrace{C^I \cup F^I}_{\alpha} \subseteq \underbrace{D^I \cup F^I}_{\beta}$. Assume that the preceding formula does

not hold, that is $\alpha \not\subseteq \beta$. This means $\exists x \in \beta$. $x \notin \alpha$, and requires that at least one of the following conditions holds:

- $x \in F^I$, but this implies $x \in \alpha$, resulting in a contradiction,

– $x \in C^I$, and thus this implies $C^I \subseteq D^I \implies x \in \alpha$, contradicting the hypothesis. \square

Lemma 2. $\forall C, D, F \in N_C . C \sqsubseteq D \implies C \sqcap F \sqsubseteq D \sqcap F$.

Proof. By considering the associated semantics the Lemma can be restated as $C^I \subseteq D^I \implies \underbrace{C^I \cap F^I}_{\alpha} \subseteq \underbrace{D^I \cap F^I}_{\beta}$. Assume that the preceding formula does not

hold, that is $\alpha \not\subseteq \beta$. This means $\exists x \in \beta . x \notin \alpha$. Note that $x \in \beta$ is equivalent to requiring that $x \in F^I \wedge x \in D^I$ holds. However, $x \in F^I \wedge x \notin \alpha \implies x \notin C^I$. Given that $x \in D^I$ holds, this contradicts the premise $C \sqsubseteq D$. \square

Lemma 3. $\forall C, D \in N_C . C \sqsubseteq D \implies \exists R.C \sqsubseteq \exists R.D$.

Proof. Assume that $\{x \mid \exists y \in C^I . \langle x, y \rangle \in R^I\} \not\subseteq \{x \mid \exists y \in D^I . \langle x, y \rangle \in R^I\}$ holds, that is, $\exists R.C \not\subseteq \exists R.D$. This requires that the following condition holds: $\exists \langle x, y \rangle \in R^I . y \in C^I \wedge y \notin D^I$. But, if such condition holds, then $C \not\subseteq D$, contradicting the premise. \square

Lemma 4. $\forall C, D \in N_C . C \sqsubseteq D \implies \forall R.C \sqsubseteq \forall R.D$.

Proof. Assume that $\{x \mid \forall \langle x, y \rangle . \langle x, y \rangle \in R^I \implies y \in C^I\} \not\subseteq \{x \mid \forall \langle x, y \rangle . \langle x, y \rangle \in R^I \implies y \in D^I\}$ holds, that is, $\forall R.C \not\subseteq \forall R.D$. This requires that the following condition holds: $(\exists x . \forall \langle x, y \rangle . \langle x, y \rangle \in R^I \implies y \in C^I) \wedge (\exists \bar{y} . \langle x, \bar{y} \rangle \in R^I \wedge \bar{y} \notin D^I)$. But, if this condition holds, then an \bar{y} exists and R^I is not empty. Therefore, since the left operand of the implication holds, then right operand also does. From this, we obtain $C^I \not\subseteq D^I$, contradicting the premise. \square

Proposition 1. *Adaptation rules application preserves ontology satisfiability.*

Proof. Ontology satisfiability is preserved because every adaptation rule is sound. We prove this for each rule separately:

- a.1 The proof directly follows from *Concept Equivalence* axiom definition.
- a.2 $E \equiv \exists R.C \rightarrow E \sqsubseteq \exists R.D$. $\exists R.C \sqsubseteq \exists R.D$ must hold: thanks to the rule precondition, $C \sqsubseteq D$, we can apply Lemma 3.
- a.3 $E \equiv \geq_n R.C \rightarrow E \sqsubseteq \geq_n R.D$. $\geq_n R.C \sqsubseteq \geq_n R.D$ must hold, but it is sufficient that $\{x \mid \exists y \in C^I . \langle x, y \rangle \in R^I\} \subseteq \{x \mid \exists y \in D^I . \langle x, y \rangle \in R^I\}$ holds. Thanks to the rule precondition, $C \sqsubseteq D$, we can apply Lemma 3.
- a.4 $E \equiv C \sqcup F \rightarrow E \sqsubseteq D \sqcup F$. $C \sqcup F \sqsubseteq D \sqcup F$ holds for Lemma 1 because $C \sqsubseteq D$ holds.
- a.5 $E \equiv C \sqcap F \rightarrow E \sqsubseteq D \sqcap F$. $C \sqcap F \sqsubseteq D \sqcap F$ holds for Lemma 2 because $C \sqsubseteq D$ holds.
- a.6 $E \equiv \neg C \rightarrow \neg D \sqsubseteq E$. $\neg D \sqsubseteq \neg C$ must hold: the semantics is $\Delta^I \setminus D^I \subseteq \Delta^I \setminus C^I$, but this contradicts $C \sqsubseteq D$.
- a.7 $C(a) \rightarrow D(a)$. $C(a) \implies D(a)$ is guaranteed by the rule precondition.
- a.8 $E \equiv \forall R.C \rightarrow E \sqsubseteq \forall R.D$. $E \equiv \forall R.C \implies E \sqsubseteq \forall R.D$ holds for Lemma 4 because $C \sqsubseteq D$ holds.

- a.9 $E \equiv_{\leq n} R.C \rightarrow E \sqsubseteq_{\leq n} R.B. \leq_n R.B \sqsubseteq_{\leq n} R.C$, but it is sufficient that $\{x \mid \exists y \in B^I . \langle x, y \rangle \in R^I\} \subseteq \{x \mid \exists y \in C^I . \langle x, y \rangle \in R^I\}$. Thanks to the rule precondition, $B \sqsubseteq C$, we can apply Lemma 3.
- a.10 $E \equiv C \sqcup F \rightarrow B \sqcup F \sqsubseteq E$. The proof for $B \sqcup F \sqsubseteq C \sqcup F$ is the dual of the one given in item (a.4).
- a.11 $E \equiv C \sqcap F \rightarrow B \sqcap F \sqsubseteq E$. The proof for $B \sqcap F \sqsubseteq C \sqcap F$ is the dual of the one given in item (a.5).
- a.12 $E \equiv \neg C \rightarrow E \sqsubseteq \neg B$. the proof for $\neg C \sqsubseteq \neg B$ is the dual of the one given in item (a.6).
- a.13 $C \sqsubseteq E \rightarrow B \sqsubseteq E$. The rule precondition, $B \sqsubseteq C$. By transitivity, this implies $B \sqsubseteq E$.
- b.1 The proof directly follows from *Role Equivalence* axiom definition.
- b.2 $\underbrace{T_0 \circ \dots \circ T_m \circ R \circ T'_0 \circ \dots \circ T'_p}_{\alpha} \sqsubseteq T \rightarrow \underbrace{T_0 \circ \dots \circ T_m \circ Q \circ T'_0 \circ \dots \circ T'_p}_{\beta} \sqsubseteq T$. assume that $\beta^I \not\subseteq \alpha^I$ holds. This requires that $\exists x_0, \dots, x_{m+p+3} . \langle x_0, x_1 \rangle \in T_0^I \wedge \dots \wedge \langle x_{m+1}, x_{m+2} \rangle \in Q^I \wedge \langle x_{m+p+2}, x_{m+p+3} \rangle \in T_p^I \wedge \langle x_{m+1}, x_{m+2} \rangle \notin R^I$. This contradicts $Q \sqsubseteq R$.
- b.3 $E \equiv \underbrace{\exists R.C}_{\alpha} \rightarrow E \sqsubseteq \underbrace{\forall Q.C}_{\beta}$. Assume that $\alpha^I \not\subseteq \beta^I$ holds. This requires that $\exists x . x \in \alpha^I \wedge x \notin \beta^I$, that is, $\exists x . ((\forall y . \langle x, y \rangle \in R^I \implies y \in C^I) \wedge (\exists y' . \langle x, y' \rangle \in Q^I \wedge y' \notin C^I))$. Given that $Q \sqsubseteq R$, if such y' exists, α cannot hold, leading to a contradiction.
- b.4 $T \equiv \underbrace{\leq_n R.C}_{\alpha} \rightarrow T \sqsubseteq \underbrace{\leq_n Q.C}_{\beta}$. Assume that $\alpha^I \not\subseteq \beta^I$. This requires that $\exists x . |\{y \mid y \in C^I \wedge \langle x, y \rangle \in R^I\}| \leq n \wedge |\{y \mid y \in C^I \wedge \langle x, y \rangle \in Q^I\}| > n$. This implies $|Q^I| > |R^I|$, contradicting $Q \sqsubseteq R$.
- b.5 $T \equiv R^- \rightarrow Q^- \sqsubseteq T$. Assume that $Q^{-I} \not\subseteq R^{-I}$. This requires that $\exists \langle x, y \rangle . \langle y, x \rangle \in Q^I \wedge \langle y, x \rangle \notin R^I$. This contradicts $Q^I \subseteq R^I$.
- b.6 $Disjoint(R, T) \rightarrow Disjoint(Q, T)$. Assume that $R^I \cap T^I = \emptyset \implies Q^I \cap T^I = \emptyset$ does not hold. This requires that $\exists \langle x, y \rangle \in Q^I \wedge \langle x, y \rangle \in T^I \wedge \langle x, y \rangle \notin R^I$ holds, but $\langle x, y \rangle \in Q^I \wedge \langle x, y \rangle \notin R^I$ contradicts $Q \sqsubseteq R$.
- b.7 $R(a, b) \rightarrow S(a, b)$. From $R \sqsubseteq S$ we have that $\forall \langle x, y \rangle . \langle x, y \rangle \in R \implies \langle x, y \rangle \in S$.
- b.8 $E \equiv \underbrace{\exists R.C}_{\alpha} \rightarrow E \sqsubseteq \underbrace{\exists S.C}_{\beta}$. Assume that $\alpha^I \not\subseteq \beta^I$. This requires that $\exists x . \exists y \in C^I . \langle x, y \rangle \in S^I \wedge \langle x, y \rangle \notin R^I$ holds. This contradicts $R \sqsubseteq S$.
- b.9 $E \equiv \underbrace{\exists R.Self}_{\alpha} \rightarrow E \sqsubseteq \underbrace{\exists S.Self}_{\beta}$. Assume that $\alpha^I \not\subseteq \beta^I$. This requires that $\exists x . \langle x, x \rangle \in S^I \wedge \langle x, x \rangle \notin R^I$ holds. This contradicts $R \sqsubseteq S$.
- b.10 $E \equiv \underbrace{\geq_n R.C}_{\alpha} \rightarrow E \sqsubseteq \underbrace{\geq_n S.C}_{\beta}$. Assume that $\alpha^I \not\subseteq \beta^I$. This requires that $\exists x . |\{y \mid y \in C^I \wedge \langle x, y \rangle \in R^I\}| \geq n \wedge |\{y \mid y \in C^I \wedge \langle x, y \rangle \in S^I\}| < n$. This implies $|R^I| > |S^I|$, contradicting $R \sqsubseteq S$.

- b.11 $T \equiv R^- \rightarrow T \sqsubseteq S^-$. Assume that $R^{-I} \not\subseteq S^{-I}$. This requires that $\exists \langle x, y \rangle \cdot \langle y, x \rangle \in R^I \wedge \langle y, x \rangle \notin S^I$, thus contradicting $R \sqsubseteq S$.
- b.12 $T_0 \circ \dots \circ T_q \sqsubseteq R \rightarrow T_0 \circ \dots \circ T_q \sqsubseteq S$. This immediately follows, by transitivity, from $R \sqsubseteq S$. \square

3.4 Temporal Complexity

Proposition 2. *The time complexity of the algorithm is in $\mathcal{O}(n)$, where n is the number of axioms of the input ontology o .*

Proof. *computePrecond* scans all the axioms of ontology o . For each of them it performs some comparison having a total cost of c_1 , so it has a cost of $n \cdot c_1$. The *for* statement of line 4 in Algorithm 1 is executed n times in the worst case (each axiom of the ontology refers to the entity in question). The *for* statement of line 5 is executed $c_2 = |\text{Rules}|$ times, where *Rules* is the set of adaptation rules. *satisfies* test requires a constant (c_3) time for checking the required conditions. Axiom rewriting and its insertion requires constant (c_4) time. The removal of old axioms requires constant time (c_5) too. The overall complexity is therefore equal to $n \cdot c_1 + c_2 \cdot c_3 \cdot c_4 \cdot n + c_5$, that belongs to $\mathcal{O}(n)$. \square

4 Experiments

In order to evaluate the practical applicability of our proposal we implemented a Java prototype based on the OWL API library³. In OWL API the axioms are immutable objects, and it supports only axiom addition and removal. Whenever possible, the rule application has been simulated with a pair of add and delete changes. In the other cases we employed Java Reflection for directly modifying the involved axiom. In addition to correctness, we also experimentally evaluated the coverage of OWL2 axioms and constructors of our set of rules. The dataset is presented in Table 3 (manual selection on the Web based on ontology size and DL expressivity).

Correctness. The developed proof-of-concept prototype has been used for testing correctness of our adaptation rules, the experimental counterpart of the proofs given in Section 3.3. More precisely, the test consists in taking as input a satisfiable ontology composed by the precondition and an axiom corresponding to the LHS of a rule r (modulo alpha renaming of the entity to delete). At this point, using *Hermit* reasoner (v1.3.7)⁴, we check the entailment of $RHS(r)[e]$.

Evaluation. An entity e is **adaptable** iff it satisfies at least one rule precondition, while an axiom a is **adaptable** iff it at least one rule r s.t. $LHS(r)[e] = a$ exists, in case $prec(r)$ holds w.r.t. e , the axiom is said **fully adaptable**. As an estimation of the practical effectiveness of our algorithm, we consider, for

³ Available here: <http://owlapi.sourceforge.net/>

⁴ Hermit and related information are available at <http://hermit-reasoner.com/>

Table 3. Dataset and coverage results presented in Section 4. Ontologies 14 and 15 are part of the dataset used by *Ontology Alignment Evaluation Initiative*, and are available at <http://oaei.ontologymatching.org/>. Coverage results are unaggregated and based on the data of Table 4. N/A means the ontology contains no roles/adaptable roles.

ID	DL	URI	Axioms	(C.1)	(C.2)	(C.3)	(C.4)	(C.2)*	(C.4)*
1.	ALUQ(D)	http://omv.ontaware.org/2009/09/OWLChanges	186	100.00	49.49	0.00	N/A	49.49	N/A
2.	SHIN(D)	http://ccdb.ucsd.edu/SAO/1.2	7767	100.00	17.92	97.22	76.17	17.95	76.17
3.	LEHI+(D)	http://swat.cse.lehigh.edu/onto/univ-bench.owl	243	97.67	35.11	36.00	45.16	37.10	45.16
4.	SHOIN(D)	http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine	747	94.81	67.88	84.62	66.96	88.29	66.96
5.	ALCO	http://purl.obolibrary.org/obo/ogms.owl	576	100.00	50.00	N/A	N/A	50.00	N/A
6.	ALQ(D)	http://owlodm.ontaware.org/OWL2	353	90.80	45.50	0.00	N/A	46.91	N/A
7.	SRIQ(D)	http://semananticscience.org/ontology/sio-core.owl	5043	100.00	48.07	100.00	93.55	48.43	93.55
8.	SHOIN	http://www.co-ode.org/ontologies/pizza/pizza.owl	939	100.00	22.14	75.00	100.00	22.15	100.00
9.	SHOIN(D)	http://sweet.jpl.nasa.gov/2.1/repSciUnits.owl	503	100.00	8.20	33.33	0.00	42.86	0.00
10.	SR	http://purl.obolibrary.org/obo/2011-11-03/haoo.owl	20027	96.22	38.53	0.00	N/A	50.47	N/A
11.	SROIF	http://purl.obolibrary.org/obo/ido.owl	3499	100.00	34.67	86.67	67.02	36.92	67.73
12.	SROIF	http://ontology.neuinfo.org/NIF/Dysfunction/NIF-Dysfunction.owl	5649	100.00	50.00	N/A	N/A	50.00	N/A
13.	SHIN(D)	http://aims.fao.org/aos/geopolitical.owl	23527	100.00	100.00	100.00	100.00	100.00	100.00
14.	S	http://human.owl	30364	99.82	49.84	0.00	N/A	49.84	N/A
15.	ALE	http://mouse.owl	11043	67.02	36.99	0.00	N/A	38.93	N/A
16.	ALCHOIN	http://owl.man.ac.uk/2005/07/sssw/people.owl	396	100.00	69.94	71.43	81.25	71.18	81.25
17.	SROIF	http://ontology.neuinfo.org/NIF/BiomaterialEntities/NIF-GrossAnatomy.owl	19849	99.75	39.62	18.18	100.00	39.62	100.00
18.	SROIN(D)	http://purl.obolibrary.org/obo/flu/dev/flu.owl	874	204	100.00	100.00	100.00	100.00	100.00
19.	ALCOIF	http://www.owl-ontologies.com/generations.owl	60	94.44	77.14	75.00	100.00	100.00	100.00
20.	AL(D)	http://protege.stanford.edu/plugins/owl/owl-library/ka.owl	404	100.00	57.03	0.00	N/A	57.18	N/A
21.	SROIF	http://ontology.neuinfo.org/NIF/BiomaterialEntities/NIF-Cell.owl	3508	100.00	46.73	0.00	N/A	46.73	N/A
22.	SOIN(D)	http://www.owl-ontologies.com/travel.owl	145	100.00	49.61	33.33	100.00	50.00	100.00
23.	ALUHN	http://www.hozo.jp/owl/YAMATO20120714.owl	4428	100.00	51.69	95.63	11.67	51.69	11.67
24.	ALHIF(D)	http://purl.org/net/ontology/beer.owl	165	84.48	42.86	22.22	100.00	42.86	100.00
25.	SH(D)	http://msi-ontology.sourceforge.net/ontology/NMR.owl	1096	100.00	50.00	N/A	N/A	50.00	N/A
26.	SHIF(D)	http://www.mada.kth.se/~mehrana/Delegation.owl	103	63.16	40.91	80.00	81.25	40.91	81.25
27.	ALCRQ(D)	http://www.biomodels.net/kisao/KISAO	2044	100.00	42.70	100.00	92.23	42.82	92.47
28.	ALCO	http://purl.obolibrary.org/obo/omrse.owl	99	100.00	50.00	0.00	N/A	50.00	N/A

each ontology in our dataset, the following scenario: we simulate the deletion of each single entity, in isolation, and we take into account the percentage of adaptable ones (i.e., such that another entity suitable for reformulation exists). For each of these adaptable entities, we also inspect how many axioms involving them would be adapted instead of simply deleted. For this reference scenario we defined *Coverage* measure as: **(C.1)** the percentage of adaptable concepts (resp. roles **(C.3)**) out of the total number of concepts (resp. roles), and **(C.2)** the percentage of adaptable axioms w.r.t. the deleted concept (resp. role, **(C.4)**) out of the number of axioms to be deleted (that is, presenting the deleted entity in their signature). The (C.)^{*} variants count the fully adaptable axioms, and evaluate the completeness of our adaptation rules (the complement of the fully adaptable axioms is not supported by our rules).

In Table 3 the coverage for each ontology in isolation is reported (computed from the raw data of Table 4), while the result considering the dataset as a whole ontology is the following: (C.1) 93.247%, (C.2) 41.757%, (C.2^{*}) 44.185%, (C.3) 73.647%, (C.4) 79.63%, (C.4^{*}) 80.847%. Table 3 shows that 10 out of 12 of the worst performing ontologies w.r.t. role coverage ((C.3), (C.4) and (C.4)^{*}) are expressed in a DL missing role hierarchy constructs (identified by letter *H* in the DL name). Without role hierarchy constructs only role equality can be used for adaptation, thus reducing the number of adaptable roles. Concept coverage (C.1) presents, instead, high values (above 60%) for all the considered ontologies, independently from the DL they are expressed with. This is not surprising because concept hierarchy constructs are available for DLs at least as expressive as \mathcal{AL} . On the contrary, coverage results for concept rules w.r.t. OWL2 axioms and constructors seem to be unrelated to either the underpinning DL or the ontology size (in terms of number of axioms and/or entities). For instance, the ontologies with worst values for (C.2)^{*} are 2. (*SHLN*(\mathcal{D})), 8. (*SHOIN*(\mathcal{D})) 11. (*SROIF*) 3. (*AL \mathcal{E} HI* + (\mathcal{D})) and 15. (*AL \mathcal{E}*), with very different number of concepts and axioms (Table 4). Similarly, among the best results for (C.2)^{*} the expressivity ranges from $\mathcal{AL}(\mathcal{D})$ to *SROIN*(\mathcal{D}), again with varying number of axioms and concepts. Ideally the proposal should adapt all the axioms: (C.2)^{*}, in particular, is far from this result, but it is well known that OWL2, despite being based on *SROIQ*, adds new constructors and axioms, that are derivable from *SROIQ* ones (they do not add expressive power). For example, *Concept Disjointness* axiom (i.e., *Disjoint*(*C*, *D*), with *C*, *D* $\in \mathcal{N}_C$) is only a shortcut for $C \sqcap D \sqsubseteq \perp$ ⁵. Our prototype strictly applies the rules of Table 1 and Table 2, so it cannot directly process the axioms and constructors not available in *SROIQ* DL, thus diminishing the number of adaptable axioms.

5 Future Work

The paper represents, to the best of our knowledge, the first proposal for ontology adaptation upon updates. In addition, the algorithm is totally automatic and supports ontology expressivity up to *SROIQ*, on top of which OWL2 is defined.

⁵ Refer to [10], Chapter 9, for further examples and details.

The present paper could be extended in several directions. The set of adaptation rules is a preliminary proposal, we plan to further enrich it in order to increase the coverage rate reported in Section 4 and to consider reasonable alternatives for each single rule (*e.g.*, sound alternatives for a.8 could be $C \sqsubseteq D, E \equiv \forall R.C \rightarrow \forall R.D \sqsubseteq E$ or $B_0 \dots B_n \sqsubseteq C, E \equiv \forall R.C \rightarrow E \sqsubseteq \forall R. \bigsqcup_{i=0}^n B$). We also plan to consider the integration of anonymous entities (*e.g.*, to use \top as superclass). Another possible extension is the integration of a complex set of update primitives (*e.g.*, concept merge and split), such as [3]. The relationship between DL updates and *Belief Revision* has been investigated [9], we plan to further investigate it w.r.t. our proposal. As an alternative to the random or manual selection of the appropriate sub/super-entity for reformulation, we plan to integrate a selection based on the computation of the least common subsumers/most specific concepts [1]. We also intend to improve our prototype up to a full support of OWL2. Our final goal will be a *Protégé* plugin, from which we hope to receive feedbacks from the community of ontology engineers and practitioners. The experimental evaluation will also be strengthened with an extended ontology dataset and temporal profiling of the prototype.

References

1. Baader, F.: Least Common Subsumers and Most Specific Concepts in a Description Logic with Existential Restrictions and Terminological Cycles. In: International Joint Conference on Artificial Intelligence, vol. 18, pp. 319–324 (2003)
2. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: Classification and survey. *Knowl. Eng. Rev.* 23(2), 117–152 (2008)
3. Hartung, M., Groß, A., Rahm, E.: COnto-Diff: Generation of Complex Evolution Mappings for Life Science Ontologies. *Journal of Biomedical Informatics* (2012)
4. Hartung, M., Groß, A., Rahm, E.: Rule-based Generation of Diff Evolution Mappings between Ontology Versions. *CoRR abs/1010.0122* (2010)
5. Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible SROIQ. In: Principles of Knowledge Representation and Reasoning – KR 2006, pp. 57–67 (2006)
6. Katsuno, H., Mendelzon, A.O.: Propositional Knowledge Base Revision and Minimal Change. *Artificial Intelligence* 52(3), 263–294 (1991)
7. Kondylakis, H., Plexousakis, D.: Ontology Evolution: Assisting Query Migration. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 331–344. Springer, Heidelberg (2012)
8. Noy, N., Klein, M.: Ontology Evolution: Not the same as Schema Evolution. *Knowledge and Information Systems* 6(4), 428–440 (2004)
9. Ribeiro, M.M., Wassermann, R., Antoniou, G., Flouris, G., Pan, J.: Belief Contraction in Web-Ontology Languages. In: Workshop on Ontology Dynamics, IWOD (2009)
10. Rudolph, S.: Foundations of Description Logics. In: Polleres, A., d’Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 76–136. Springer, Heidelberg (2011)
11. Stojanovic, L.: Methods and Tools for Ontology Evolution. Ph.D. thesis, University of Karlsruhe (2004)
12. W3C as Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer (2009), <http://www.w3.org/TR/owl2-primer/>

Caching and Prefetching Strategies for SPARQL Queries

Johannes Lorey and Felix Naumann

Hasso Plattner Institute,
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
{johannes.lorey, felix.naumann}@hpi.uni-potsdam.de

Abstract. Linked Data repositories offer a wealth of structured facts, useful for a wide array of application scenarios. However, retrieving this data using SPARQL queries yields a number of challenges, such as limited endpoint capabilities and availability, or high latency for connecting to it. To cope with these challenges, we argue that it is advantageous to cache data that is relevant for future information needs. However, instead of retaining only results of previously issued queries, we aim at retrieving data that is potentially interesting for subsequent requests in advance. To this end, we present different methods to modify the structure of a query so that the altered query can be used to retrieve additional related information. We evaluate these approaches by applying them to requests found in real-world SPARQL query logs.

1 Introduction

Linked Data sources offer a wealth of information about a multitude of topics, including geo-location facts¹, government data², or cross-domain information³. The SPARQL Protocol and RDF Query Language (SPARQL) has become the de facto query language to retrieve this information from publicly available endpoints.

However, whereas in principle SPARQL facilitates various information needs by defining complex query constructs, typical public SPARQL endpoint characteristics, such as high latency, limited hardware resources, or unavailability restrict on-demand utilization of the data. In this work, we propose a novel approach for discovering and aggregating potentially relevant data for subsequent user requests based on previous query patterns by rewriting these preceding queries. To this end, we introduce some fundamental concepts of SPARQL in Sec. 1.1 and outline the contribution and organization of this paper in Sec. 1.2.

1.1 SPARQL Preliminaries

One central concept of a SPARQL query is that of a triple pattern $T = (s, p, o) \in (V \cup U) \times (V \cup U) \times (V \cup U \cup L)$ with V being a set of variables, U being a set of

¹ <http://linkedgedata.org>

² <http://data-gov.tw.rpi.edu/wiki>

³ <http://dbpedia.org>

URIs, and L being a set of literals. A SPARQL query contains a number of graph patterns P_1, P_2, \dots , which are defined recursively: (i) A valid triple pattern T is a graph pattern. (ii) If P_1 and P_2 are graph patterns, then P_1 AND P_2 , P_1 UNION P_2 , and P_1 OPTIONAL P_2 are graph patterns [12].

In terms of relational operations, AND represents an inner join of two graph patterns, UNION denotes their union, and OPTIONAL indicates a left outer join between P_1 and P_2 . In addition, AND takes precedence over UNION, and OPTIONAL is always left-associative [12]. While UNION and OPTIONAL are reserved keywords in actual SPARQL queries to indicate the corresponding connection between two graph patterns, the AND keyword is omitted.

We call P the *parent graph pattern* of graph patterns P_1, \dots, P_n , when it has the form $P = P_1 \oplus \dots \oplus P_n$, where \oplus is any one of the introduced keywords. Whereas \oplus is symmetric for AND and UNION, it is not for OPTIONAL [12]. In any SPARQL query Q , the graph pattern P with no non-trivial parent graph pattern is referred to as the *query pattern* P_Q .

Typically, curly braces are used to delimit graph patterns: $\{P\}$. Here, if such a delimited graph pattern P represents a conjunction of multiple triple patterns T_i , i.e., its form is $P = T_1$ AND $T_2 \dots$ AND T_n , we call P a *basic graph pattern* (BGP). While there is the notion of empty graph patterns in SPARQL, we consider only non-empty graph patterns.

In our work, we focus on SELECT queries. In general, SPARQL allows to limit the projection to certain variables listed after the SELECT statement. Moreover, graph patterns may contain so-called *filter conditions* indicated by the keyword FILTER. In a filter condition, specific restrictions, such as regular expressions or inequality relations, can be placed on resources to modify the scope of the selection of a query.

1.2 Contribution and Paper Organization

As with traditional Web search, SPARQL is suitable for different information needs, such as data exploration or navigational querying. However, in contrast to simple keyword-based queries, the well-defined structure and constructs of SPARQL queries allow for more fine-grained expressions representing the user intent. We build on this notion and present methods to modify queries in order to retrieve information relevant for subsequent related requests. Further, we discuss features of queries and query sequences influencing the suitability and effects of these modification methods.

By locally materializing results potentially relevant for subsequent queries, overall query execution time can be tremendously decreased for a client. Moreover, if the SPARQL endpoint becomes unavailable for a period of time, thus disallowing access to the remote information, a replicated subset of the data may be utilized instead. On the other hand, a SPARQL endpoint may also leverage knowledge about what data may be relevant for future queries, e.g., by storing it in main memory. The main trade-offs for this caching approach are increased storage requirements and potential data staleness, which may be negligible given the concrete scenario.

The remainder of this paper is structured as follows: In Sec. 2, we discuss related work for this work and point out differences in our approach. In Sec. 3, we introduce a means to group SPARQL queries by matching similarly structured requests. We then present our different query rewriting approaches in Sec. 4 and present an evaluation using real-world query logs in Sec. 5. Finally, we conclude this paper in Sec. 6 and highlight some future work in this context.

2 Related Work

Related work for this paper draws mainly from two fields: (i) *semantic caching and prefetching*, e.g., techniques to retain previously fetched data or retrieve data relevant for subsequent queries, and (ii) *query relaxation*, e.g., parsing and modifying a user query to discover relevant resources.

2.1 Semantic Caching and Prefetching

Semantic caching builds on the idea of maintaining a local copy of retrieved data that can be used for subsequent queries. As with traditional caching, one of the major motivations for semantic caching is to reduce the transmission overhead when retrieving data over a network link. Conventional approaches, such as tuple or page caching, usually retain fetched data based on either temporal or frequency aspects, e.g., by prioritizing least-recently or most-frequently used items. Such techniques also exist for SPARQL query result caching [11,15]. Compared to these works, semantic caching employs more fine-grained information to characterize data, e.g., in order to establish variable-sized semantic regions containing related tuples [4] or detect data items with similar geolocation information [13].

Closely related to semantic caching and our work is *prefetching*. Instead of simply retaining tuples retrieved previously, prefetching allows to gather data that is potentially useful for subsequent queries based on semantic information derived from past queries or the overall system state. In computer architecture design, prefetching is usually employed to request instructions that are anticipated to be executed in the future and place them in the CPU cache. For information retrieval, query prefetching typically assumes a probabilistic model, e.g., considering temporal or spatial features [6,9]. However, there have been no attempts to prefetch RDF data based on the structure of sequential related SPARQL queries within and across query sessions.

2.2 Query Relaxation

Query relaxation aims at discovering interesting related information based on a user request. For keyword queries, this process is sometimes referred to as query expansion and has been a major research topic in the field of information retrieval (cf. [3]). In contrast to query refinement, which aims at increasing precision by restricting the scope of a query, the goal of query relaxation or query expansion

is to improve the recall in retrieval effectiveness. To this end, often precomputed metadata, such as language models, is utilized.

There exist a number of projects for implementing query relaxation for retrieving Linked Data. The authors of [8] suggest logical relaxations based on ontological metadata. In contrast, the approach in [7] relies on precomputed similarity tables for attribute values whereas in [5] the authors utilize a language model derived from the knowledge base.

In comparison, our rewriting strategies are not targeted at increasing recall when executing a single query, but instead aim to retrieve additional data related to future queries. Moreover, we do not assume any knowledge of the data source itself or of metadata describing it. Thus, while most previous approaches require at least some precomputation, our approach can be used ad-hoc solely by analyzing and modifying queries issued during run-time.

3 Query Matching

For different aspects of our work, we need to identify and cluster similarly-structured queries. To this end, we introduce our bottom-up query pattern matching approach based on matching similar triple patterns they contain.

3.1 Triple Pattern Distance

To match triple patterns, we determine their distance by accumulating the distance scores between their parts, i.e., between the two subjects, predicates, and objects, respectively. Here, if two triple pattern parts are both variables, their distance is 0. In case they are both URIs or both literals, their distance is the normalized Levenshtein distance of the respective strings. Otherwise, e.g., when one subject is a variable and the other a URI, the distance is 1. More formally, if x_1, x_2 are either the subjects, predicates, or objects of two triple patterns T_1, T_2 , respectively, we define their symmetric distance score $\Delta(x_1, x_2)$ as

$$\Delta(x_1, x_2) := \begin{cases} 0, & \text{if } x_1 \in V \wedge x_2 \in V \\ \frac{\textit{levenshtein}(x_1, x_2)}{\max(|x_1|, |x_2|)}, & \text{if } (x_1 \in U \wedge x_2 \in U) \\ & \vee (x_1 \in L \wedge x_2 \in L) \\ 1, & \text{else.} \end{cases} \quad (1)$$

We determine the overall distance $\Delta(T_1, T_2) = \Delta(T_2, T_1)$ of two triple patterns T_1, T_2 by aggregating the individual triple pattern parts distance scores $\Delta(s_1, s_2)$, $\Delta(p_1, p_2)$, $\Delta(o_1, o_2)$ as follows: In case $\Delta(s_1, s_2) = \Delta(p_1, p_2) = \Delta(o_1, o_2) = 0$, we define $\Delta(T_1, T_2) := 0$. Otherwise, there exists a minimum triple pattern part distance score $\min_{\Delta} := \min(\Delta(s_1, s_2), \Delta(p_1, p_2), \Delta(o_1, o_2))$ with $\min_{\Delta} > 0$. In this case, the triple pattern distance score is defined as

$$\Delta(T_1, T_2) := \lceil \Delta(s_1, s_2) \rceil + \lceil \Delta(p_1, p_2) \rceil + \lceil \Delta(o_1, o_2) \rceil - (1 - \min_{\Delta}) \quad (2)$$

This way, a distance $\Delta(T_1, T_2) \leq 1$ always indicates a dissimilarity in at most one triple pattern part, whereas for two non-equal triple pattern parts $1 < \Delta(T_1, T_2) \leq 2$, and a distance score $\Delta(T_1, T_2) > 2$ hints at differences between the two subjects, predicates, and objects.

Consider the two basic graph patterns BGP_1 and BGP_2 in Listing 1 and Listing 2, respectively, where the line numbers serve as identifiers for the included triple patterns. Here, the most similar triple pattern for T_1 in BGP_2 can be determined by computing $\min(\Delta(T_1, T_4), \Delta(T_1, T_5), \Delta(T_1, T_6))$, which results in $\Delta(T_1, T_5) = (\lceil 0 \rceil + \lceil 0 \rceil + \lceil \frac{12}{16} \rceil - \frac{4}{16}) = 0.75$. For T_2 , the minimum value is $\Delta(T_2, T_6) = (\lceil 1 \rceil + \lceil 0 \rceil + \lceil 0 \rceil - 0) = 1$, and for T_3 it is $\Delta(T_3, T_4) = (\lceil 0 \rceil + \lceil \frac{5}{14} \rceil + \lceil \frac{5}{9} \rceil - \frac{9}{14}) \approx 1.36$. Thus, the most similar triple patterns for T_1, T_2, T_3 in BGP_2 are T_5, T_6 , and T_4 , respectively.

1	?city1	rdfs:label	"Paris"@fr	.
2	?person	?relationWith	?city1	.
3	:Auguste_Comte	foaf:givenName	"Auguste"	.

Listing 1. Basic Graph Pattern Example BGP_1

4	:Auguste_Comte	foaf:surname	"Comte"	.
5	?city2	rdfs:label	"Montpellier"@en	.
6	:Auguste_Comte	?association	?city2	.

Listing 2. Basic Graph Pattern Example BGP_2

3.2 Basic Graph Pattern Matching

Using the triple pattern distance scores, we can now determine matchings between basic graph patterns. In our work, finding this matching is equivalent to deriving a perfect (complete) bipartite cover with minimum cost between the triple patterns of the two individual basic graph patterns where the cost is determined by the triple pattern distance $\Delta(T_i, T_j)$. Obviously, a perfect matching of triple patterns is only possible for a complete bipartite graph, i.e., for two BGPs with the same number of triple patterns. If this is not the case, in order to generate a biclique of triple patterns we pad the basic graph pattern containing fewer elements using dummy triple patterns T_ε so that for any triple pattern T the score $\Delta(T, T_\varepsilon) = \Delta(T_\varepsilon, T) := \infty$.

As optimal solutions for generating maximal matchings can be determined in polynomial time and the input size (i.e., the number of triple patterns in the two BGPs) is reasonably small, we choose the well-known Hungarian Method [10] to create an assignment with minimum cost. Furthermore, we assign a maximum cost threshold to all derived matchings of triple patterns. Here, we consider only triple pattern matchings with cost $\Delta(T_i, T_j) \leq 1$, i.e., all matched triple patterns

are either identical or differ in at most one of non-variable subject, predicate, or object, respectively. The cost for triple pattern matchings with higher cost, i.e., matchings (T_i, T_j) with $\Delta(T_i, T_j) > 1$ is set to ∞ .

The score $\Delta(BGP_1, BGP_2)$ of derived complete matchings $M_T \subset \{(T_i, T_j) | (T_i, T_j) \in BGP_1 \times BGP_2\}$ is defined as:

$$\Delta(BGP_1, BGP_2) := \frac{\sum_{(T_i, T_j) \in M_T} \Delta(T_i, T_j)}{|M_T|} \quad (3)$$

If the result of the Hungarian method for M_T contains any individual triple pattern matching with infinite cost, $\Delta(BGP_1, BGP_2)$ is also infinite. In this case, no complete matching with only finite triple pattern distance scores can be determined between BGP_1 and BGP_2 . Assuming such a complete matching containing only valid triple pattern matchings exists, this BGP matching would have been the result of the algorithm as its cost would be $< \infty$.

Conversely, if the algorithm determines an optimal matching with infinite cost, any other matching with cost $< \infty$ cannot be complete as the algorithm does not terminate before discovering a maximal matching. Given our setting of a complete bipartite graph (or biclique), any maximal matching is always bound to be a complete matching.

Applying this approach on the basic graph patterns illustrated in Listing 1 and Listing 2 to determine a complete matching with minimum cost yields the same triple pattern matchings listed earlier. Thus, the optimal matching $\{(T_1, T_5), (T_2, T_6), (T_3, T_4)\}$ has cost $\frac{0.75+1+\infty}{3}$, i.e., BGP_1 and BGP_2 cannot be matched to one another.

3.3 Query Pattern Matching

Real-world SPARQL query patterns can be more complex than simple basic graph patterns, i.e., they may contain multiple recursively layered BGPs connected using the UNION or OPTIONAL keyword as introduced in Sec. 1.1. Along the lines of the previous subsection, where we defined BGP matching using the contained triple patterns, our matching approach for general query patterns is based on recursively matching the (basic) graph patterns they consist of.

Due to the recursive structure of SPARQL queries Q_1, Q_2 , we can only try to match two basic graph patterns BGP_i, BGP_j to one another if these BGPs reside at the same recursion level of the query patterns P_{Q_1}, P_{Q_2} , respectively. Additionally, the two basic graph patterns and any of their parent graph patterns also need to be connected by the same keyword to other (basic) graph patterns at their respective recursion level. If these two conditions are met, we say that BGP_i, BGP_j can be *aligned* to each other. More generally, if all (basic) graph patterns contained in a parent graph pattern P_1 can be aligned to at least one (basic) graph pattern of another parent graph pattern P_2 and vice versa, P_1, P_2 can also be aligned to each other.

Consider the following three SPARQL group graph patterns P_1, P_2 , and P_3 with

$$\begin{aligned} P_1 &:= BGP_1 \text{ OPTIONAL } (BGP_2 \text{ UNION } BGP_3), \\ P_2 &:= BGP_4 \text{ OPTIONAL } (BGP_5 \text{ UNION } BGP_6), \\ P_3 &:= BGP_7 \text{ OPTIONAL } BGP_8, \end{aligned}$$

where BGP_1, \dots, BGP_8 are basic graph patterns. Here, BGP_1 can be aligned to BGP_4 or BGP_7 . Additionally BGP_2 or BGP_3 can be aligned to either BGP_5 and BGP_6 , respectively. However, BGP_8 cannot be aligned to any other basic graph pattern in P_1 or P_2 , because the recursion level of BGP_8 is different from those of $BGP_2, BGP_3, BGP_5, BGP_6$, and OPTIONAL is not symmetric, i.e., $BGP_7 \text{ OPTIONAL } BGP_8 \neq BGP_8 \text{ OPTIONAL } BGP_7$. Thus, while P_1 and P_2 can be aligned to each other, P_3 cannot be aligned to either of them.

We use a bottom-up approach to try to match two query patterns P_{Q_1}, P_{Q_2} to one another. Similarly to the matching method introduced in the previous subsection, we try to derive a complete matching with minimal (finite) cost between all basic graph patterns of P_{Q_1} and P_{Q_2} that can be aligned to each other. Here, the cost between two basic graph patterns is determined by their distance score $\Delta(BGP_1, BGP_2)$.

If for any BGP in P_{Q_1} or P_{Q_2} no alignment is possible or its matching has infinite cost, P_{Q_1} or P_{Q_2} cannot be matched. If we derive a complete matching for all basic graph patterns, we check whether the parent graph patterns of all pairs of matched BGPs can be aligned to each other. If this is the case, we continue checking the alignment of the parent graph patterns until we reach the query pattern.

In case no alignment is possible, the two query patterns P_{Q_1} and P_{Q_2} cannot be matched. Conversely, two query patterns P_{Q_1} and P_{Q_2} can be matched if they can be aligned to each other and a complete matching with finite cost can be established between all basic graph patterns BGP_i and BGP_j in P_{Q_1} and P_{Q_2} , respectively. Canonically, any graph pattern can always be matched to itself.

To group structurally similar queries, we introduce the notion of *query clusters*. All queries in a query cluster can be matched to all other queries within the cluster, i.e., there exists a pairwise complete matching with finite cost between all BGPs of any two queries in the same cluster. Note that query clusters may be overlapping, i.e., a query can be element of multiple query clusters.

4 Query Augmentation

The main motivation of our work lies in retrieving information for SPARQL queries that is also relevant for subsequent related requests. Beyond basic caching, we argue for prefetching results: Here, we attempt to modify a query to retrieve additional data potentially relevant for future information needs. In this section, we motivate and illustrate different approaches for modifying queries accordingly.

4.1 Augmentation Concepts

We call the process of modifying the query contents *query augmentation* to emphasize that the results retrieved by issuing the original query are included in the result set for the modified query. In other words, the matches for the unmodified query form a subgraph of the matches for the augmented query. In this work, we are typically interested in retrieving additional information related to a *central concept*, namely the subject (i.e., either a variable or URI) occurring most often in the query pattern. Moreover, we require a central concept to be either part of the projection if it is a variable or to influence the selection if it is a URI. We assume that a URI influences the selection if at least one triple pattern, in which the URI is the subject, contains a projection variable.

Our intuition of query augmentation builds on concepts from information retrieval. For example, in traditional keyword-based search engines, a user might be unaware of the most suitable string pattern to enter to retrieve all relevant results at once. However, in several iterations the user may choose to refine the initial query based on retrieved results. In Linked Data terms, a user might query for more detailed information about a certain resource or for similar information of related resources after analyzing preliminary results, thus incrementally modifying the initial query.

In the remainder of this section, we introduce the different augmentation strategies we have implemented while emphasizing the particular requirements for their application. We illustrate the effect of applying these strategies on Query 1. Put simply, for the central concept French philosopher `:Auguste_Comte` this query retrieves the birth place(s) located in France alongside all influences on him that died in Paris. Table 1 lists all bindings retrieved by issuing Query 1 against the public DBpedia SPARQL endpoint⁴ currently containing DBpedia 3.8 data.

```
PREFIX      : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?birthPlace ?influence WHERE {
    :Auguste_Comte dbo:birthPlace ?birthPlace .
    ?birthPlace   dbo:country      :France      .
    :Auguste_Comte dbo:influencedBy ?influence .
    ?influence    dbo:deathPlace   :Paris      .
}
```

Query 1. Example of a SPARQL query

⁴ <http://dbpedia.org/sparql>

Table 1. All results for Query 1 in DBpedia 3.8

birthPlace	influence
:Montpellier	:Jean-Baptiste_Say
:Montpellier	:Émile_Durkheim

4.2 Exploratory Augmentation

In *exploratory augmentation*, we query for additional facts that are available for the central concept. The idea of exploratory augmentation is that a user might be interested in more information for a specific resource. However, this augmentation strategy is also helpful if the initial result set is empty, e.g., because of misspelled or ambiguous vocabulary terms (e.g., `foaf:img` and `foaf:Image`).

Potentially, there may also exist certain divergences between ontological information assumed by the user and the vocabulary used in actual data. For example, we discovered that although a number of properties are used frequently for instances of certain types in DBpedia, they are not defined in the ontology (e.g., `dbo:anthem` for `dbo:Country`). On the other hand, there are also a number of defined properties that are rarely used in instance data for the corresponding classes (e.g., `dbo:depth` for `dbo:Place`) [1].

This augmentation is applied by adding a triple pattern to the query, where the subject is the central concept and predicate as well as object are unique variables. Moreover, these two variables are added to the projection, thus evaluating all facts in which the central concept is subject. We highlight the corresponding changes in the resulting Query 2 by underlining modified or added sections. An excerpt of the extended result set for Query 2 is listed in Tab. 2.

```

PREFIX      : <http://dbpedia.org/resource/>
PREFIX      dbo: <http://dbpedia.org/ontology/>

SELECT ?p ?o ?birthPlace ?influence WHERE {
  :Auguste_Comte dbo:birthPlace   ?birthPlace .
  ?birthPlace   dbo:country      :France      .
  :Auguste_Comte dbo:influencedBy ?influence .
  ?influence    dbo:deathPlace   :Paris      .
  :Auguste Comte ?p              ?o
}

```

Query 2. Exploratory augmentation of Query 1

4.3 Template Augmentation

For the *template augmentation* approach, we first need to identify the cluster the current query belongs to as discussed in Sec. 3.3 by matching the current

Table 2. Some results for Query 2 in DBpedia 3.8 (191 results in total)

p	o	birthPlace	influence
rdf:type	dbo:Person	:Montpellier	:Jean-Baptiste_Say
dbo:birthDate	1798-01-19	:Montpellier	:Émile_Durkheim
dbo:notableIdea	:Positivism	:Montpellier	:Jean-Baptiste_Say
dbo:influenced	:Karl_Marx	:Montpellier	:Émile_Durkheim
...			

query pattern to those of preceding requests. If we determine a matching with finite cost, we require this matching to be non-trivial, i.e., to include at least one match between two non-identical basic graph patterns. This is true if the overall distance score of the query pattern matching is greater than 0, i.e., if any two matched basic graph patterns contain a pair of triple patterns (T_i, T_j) for which either the subjects, predicates, or objects differ.

A matching between two query patterns P_1 and P_2 meeting these requirements allows us to construct a *query template*: Initially, the template query string is identical to the current query string. We then introduce unique variables to replace all differing triple pattern parts included in the matching, i.e., either the non-matching subject, predicate, or object of triple patterns (T_i, T_j) for which $0 < \Delta(T_i, T_j) \leq 1$. Here, if the same resource in a BGP is replaced more than once, we use the same unique variable to ensure consistency.

Instead of issuing many similarly structured queries with only little variance, e.g., by using a crawler, a query template instead retrieves all relevant information using only a single query. A possible query template for Query 1 is illustrated in Query 3. Here, a specific resource has been replaced by a variable, which has also been added to the projection of the query in the **SELECT** statement. As indicated in Tab. 3, the result set contains the previous bindings as well as information about other persons with similar properties, e.g., about poet Paul Fort or mathematician Pierre de Fermat.

```

PREFIX      : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?s ?birthPlace ?influence WHERE {
  ?s      dbo:birthPlace ?birthPlace .
  ?birthPlace dbo:country      :France .
  ?s      dbo:influencedBy ?influence .
  ?influence dbo:deathPlace   :Paris .
}

```

Query 3. Template augmentation of Query 1, float=h

Table 3. Some results for Query 3 in DBpedia 3.8 (109 results in total)

s	birthPlace	influence
:Auguste_Comte	:Montpellier	:Jean-Baptiste_Say
:Auguste_Comte	:Montpellier	:Émile_Durkheim
:Paul_Fort	:Reims	:Paul_Verlaine
:Pierre_de_Fermat	:Beaumont-de-Lomagne	:François_Viète
...		

4.4 Type Augmentation

If class membership information in the knowledge base is available for the central concept, exploiting this ontological data can help in discovering information for related resources. In *type augmentation* we identify the `rdf:type` of the central concept and retrieve data for the instances belonging to the same classes by replacing the central concept with a unique variable throughout the query.

The goal of type augmentation is similar to that of template augmentation, i.e., querying information about different related resources. Whereas in template augmentation this relatedness is solely determined by the context of the replaced triple pattern part, in type augmentation it is derived by exploiting both structural and ontological information. Assuming the central concept is identical to the triple pattern part replaced for matching BGPs, all bindings retrieved using type augmentation are also retrieved using template augmentation. However, for type augmentation the connection between different resources may be stronger than for template augmentation given the type information of the resources.

According to the RDF Schema⁵, a resource may be instance of multiple classes, where these classes may either be unrelated or reside at different levels of the same type hierarchy. Therefore, RDF resources may be instances of very generic types, such as `owl:Thing`. Hence, one challenge for type augmentation lies in determining a suitable class for which instance data is retrieved, especially without assuming any a priori knowledge of the underlying ontology.

A number of techniques can be employed to gather this data, e.g., using multiple preliminary queries to construct a simple type hierarchy or utilizing aggregate functions, such as `COUNT`, to generate heuristics about the distribution of different types. In our approach, we introduce a `FILTER NOT EXISTS` to exclude all those (generic) types that have (more specific) subclasses. By doing so, we assume that the endpoint supports SPARQL 1.1 expressions and all resources are instances of at least one leaf node in the type hierarchy. If this is not the case, we exclude the filter condition.

Query 3 illustrates the result of applying type augmentation on the reference query, i.e., by introducing the new triple patterns regarding `rdf:type` information, exchanging the central concept in all other triple patterns, and applying the filter condition. Whereas the results for template augmentation also include bindings for `?s` that are (only) instances of rather generic classes such as `dbo:Person`,

⁵ <http://www.w3.org/TR/rdf-schema>

the results for Query 3 listed in Tab. 4 include resources that are (also) instances of specific subclasses, e.g., `dbo:Philosopher`.

```

PREFIX      : <http://dbpedia.org/resource/>
PREFIX  dbo: <http://dbpedia.org/ontology/>
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?s ?birthPlace ?influence WHERE {
  :Auguste_Comte rdf:type ?type .
  ?s              rdf:type ?type .
  ?s              dbo:birthPlace ?birthPlace .
  ?birthPlace    dbo:country      :France .
  ?s              dbo:influencedBy ?influence .
  ?influence      dbo:deathPlace  :Paris .
  FILTER NOT EXISTS {?t1 rdfs:subClassOf ?type}
}

```

Query 4. Type augmentation of Query 1

Table 4. Some results for Query 4 in DBpedia 3.8 (397 results in total)

s	birthPlace	influence
:Auguste_Comte	:Montpellier	:Jean-Baptiste_Say
:Auguste_Comte	:Montpellier	:Émile_Durkheim
:Jean-Paul_Sartre	:Paris	:Maurice_Merleau-Ponty
:René_Descartes	:Descartes,_Indre-et-Loire	:Marine_Mersenne
...		

4.5 Holistic Augmentation

The intuition of *holistic augmentation* is that the scope of SPARQL queries can be broadened by removing certain triple patterns they contain. However, to ensure that the result set of this modified query still contains all results of the original one, the removed parts must not contain variables essential to the projection or selection of a query. In other words, the variables in the `SELECT` statement still need to be present in the modified query so that they may be bound to an RDF term in a graph matching.

Typically, if we select a triple pattern to remove from the basic graph pattern BGP_i that contains it, we also remove the same triple pattern from any other basic graph pattern BGP_j in the query that can be matched to BGP_i

as described in Sec. 3.3. We call such a removal *valid*, if applying it on a valid SPARQL query results in a valid SPARQL query, i.e., if all projection variables are referenced in at least one remaining triple pattern of the query pattern. Note that there exist queries for which no valid triple pattern removal is possible, e.g., queries containing only one triple pattern.

To identify the most suitable triple pattern to remove from a query, we utilize the *variable counting* heuristic introduced in [14]. Essentially, this heuristic is based on the assumption that unbound subjects are more selective than unbound objects, which in turn are more selective than unbound predicates. Hence, generally a triple pattern with an unbound predicate, but bound subject and object matches fewer RDF statements in a knowledge base than a triple pattern with either only an unbound subject or object, i.e., is more selective. Also, a triple pattern with two or three unbound parts is less selective than a triple pattern with only one or two unbound parts, respectively. Thus, the least selective triple pattern is the one containing only variables.

In any query pattern there can be more than one triple pattern with maximum selectivity. In this case, we select an arbitrary one for removal. If this removal is not valid, we check whether a valid removal can be achieved for a different triple pattern with same or lesser selectivity. We continue until we have either exhaustively checked all triple patterns or discovered a validly removable triple pattern. In the latter case, we modify the query by deleting the triple pattern from the parent basic graph pattern and any other basic graph pattern this BGP can be matched to within the query.

Removing a highly-selective query triple pattern not essential for the projection mostly assists in situations where (i) the query is too restrictive, (ii) the query contains invalid statements, or (iii) the data or ontology in the knowledge base is inconsistent, e.g., as described in [1]. One of the two most selective triple pattern in Query 1 is crossed out in Query 5, thereby indicating its removal in this augmented query. Table 5 lists some results for Query 5.

```

PREFIX      : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?birthPlace ?influence WHERE {
  :Auguste_Comte dbo:birthPlace    ?birthPlace .
  ?birthPlace    dbo:country        :France      .
  :Auguste_Comte dbo:influencedBy ?influence   .
  ?influence   dbo:deathPlace    :Paris      .
}

```

Query 5. Holistic Augmentation of Query 1

Table 5. Some results for Query 5 in DBpedia 3.8 (10 results in total)

birthPlace	influence
:Montpellier	:Adam_Smith
:Montpellier	:David_Hume
:Montpellier	:Jean-Baptiste_Say
:Montpellier	:Émile_Durkheim
	...

5 Evaluation

To evaluate the applicability of our prefetching strategies for SPARQL queries, we analyzed parts of the DBpedia 3.6 and LinkedGeoData (LGD) query logs provided for the USEWOD 2013 data challenge [2]. The log files contain a number of requests received by the respective public SPARQL endpoints and were collected for different dates in 2011. While the specific metadata provided in the Apache Common Log Format differs slightly for the individual services, all requests contain the sender’s anonymized IP address and a timestamp in addition to the actual query.

5.1 Methodology

We use the timestamp information to provide a meaningful segmentation for successive queries by introducing *query sessions*. A query session is a chronologically ordered sequence of at least two queries issued sequentially by the same user (represented by a unique IP address) over a period of time. We call query sessions *homogeneous*, if they contain only queries belonging to the same cluster, i.e., any query in a session can be matched to any other query in the same session. Otherwise, we call this query session *heterogeneous*.

We exploit statistical features and metadata extracted from the query logs format for delimiting a query session. Specifically, we restrict the duration of a session by comparing the timestamps of issued queries with that of the initial query of a session. Once this difference exceeds a certain threshold, we assume a new query session has started. Depending on the analyzed dataset, we put an upper limit on the number of queries a single session may contain, thereby splitting a single query sequence into separate sessions if the number of successive queries is greater than this limit.

Matchings for triple patterns can easily be transformed into triples by applying the corresponding variable bindings. By materializing these triples for the original, unmodified query and for all queries generated by applying the augmentation strategies introduced in Sec. 4, we maintain individual result set caches. We then evaluate how many triples generated for bindings of subsequent queries are already contained in the individual caches. If we determine that a newly generated triple is contained in a cache, we consider this a *cache hit*. Note that the set of cached triples for each augmentation strategy includes at least the cached triples for the original query.

We use OpenLink Virtuoso Open-Source Edition version 6.1.3 as SPARQL endpoint containing the English DBpedia 3.6 dataset and ontology released on January 17, 2011, and the LinkedGeoData dump released on April 26, 2011 in separate named graphs. For each augmentation strategy, we restrict the number of retrieved results for performance reasons to a maximum of 100,000 using the LIMIT keyword for SPARQL.

5.2 DBpedia 3.6 Query Logs

The requests included in the DBpedia 3.6 query logs exhibit timestamps in hourly resolution, e.g., [24/Jan/2011 01:00:00 +0100]. In our analysis, only successive queries from the same user with identical timestamps may belong to the same session. However, as this heuristic introduces some vagueness regarding the contents of a session, e.g., by ignoring session timeouts, we also limit the maximum number of queries belonging to any single session to 25. Notice that by applying this conservative restriction, we potentially lower the number of cache hits in our approach, which most likely increases for longer sessions. In summary, once a user query with a different timestamp is detected or we discover more than 25 successive queries, we assume a new session has started.

Consequently, we base our evaluation on 288 query sessions for which we were able to retrieve results for at least one contained query. Of these query sessions, 176 (61%) were homogeneous. On average, the query sessions contained around 21 queries. In around 34% of all query sessions, we could not identify any cache hits. We assume that this observation is due to the small total result set size for these sessions: The sessions with no cache hits result in only about 100 generated triples compared to around 3,300 triples for sessions with cache hits. There are two possible reasons for this: (i) Our local SPARQL endpoint did not contain the entire DBpedia 3.6 corpus and (ii) some queries did not yield any results even when executed against the public DBpedia endpoint (e.g., because of syntax errors).

For all sessions with at least one cache hit, we illustrate the total number of cache hits in relation to the total number of generated (unique) triples for all unmodified queries in a session in Fig. 1. Each marker represents the best augmentation strategy resulting in the most cache hits for this session. If for none of the augmentation strategies the number of hits was greater than the cache hits of the original query, the marker “no augmentation” is used.

Overall, our findings indicate that caching the results of the first, un-augmented query of a session yields the most amount of cache hits in about 38% of all analyzed query sessions. The results for type and template augmentation have the most cache hits in 28% and 23% of the sessions, respectively, whereas applying exploratory and holistic augmentation on the first session query results in the most cache hits only for 7% and 4% of all sessions.

When considering homogeneous query sessions only, type augmentation yields the most cache hits in 39% of all sessions, followed by no augmentation (30%), template augmentation (28%), exploratory and holistic augmentation (1% each). Notice that we discovered a number of homogeneous query sessions containing

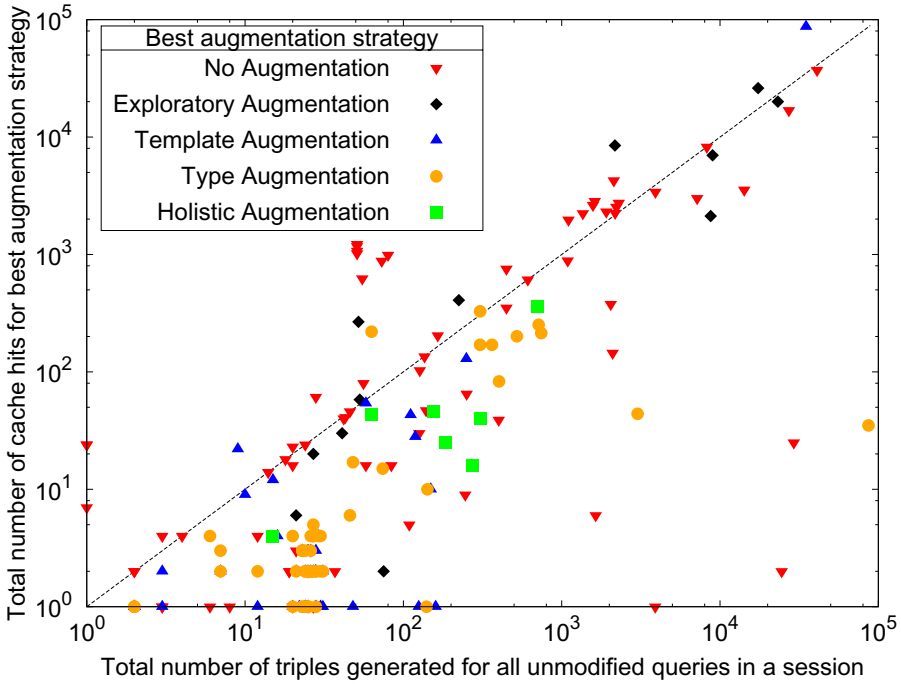


Fig. 1. Best augmentation strategy when caching results of the first query in a session in the DBpedia 3.6 log files

the exact same query multiple times. For example, this is the case if only one triple is generated for all queries in a session and identified as cache hit repeatedly as represented by markers located on the y-axis. Obviously, if the exact same query is issued over again within the course of a session, no additional cache hits can be generated when applying an augmentation strategy.

The mean number of cache hits is highest for exploratory augmentation (4,541 cache hits) and lowest for type augmentation (33 cache hits) when considering only those sessions where these two augmentation strategies yield the most cache hits. On average, in all sessions with markers above the diagonal (indicated by the dashed line in Fig. 1), each generated triple represents a cache hit at least once during the course of the query session.

We also evaluated how caching the result of every (augmented) query influenced the number of cache hits for subsequent (unmodified) queries in a session. While the number of query sessions with no cache hits dropped to around 24%, for those query sessions with cache hits we observed comparable results to the ones illustrated. Hence, we assume that by analyzing the first query, a suitable caching strategy can be determined for all subsequent queries of the same session. For example, for homogeneous query sessions applying template or type

augmentation on the first query most likely results in the same augmented query as applying it on any of the subsequent session requests.

In general, due to the large number of homogeneous query sessions in the DBpedia query logs, type and template augmentation appear to be the most successful among the augmentation strategies. Given the large number of resources in DBpedia and our restriction on the maximum number of results, we are impairing the success of these strategies to some extent, as many potentially relevant facts are simply not retrieved. Without this restriction, these prefetching strategies should yield even better results.

On the other hand, the amount of query sessions benefiting most from holistic or exploratory augmentation is limited. This might stem from the apparently small number of queries issued by human users, towards which these strategies are targeted. Moreover, in more than half the query sessions (55%) holistic augmentation could not be applied as no valid triple pattern removal was possible. Naturally, in these cases the number of cache hits for holistic augmentation equals the one of not applying any augmentation.

5.3 LinkedGeoData Query Logs

As the timestamps provided for the queries in the LGD logs are precise to the second, we delimit query sessions in these logs more accurately by introducing a session timeout and maximum session duration. If for a query from a specific user we cannot discover another query from the same user within 10 minutes time, we assume the identified query is the last one in a session. Overall, we delimit a query session by restricting its duration to a maximum of 60 minutes, its session timeout to 10 minutes, and its maximum number of queries to 50 (whichever comes first).

As with the DBpedia query logs, we analyzed only those query sessions in which we determined at least one query for which we were able to generate a result as described above, i.e., we based our evaluation on 440 query sessions. This time, for only 9% of these query sessions no cache hits could be discovered at all. The analysis of which augmentation strategy resulted in the most cache hits for the remaining 424 query sessions is illustrated in Fig. 2.

For the LGD logs, caching the results of unmodified queries resulted in the most hits (48% of all query sessions), followed by exploratory augmentation (26%), template augmentation (15%), type and holistic augmentation (5% each). For heterogeneous query sessions (55% of all query sessions), exploratory augmentation is the best augmentation strategy (30%), followed by template augmentation (19%), holistic augmentation (7%), and type augmentation (6%).

We also discovered that the number of mean cache hits was much higher than the ones for the DBpedia log files: For those sessions that benefited from template augmentation the average number of cache hits was highest with 72,917 and lowest for type augmentation with 48,320. On the other hand, the average session length was comparatively small with only 12 queries per session. This could be because the LGD log queries were actually issued by human users (as opposed to crawlers or other software agents). Intuitively, this would also

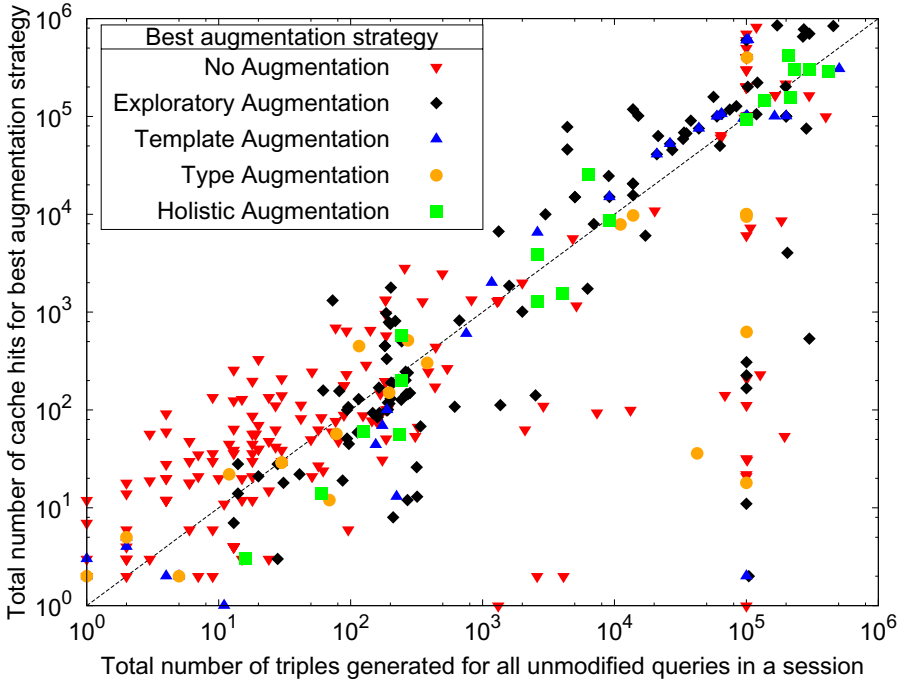


Fig. 2. Best augmentation strategy when caching results of the first query in a session in the LinkedGeoData log files

explain why the majority of query sessions are heterogeneous: Whereas software agents use somewhat hard-coded HTTP requests to retrieve Linked Data, human users are more flexible when issuing queries, e.g., through the LGD SPARQL web interface⁶.

Again, caching the result of every query in a session had only little impact on the choice of augmentation strategy, which is to be expected considering the small mean number of queries in a session. However, the percentage of query sessions not benefiting from any caching decreased slightly to 6%. In general, the cached results for the queries can almost always be used for subsequent queries in a session for the LGD logs. Again, our local SPARQL endpoint might have not contained all data available in the public SPARQL endpoint. However, the impact on our results is negligible as we were able to generate query results for the vast majority of sessions (75%) and cache hits in almost all of these (91%).

6 Conclusion and Outlook

In this paper, we have presented a number of approaches to modify SPARQL queries with the goal of retrieving additional results that are potentially relevant

⁶ <http://linkedgedata.org/sparql>

for subsequent requests of the same query session. We evaluated how often and how many of these additional results are actually relevant by identifying query sessions that can exploit this prefetched data in subsequent requests compared to only caching the results of the unmodified query.

While the majority of all analyzed query sessions benefited from caching the results of the unmodified or augmented first query (around 66% for DBpedia and 91% for LinkedGeoData), the most suitable augmentation strategy (if any) differs from session to session. For example, large-scale homogeneous query sessions can exploit a cache containing similar data of related resources, e.g., as generated by applying template and type augmentation.

On the other hand, human users might retrieve more “diverse” information, e.g., specific facts about a resource as generated by applying exploratory augmentation. Overall, whereas not all augmentation strategies can be applied on certain queries (e.g., holistic augmentation for queries containing only one triple pattern), combining different strategies during the course of a query session might still prove advantageous for such diversified scenarios.

In future work, we aim at capturing the intent in a query session more accurately, e.g., by using more fine-grained approaches than that of a central concept. Based on this, more customized augmentation strategies are conceivable, e.g., by analyzing individual resources contained in a query. Finally, we want to be able to discover the most suitable augmentation strategy or a combination of those strategies based on the contents of a query session and thus adjust other query sessions accordingly.

References

1. Abedjan, Z., Lorey, J., Naumann, F.: Reconciling ontologies and the web of data. In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), Maui, HI, USA, pp. 1532–1536 (October 2012)
2. Berendt, B., Hollink, L., Luczak-Rösch, M., Möller, K.H., Vallet, D.: USE-WOD2013 – 3rd international workshop on usage analysis and the web of data. In: 10th Extended Semantic Web Conference (ESWC) – Semantics and Big Data, Montpellier, France (2013)
3. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.* 44(1), 1:1–1:50 (2012)
4. Dar, S., Franklin, M.J., Jónsson, B.T., Srivastava, D., Tan, M.: Semantic data caching and replacement. In: Proceedings of the International Conference on Very Large Databases (VLDB), Bombay, India, pp. 330–341 (1996)
5. Elbassuoni, S., Ramanath, M., Weikum, G.: Query relaxation for entity-relationship search. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part II. LNCS*, vol. 6644, pp. 62–76. Springer, Heidelberg (2011)
6. Fagni, T., Prego, R., Silvestri, F., Orlando, S.: Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM Transactions on Information Systems* 24(1), 51–78 (2006)
7. Hogan, A., Mellotte, M., Powell, G., Stampouli, D.: Towards fuzzy query-relaxation for RDF. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012. LNCS*, vol. 7295, pp. 687–702. Springer, Heidelberg (2012)

8. Hurtado, C.A., Poullovassilis, A., Wood, P.T.: Query relaxation in RDF. In: Spaccapietra, S. (ed.) *Journal on Data Semantics X*. LNCS, vol. 4900, pp. 31–61. Springer, Heidelberg (2008)
9. Jonassen, S., Cambazoglu, B.B., Silvestri, F.: Prefetching query results and its impact on search engines. In: *Proceedings of the ACM International Conference on Information Retrieval (SIGIR)*, Portland, OR, USA, pp. 631–640 (2012)
10. Kuhn, H.W.: The hungarian method for the assignment problem. *Naval Research Logist. Quarterly* 2(1-2), 83–97 (1955)
11. 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, Part II*. LNCS, vol. 6089, pp. 304–318. Springer, Heidelberg (2010)
12. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)* 34(3), 16:1–16:45 (2009)
13. Ren, Q., Dunham, M.H.: Using semantic caching to manage location dependent data in mobile computing. In: *Proceedings of the International Conference on Mobile Computing and Networking*, Boston, MA, United States, pp. 210–221 (2000)
14. Stocker, M., Seaborne, A., Bernstein, A., Kiefer, C., Reynolds, D.: SPARQL basic graph pattern optimization using selectivity estimation. In: *Proceedings of the International World Wide Web Conference (WWW)*, New York, NY, USA, pp. 595–604 (2008)
15. Yang, M., Wu, G.: Caching intermediate result of SPARQL queries. In: *Proceedings of the International World Wide Web Conference (WWW)*, Hyderabad, India, pp. 159–160 (2011)

Characterising Citations in Scholarly Documents: The CiTalO Framework

Angelo Di Iorio¹, Andrea Giovanni Nuzzolese^{1,2}, and Silvio Peroni^{1,2}

¹ Department of Computer Science and Engineering, University of Bologna, Italy
{diiorio,nuzzoles,essepuntato}@cs.unibo.it

² STLab-ISTC Consiglio Nazionale delle Ricerche, Italy

Abstract. The reasons why an author cites other publications are varied: an author can cite previous works to gain assistance of some sort in the form of background information, ideas, methods, or to review, critique or refute previous works. The problem is that the best possible way to retrieve the nature of citations is very time consuming: one should read article by article to assign a particular characterisation to each citation. In this paper we propose an algorithm, called *CiTalO*, to infer automatically the function of citations by means of Semantic Web technologies and NLP techniques. We also present some preliminary experiments and discuss some strengths and limitations of this approach.

Keywords: CiTO, CiTalO, OWL, WordNet, citation function, semantic publishing.

1 Introduction

The academic community lives on bibliographic citations. First of all, these references are *tools for linking* research. Whenever a researcher writes a paper she/he uses bibliographic references as pointers to related works, to sources of experimental data, to background information, to standards and methods linked to the solution being discussed, and so on. Similarly, citations are *tools for disseminating* research. Not only on academic conferences and journals. Dissemination channels also include publishing platforms on the Web like blogs, wikis, social networks. More recently, semantic publishing platforms are also gaining relevance [15]: they support users in expressing semantic and machine-readable information. From a different perspective, citations are *tools for exploring* research. The network of citations is a source of rich information for scholars and can be used to create new and interesting ways of browsing data. A great amount of research is also being carried on sophisticated visualisers of networks of citations and powerful interfaces allowing users to filter, search and aggregate data. Finally, citations are *tools for evaluating* research – e.g. quantitative metrics on bibliographic references are commonly used for measuring the importance of a journal (the *impact factor*) or the scientific productivity of an author (the *h-index*).

This work begins with the basic assumption that all these activities can be radically improved by exploiting the actual nature of citations. Let us consider

citations as means for evaluating research. Could a paper that is cited many times with negative reviews be given a high score? Could a paper containing several citations of the same research group be given the same score of a paper with heterogeneous citations? How can a paper cited as plagiarism be ranked? These questions can be answered by looking at the nature of the citations, not only their existence. On top of such characterisation, it will also be possible to automatically analyse the pertinence of documents to some research areas, to discover research trends and the structure of communities, to build sophisticated recommenders and qualitative research indicators, and so on.

There are in fact ontologies for describing the nature of citations in scientific research articles and other scholarly works. In the Semantic Web community, the most prominent one is *CiTO* (*Citation Typing Ontology*)¹ [12]. CiTO is written in OWL and is connected to other works in the area of semantic publishing. It is then a very good basis for implementing sophisticated services and for integrating citational data with linked data silos.

The goal of this paper is to present a novel approach to automatically annotate citations with properties defined in CiTO. We present an algorithm and its implementation, called *CiTalO* (from merging the words **CiTO** and *al gorithm*), that takes as input a sentence containing a reference to a bibliographic entity and infers the function of that citation by exploiting Semantic Web technologies and Natural Language Processing (NLP) techniques. The tool is available online at <http://wit.istc.cnr.it:8080/tools/citalo>.

We also present some preliminary tests on a small collection of documents, that confirmed some strengths and weaknesses of such approach. The research direction looks very promising and the CiTalO infrastructure is flexible and extensible. We plan to extend the current set of heuristics and matching rules for a wide practical application of the method.

The paper is then structured as follows. In Section 2 we introduce previous works on classification of citations. In Section 3 we describe our algorithm introducing its structure and presenting the technologies (NLP tools, sentiment analysis procedures, OWL ontologies) we used to develop it. In Section 4 we present the outcome of the algorithm run upon some scientific documents and we discuss those results in Section 5. Finally, in Section 6, we conclude the paper sketching out some future works.

2 Related Works

The automatic analysis of networks of citations is gaining importance in the research community. Copestake *et al.* [4] present an infrastructure called SciBorg that allows one to automatically extract semantic characterisations of scientific texts. In particular, they developed a module for discourse and citation analysis based on the approach proposed by Teufel *et al.* [17] called *Argumentative Zoning* (AZ). AZ provides a procedural mechanism to annotate sentences of an article according to one out of seven classes of a given annotation scheme (i.e.

¹ CiTO: <http://purl.org/spar/cito>

background, own, aim, textual, contrast, basis and other), thus interpreting the intended authors' motivation behind scientific content and citations.

Teufel *et al.* [18] [19] study the *function* of citations – that they define as “author’s reason for citing a given paper” – and provide a categorisation of possible citation functions organised in twelve classes, in turn clustered in *Negative*, *Neutral* and *Positive* rhetorical functions. In addition, they describe the outcomes of some tests involving hundreds of article in computational linguistics (stored as XML files), several human annotators and a machine learning (ML) approach for the automatic annotation of citation functions. Their approach is quite promising; however the agreement between humans (i.e. $K = 0.72$) is still higher than the one between the humans and the ML approach (i.e. $K = 0.57$).

Jorg [9] introduces an analysis of the ACL Anthology Networks² and identifies one hundred fifty *cue verbs*, i.e. verbs usually used to carry important information about the nature of citations: *based on, outperform, focus on, extend*, etc. She maps cue verbs to classes of citation functions according to the classification provided by Moravcsik *et al.* [10] and makes the bases to the development of a formal citation ontology. This works actually represent one of the sources of inspiration of *CiTO* (the *Citation Typing Ontology*) developed by Peroni *et al.* [12], which is an ontology that permits the motivations of an author when referring to another document to be captured and described by using Semantic Web technologies such as RDF and OWL.

Closely related to the annotation of citation functions, Athar [1] proposes a sentiment-analysis approach to citations, so as to identify whether a particular act of citing was done with positive (e.g. praising a previous work on a certain topic) or negative intentions (e.g. criticising the results obtained through a particular method). Starting from empirical results Athar *et al.* [2] expand the above study and show how the correct sentiment (in particular, a negative sentiment) of a particular citation usually does not emerge from the citation sentence – i.e. the sentence that contains the actual pointer to the bibliographic reference of the cited paper. Rather, it actually becomes evident in the last part of the *context window*³ [14] in consideration.

Hou *et al.* [8] use an alternative approach to understand the importance (seen as a form of positive connotation/sentiment) of citations: the citation counting in text. Paraphrasing the authors, the idea is that the more a paper is cited within a text, the more its scientific contribution is significative.

3 Our Approach

In this section, we introduce CiTalO, a tool that infers the function of citations by combining techniques of ontology learning from natural language, sentiment-analysis, word-sense disambiguation, and ontology mapping. These techniques

² ACL Anthology Network: <http://clair.eecs.umich.edu/aan/index.php>

³ The *context window* [14] of a citation is a chain of sentences implicitly referring to the citation itself, which usually starts from the citation sentence and involves few more subsequent sentences where that citation is still implicit [3].

are applied in a pipeline whose input is the textual context containing the citation and the output is a one or more properties of CiTO [12].

The overall CiTaLO schema is shown in Fig. 1. It was inspired by Gangemi *et al.*'s work [7], in which a similar pipeline was used with good results for automatically typing DBpedia resources by analysing corresponding Wikipedia abstracts. Five steps (described below) compose the architecture, and each one is implemented as a pluggable OSGi component [11] over a Pipeline Manager that coordinates the process.

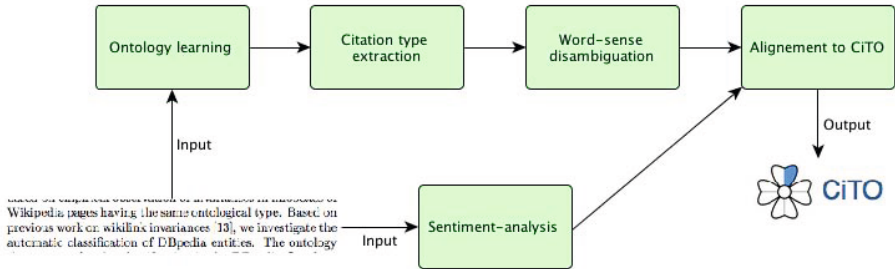


Fig. 1. Pipeline used by CiTaLO. The input is the textual context in which the citation appears and the output is a set of properties of the CiTO ontology.

In order to detail the components of CiTaLO we will discuss how the algorithm works on the following sample sentence: “It extends the research outlined in earlier work X.”, where “X” is the cited work.

Sentiment-analysis to Gather the Polarity of the Citational Function.

The aim of the sentiment-analysis in our context is to capture the sentiment polarity emerging from the text in which the citation is included. The importance of this step derives from the classification of CiTO properties according to three different polarities, i.e., positive, neuter and negative. This means that being able to recognise the polarity behind the citation would restrict the set of possible target properties of CiTO to match. We are currently using AlchemyAPI⁴, a suite of sentiment-analysis and NLP tools that exposes its services through HTTP REST interfaces. The output returned by this component with respect to our example is a positive polarity.

Ontology Extraction from the Textual Context of the Citation. The first mandatory step of CiTaLO consists of deriving a logical representation of the sentence containing the citation. The ontology extraction is performed by using FRED [13], a tool for ontology learning based on discourse representation theory, frames and ontology design patterns. Such an approach follows the one proposed by Gangemi *et al.* [7], which exploited FRED for automatically typing DBpedia entities. The transformation of the sentence into a logical form allows us

⁴ AlchemyAPI: <http://www.alchemyapi.com>

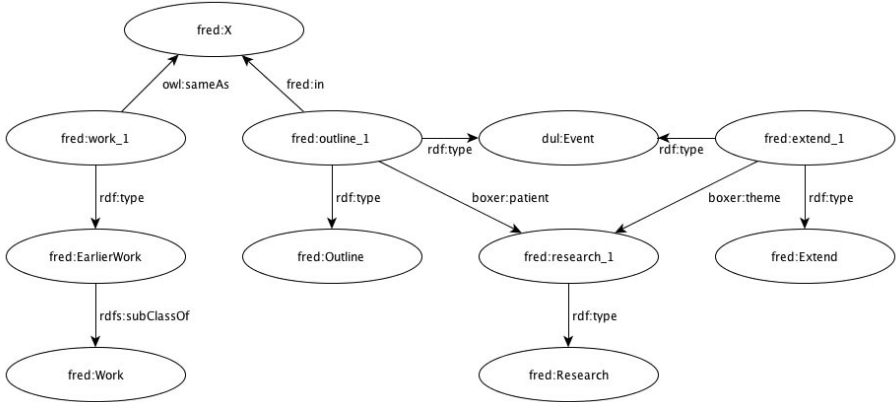


Fig. 2. FRED result for “It extends the research outlined in earlier work X”

to recognise graph-based heuristics in order to detect possible types of functions of the citation. The output of FRED on our example is shown in Fig. 2. FRED recognises two events, i.e., *Outline* and *Extend*, and the cited work *X* is typed as *EarlierWork* that is subclass of *Work*.

Citation Type Extraction through Pattern Matching. The second step consists of extracting candidate types for the citation, by looking for patterns in the FRED result. In order to collect these types we have designed ten graph-based heuristics and we have implemented them as SPARQL queries. The pattern matcher tries to apply all the patterns, which are namely:

```

SELECT ?type WHERE {?subj ?prop fred:X ; a ?type}
SELECT ?type WHERE {?subj ?prop fred:X ; a ?typeTmp .
  ?typeTmp rdfs:subClassOf+ ?type}
SELECT ?type WHERE {?subj a dul:Event , ?type .
  FILTER(?type != dul:Event)}
SELECT ?type WHERE {?subj a dul:Event , ?typeTmp .
  ?typeTmp rdfs:subClassOf+ ?type.FILTER(?type != dul:Event)}
SELECT ?type WHERE {?subj a dul:Event ;
  boxer:theme ?theme . ?theme a ?type}
SELECT ?type WHERE {?subj a dul:Event ; boxer:theme ?theme .
  ?theme a ?typeTmp . ?typeTmp rdfs:subClassOf+ ?type}
SELECT ?type WHERE {?subj a dul:Event ;
  boxer:patient ?patient . ?patient a ?type}
SELECT ?type WHERE {?subj a dul:Event ; boxer:patient ?pat .
  ?pat a ?typeTmp . ?typeTmp rdfs:subClassOf+ ?type}
SELECT ?type WHERE {?subj a dul:Event ; boxer:patient ?pat .
  ?pat ?prop ?any . ?any a ?type}
SELECT ?type WHERE {?subj a dul:Event ; boxer:patient ?pat .
  ?pat ?prop ?any . ?any a ?typeTmp .
  ?typeTmp rdfs:subClassOf+ ?type}

```

Applying these patterns to the example the following candidate types are found: *Outline*, *Extend*, *EarlierWork*, *Work*, and *Research*. The current set of patterns is quite simple and incomplete. We are investigating new patterns and we are continuously updating the catalogue.

Word-sense Disambiguation. In order to gather the sense of candidate types we need a word-sense disambiguator. For this purpose we used IMS [20], a tool based on linear support vector machines. The disambiguation is performed with respect to OntoWordNet [6] (the OWL version of WordNet) and produces a list of synsets for each candidate type. The following disambiguations are returned on our example: (i) *Extend* is disambiguated as `own:synset-prolong-verb-1`, (ii) *Outline* as `own:synset-delineate-verb-3`, (iii) *Research* as `own:synset-research-noun-1`, (iv) *EarlierWork* and *Work* as `own:synset-work-noun-1`.

The output of this step can also be extended by adding *proximal synsets*, i.e. synsets that are not directly returned by IMS but whose meaning is close to those found while disambiguating. To do so, we use the RDF graph of proximity introduced in [7].

Alignment to CiTO. The final step consists of assigning CiTO types to citations. We use two ontologies for this purpose: *CiTOFunctions* and *CiTO2Wordnet*. The CiTOFunctions ontology⁵ classifies each CiTO property according to its factual and positive/neutral/negative rhetorical functions, using the classification proposed by Peroni *et al.* [12].

CiTO2Wordnet⁶ maps all the CiTO properties defining citations with the appropriate Wordnet synsets (as expressed in OntoWordNet). This ontology was built in three steps:

- *identification step.* We identified all the Wordnet synsets related to each of the thirty-eight sub-properties of *cites* according to the verbs and nouns used in property labels (i.e. *rdfs:label*) and comments (i.e. *rdfs:comment*) – for instance, the synsets *credit#1*, *accredit#3*, *credit#3*, *credit#4* refers to the property *credits*;
- *filtering step.* For each CiTO property, we filtered out all those synsets of which the *gloss*⁷ is not aligned with the natural language description of the property in consideration – for instance, the synset *credit#3* was filtered out since the gloss “accounting: enter as credit” means something radically different to the CiTO property description “the citing entity acknowledges contributions made by the cited entity”;
- *formalisation step.* finally, we linked each CiTO property to the related synsets through the property *skos:closeMatch*. An example in Turtle is: `cito:credits skos:closeMatch own:synset-credit-verb-1`.

⁵ CiTOFunctions:<http://www.essepuntato.it/2013/03/cito-functions>

⁶ CiTO2Wordnet ontology:<http://www.essepuntato.it/2013/03/cito2wordnet>

⁷ In Wordnet, the *gloss* of a synset is its natural language description.

The final alignment to CiTO is performed through a SPARQL CONSTRUCT query that uses the output of the previous steps, the polarity gathered from the sentiment-analysis phase, OntoWordNet and the two ontologies just described. In the case of empty alignments, the CiTO property *citesForInformation* is returned as base case. In the example, the property *extends* is assigned to the citation.

4 Testing and Evaluation

The test consisted of comparing the results of CiTalO with a human classification of the citations. The test bed we used for our experiments includes some scientific papers (written in English) encoded in XML DocBook, containing citations of different types. The papers were chosen among those published in the proceedings of the Balisage Conference Series. In particular, we automatically extracted citation sentences, through an XSLT document⁸, from all the papers published in the seventh volume of Balisage Proceedings, which are freely available online⁹. For our test, we took into account only those papers for which the XSLT transform retrieved at least one citation (i.e. 18 papers written by different authors). The total number of citations retrieved was 377, for a mean of 20.94 citations per paper. Notice that the XSLT transform was quite simple at that stage. It basically extracted the *citation sentence* around a citation (i.e. the sentence in which that citation is explicitly used), preparing data for the actual CiTalO pipeline.

We first filtered all the citation sentences from the selected articles, and then we annotated them manually using the CiTO properties. Since the annotation of citation functions is actually an hard problem to address – it requires an interpretation of author intentions – we mark only the citations that are accompanied by verbs (*extends*, *discusses*, etc.) and/or other grammatical structures (*uses method in*, *uses data from*, etc.) carrying explicitly a particular citation function. We considered that rule as a strict guideline as also suggested by Teufel *et al.* [18].

We marked 106 citations of out the 377 originally retrieved, obtaining at least one representative citation for each of the 18 paper used (with a mean of 5.89 citations per paper). We used 21 CiTO properties out of 38 to annotate all these citations, as shown in Table 1.

Interesting similarities can be found between such a classification and the results of Teufel *et al.* [19]. In this paper, the neutral category *Neut* was used for the majority of annotations by humans; similarly the most neutral CiTO property, *citesForInformation*, was the most prevalent function in our dataset too. The second most used property was *usedMethodIn* in both analyses.

⁸ Available at <http://www.essepuntato.it/2013/sepublica/xslt>

⁹ Proceedings of Balisage 2011:<http://balisage.net/Proceedings/vol7/cover.html>

Table 1. The way we marked the citations within the 18 Balisage papers.

# Citations	CiTO property
53	citesForInformation
15	usesMethodIn
12	usesConclusionsFrom
11	obtainsBackgroundFrom
8	discusses
4	citesAsRelated, extends, includesQuotationFrom, citesAsDataSource, obtainsSupportFrom
< 4	credits, critiques, useConclusionsFrom, citesAsAuthority, usesDataFrom, supports, updates, includesExcerptFrom, includeQuotationForm, citesAsRecommendedReading, corrects

Table 2. The number of true positives, false positives and false negatives returned by running CiTalO with the eight different configurations

Configuration	TP	FP	FN
Filtered (with or without Sentiment)	47	88	59
Filtered + Proximity	40	137	66
Filtered + Proximity + Sentiment	41	136	65
All (with or without Sentiment)	52	114	54
All + Proximity (with or without Sentiment)	45	174	64

We run CiTalO on these data (i.e. 106 citations in total) and compared results with our previous analysis¹⁰. We also tested eight different configurations of CiTalO, corresponding to all possible combinations of three options:

- activating or deactivating the sentiment-analysis module;
- applying or not the proximal synsets¹¹ to the word-disambiguation output;
- using the CiTO2Wordnet ontology as described in Section 3, or an extended version that also includes all the discarded synsets during the filtering step.

The number of *true positives* (TP), *false positives* (FP) and *false negatives* (FN) obtained comparing CiTalO outcomes with our annotations are shown in Table 2.

¹⁰ All the source materials we used for the test is available online at <http://www.essepuntato.it/2013/sepublica/test>. Note that a comparative evaluation with other approaches, such as Teufel’s, was not feasible at this stage since input data and output categories were heterogeneous and were not directly comparable.

¹¹ We used the same the RDF graph of proximal synsets introduced in [7].

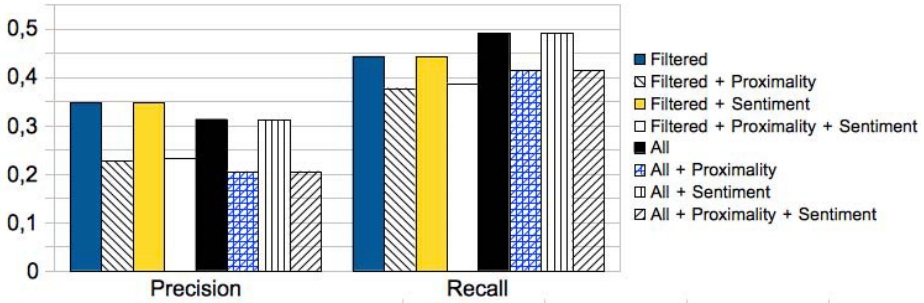


Fig. 3. Precision and recall according to the different configurations used

We calculated the precision – i.e. $TP / (TP + FP)$ – and the recall – i.e. $TP / (TP + FN)$ – obtained by using each configuration. As shown in Fig. 3, *Filtered* and *Filtered+Sentiment* had the best precision (i.e. 0.348) and the second recall (i.e. 0.443), while *All* and *All+Sentiment* had the second precision (i.e. 0.313) and the best recall (i.e. 0.491).

There is no configuration that emerges as the best one from these data. They rather suggest an hybrid approach that also takes into account some of the discarded synsets. It is evident that the worst configurations were those that took into account all the proximal synsets. It looks that the more synsets CiTaLO uses, the less the citation functions retrieved conform to humans’ annotations.

In general, the values of precision and recall of our experiments are quite low. However, our preliminary tests aimed at defining a baseline for future developments of our approach, more than a definitive evaluation of CiTaLO effectiveness.

5 Limitations and Future Research Directions

In this section we discuss some limitations and possible improvements of CiTaLO outlined by the tests and that we plan to address in future releases of the tool.

Coverage of CiTO Properties. The manual annotation process highlighted that CiTO properties do not cover all the citation scenarios addressed in the experiment. For instance, let us consider the following sentence from [16]: “*We speculate that some Goddag-based structure analogous to the multi-coloured trees of [Jagadish et al. 2004] may be a useful solution*”.

The verb *speculate* used above is very specific and refers to synsets that are not included in the mapping defined in the CiTO2Wordnet ontology. This kind of citation is not explicitly mentioned in CiTO neither. The same happens for citations – usually they are introduced by modal verbs – that suggest a work as *potential solution* for an issue related to the paper in consideration, for instance the aforementioned sentence and the following one (again from [16]): “*Mechanisms like Trojan Horse markup ([DeRose 2004], [Bauman 2005]) can be used to serialize discontinuous elements*”.

What is needed here is an accurate analysis of citations in papers so as to suggest some extensions to CiTO itself. Towards this direction, a good

starting point is to use Jorg’s previous work on cue verbs [9], where she listed one hundred-fifty verbs that are typically used in citations within scientific articles.

Noise of Proximity Synsets. The diagram in Fig. 3 clearly shows that using proximity synsets decreased both precision and recall. One would expect, on the other hand, that a larger set of synsets produced better results.

This depends on the number of *citesAsInformation* retrieved by CiTalO (re-mind that *citesAsInformation* is assigned when no further CiTO property is identified). Let us consider the case of *Filtered: citesForInformation* was assigned correctly 42 times out of 47 occurrences¹², while using *Filtered+Proximity* the same property was detected only 31 times and other more specific CiTO properties were assigned instead. The problem is that those assignments are not correct, as they derive from proximal synsets that are actually too far from the ones being processed in CiTO2Wordnet. These synsets should not be considered or, at least, should be given less importance than others that are closer to the ones in CiTO2Wordnet. For future releases of CiTalO, in fact, we plan to use proximal synsets distance in order to reduce such a noise.

Matching Synsets and Compound-word Properties. The current CiTalO alignment between synsets and CiTO properties does not work properly with properties described by compound words, such as *useMethodIn*. In fact, CiTalO returns a match whether one of the synsets of the compound words matches with a CiTO property. For instance, let us consider the following sentence (from [16]): “*Later versions of the TEI Guidelines [ACH/ACL/ALLC 1994] define more powerful methods of encoding discontinuity*”.

CiTalO returns the property *usesMethodIn* since one of the related synsets of that property, i.e. *synset:method-noun-1*, was actually found. This output is not correct, since that property should be returned only if there exists evidence that the current work *uses* (a term that is actually missed from that sentence) a particular *method* from another article, while here it seems not to be the case. Future version of CiTalO must take into account these scenarios too.

Identification of the Context Window of Citations. In our experiments, we always used the citation sentence as input of CiTalO. However, as previously noticed by Athar *et al.* [2], the actual intended sentiment and motivation of a citation is not always present in the citation sentence. It may be explicit in some other sentences close to the citation sentence and can refer implicitly to the cited work (through authors’ names, project’s name, pronouns, etc.). The identification of the right citational *context window* [14] is a complex issue that should be addressed to improve the effectiveness of CiTalO.

Identification of Implicit Citations. The identification of *implicit citations* [3] is another issue related to the one being discussed. Let us consider some sentences of a paragraph from [16]: “*XCONCUR and similar mechanisms [Hilbert/*

¹² The other citation functions retrieved are: *citesAsRecommendedReading*, *uses-DataFrom*, *citesAsDataSource*, *extends* and *usesMethodIn* – all of them used just one time within the true positive set.

Schonefeld/Witt 2005] already incorporate the containment/dominance distinction to a certain degree. [...] And like non-concurrent XML, XCONCUR has no conception of discontinuous elements”.

While in the first sentence, it seems that the authors want to praise with a positive connotation the work done by others (i.e. XCONCUR), in the latter sentence they criticise them. The “XCONCUR” in the latter sentence actually represents an implicit citation of the reference contained in the former sentence and, in this case, delimits also the context window of the citation itself. Detecting such scenarios is a further refinement that can improve CiTalO results.

Using Rhetoric Structures. According to Teufel *et al.* [18], recognising implicit citations and context windows “is often not informative enough for the searcher to infer the relation” of citations. Further information can be given by also identifying the rhetorical function of the entire paragraph or section in which the citation appears. For instance, all the references in the “related works” section are usually used to indicate related articles (i.e. *citesAsRelated*) to the topic under consideration, while citations in the introduction present background information (i.e. *obtainsBackgroundFrom*) of the field in which the work described in the article is placed. We are thinking to apply existing techniques of automatic recognition of document structures, e.g. that proposed by Di Iorio *et al.* [5], to retrieve the rhetoric function of sections in scientific articles and integrate such analysis with CiTalO.

6 Conclusions

The implementation of CiTalO is still at an early stage; current experiments are admittedly not enough to fully validate this approach. However, the overall approach is very open to incremental refinements. The goal of this work, in fact, was to build such a modular architecture, to perform some exploratory experiments and to identify issues and possible developments of our approach. We are currently working to include a mechanism for the automatic identification of *context windows* of citations given an input article and to improve *patterns’ matching* phases in CiTalO. In addition, we plan to perform exhaustive tests with a larger set of documents and users.

References

1. Athar, A.: Sentiment Analysis of Citations using Sentence Structure-Based Features. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 81–87 (2011)
2. Athar, A., Teufel, S.: Context-Enhanced Citation Sentiment Detection. In: Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, pp. 597–601 (2012)
3. Athar, A., Teufel, S.: Detection of implicit citations for sentiment detection. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 18–26 (2012)

4. Copestake, A., Corbett, P., Murray-Rust, P., Rupp, C.J., Siddharthan, A., Teufel, S., Waldron, B.: An architecture for language processing for scientific text. In: Proceedings of the UK e-Science All Hands Meeting (2006)
5. Di Iorio, A., Peroni, S., Poggi, F., Vitali, F.: A first approach to the automatic recognition of structural patterns in XML documents. In: Proceedings of the 2012 ACM Symposium on Document Engineering, pp. 85–94 (2012), doi:10.1145/2361354.2361374
6. Gangemi, A., Navigli, R., Velardi, P.: The OntoWordNet Project: Extension and Atomization of Conceptual Relations in WordNet. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS/DOA/ODBASE 2003. LNCS, vol. 2888, pp. 820–838. Springer, Heidelberg (2003)
7. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic Typing of DBpedia Entities. In: Proceedings of the 11th International Semantic Web Conference, pp. 65–81 (2012), doi:10.1007/978-3-642-35176-1_5
8. Hou, W., Li, M., Niu, D.: Counting citations in texts rather than reference lists to improve the accuracy of assessing scientific contribution. *BioEssays* 33(10), 724–727 (2011), doi:10.1002/bies.201100067
9. Jorg, B.: Towards the Nature of Citations. In: Poster Proceedings of the 5th International Conference on Formal Ontology in Information Systems (2008)
10. Moravcsik, M.J., Murugesan, P.: Some Results on the Function and Quality of Citations. *Social Studies of Science* 5(1), 86–92 (1975)
11. OSGi Alliance, OSGi service platform, release 3. IOS Press, Inc. (2003)
12. Peroni, S., Shotton, D.: FaBiO and CiTO: ontologies for describing bibliographic resources and citations. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 17, 33–43 (2012), doi:10.1016/j.websem.2012.08.001
13. Presutti, V., Draicchio, F., Gangemi, A.: Knowledge extraction based on discourse representation theory and linguistic frames. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d’Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 114–129. Springer, Heidelberg (2012)
14. Qazvinian, V., Radev, D.R.: Identifying non-explicit citing sentences for citation-based summarization. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 555–564 (2010)
15. Shotton, D.: Semantic publishing: the coming revolution in scientific journal publishing. *Learned Publishing* 22(2), 85–94 (2009), doi:10.1087/2009202
16. Sperberg-McQueen, C.M., Huitfeldt, C.: Markup Discontinued: Discontinuity in TexMeCs, Goddag structures, and rabbit/duck grammars. In: Proceedings of Balisage: The Markup Conference 2008 (2008), doi:10.4242/BalisageVoll1.Sperberg-McQueen01
17. Teufel, S., Carletta, J., Moens, M.: An annotation scheme for discourse-level argumentation in research articles. In: Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics, pp. 110–117 (1999)
18. Teufel, S., Siddharthan, A., Tidhar, D.: Automatic classification of citation function. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 103–110 (2006)
19. Teufel, S., Siddharthan, A., Tidhar, D.: An annotation scheme for citation function. In: Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue, pp. 80–87 (2009)
20. Zhong, Z., Ng, H.T.: It Makes Sense: A wide-coverage word sense disambiguation system for free text. In: Proceedings of the ACL 2010 System Demonstrations, pp. 78–83 (2010)

YASGUI: Not Just Another SPARQL Client*

Laurens Rietveld¹ and Rinke Hoekstra^{1,2}

¹ Department of Computer Science, VU University Amsterdam, The Netherlands
{laurens.rietveld,rinke.hoekstra}@vu.nl

² Leibniz Center for Law, University of Amsterdam, The Netherlands
hoekstra@uva.nl

Abstract. This paper introduces YASGUI, a user-friendly SPARQL client. We compare YASGUI with other SPARQL clients, and show the added value and ease of integrating Web APIs, services, and new technologies such as HTML5. Finally, we discuss some of the challenges we encountered in using these technologies for a building robust and feature rich web application.

Keywords: SPARQL, endpoints, API, Web service, Semantic Web.

1 Introduction

Developers that use traditional Web technologies are pampered with full-featured development tools such as in-browser debugging, integrated development environments, increasingly simple and lightweight services (RESTful APIs), and broad take up in industry. Semantic Web technologies have some catching up to do. The recent start of the W3C Linked Data Platform working group¹ is a good step in bringing triple-store querying closer to traditional RESTful APIs. However, the ingenious developer wanting to have a first taste of Linked Data is scared away by austere clients for a rich but complex query language: SPARQL.

Indeed, several good RDF programming libraries exist, but uptake of these still relies on a good understanding of SPARQL and the underlying Semantic Web paradigm which can only be attained with simple, lightweight and user friendly clients for interacting with Linked Data. This observation holds for Semantic Web savvy developers as well: trying and testing SPARQL queries is often a cumbersome and painful experience: all who know the RDF namespace by heart raise their hands now! A related question that is hard to answer for many: “Where is that Linked Data?”. Most of us will know the DBpedia endpoint URL, but can perhaps mention only a handful of other endpoints in total.

This paper introduces yet another SPARQL GUI (YASGUI²), a SPARQL client that shows the added value of combining Web 2.0 and Semantic Web technologies [1,2] for providing a more gentle Linked Data interaction environment. We find that most existing SPARQL clients do not offer functionality that

* This work was supported by the Dutch national program COMMIT.

¹ See http://www.w3.org/2012/ldp/wiki/Main_Page

² See <http://aers.data2semantics.org/yasgui/>

goes far beyond a simple HTML form. These implementations convey a rather *narrow* interpretation of what a SPARQL client interface should do: POST (or GET) a SPARQL query string to an endpoint URL. As a result, they currently offer only a selection of the features that we, as a community, could offer to both ourselves as well as new users of Semantic Web technology.

YASGUI is a web-based SPARQL client that functions as a wrapper for both remote and local endpoints. It integrates linked data services and web APIs to offer features such as autocompletion and endpoint lookup. It supports query retention – query texts persist across sessions – and query permalinks, as well as syntax checking and highlighting. YASGUI is easy to deploy locally, and it is robust. Because of its dependency on third party services, we have paid extra attention to graceful degradation when these services are inaccessible or produce unintelligible results

The following sections give a brief overview of the featureset of the current state of the art features of SPARQL clients, followed by a more detailed description of YASGUI.

2 SPARQL Client Features

Table 1 lists eleven currently existing SPARQL clients, ranging from very basic to elaborate. The features of these clients fall into several categories, *syntactic* features (autocompletion, syntax highlighting and checking), *applicability* features (endpoint or platform dependent or independent) and *usability* (query retention, results rendering and download, quick evaluation). This section describes these features in more detail, and discusses whether and how various SPARQL clients implement these features.

2.1 Syntactic Features

Most modern applications containing textual input support *autocompletion*. Examples are the Google website which shows an autocompletion list for your search query, or your browser which (based on forms you previously filled in) shows autocomplete lists for text inputs. One advantage of autocompletion is that it saves you from writing the complete text. Another advantage is the increase in transparency, as the autocompletion suggestions may contain information the user was not aware of. The latter is especially interesting for SPARQL, where users might not always know the exact prefix he/she would like to use, or where the user might not know all available properties in a triplestore. The only SPARQL interface that currently makes use of this functionality is the FLINT SPARQL Editor³, which uses autocompletion to suggest classes and properties.

Syntax highlighting is a common functionality for programming language editors. It allows user to distinguish between different properties, variables, strings, etc. The same advantage holds for query languages such as SPARQL, where

³ See <http://openuplabs.tso.co.uk/demos/sparqleditor>

you would like to distinguish between literals, URIs, query variables, function calls, etc. The only SPARQL editor currently supporting syntax highlighting is the FLINT SPARQL Editor, which uses the CodeMirror JavaScript library⁴ to bring color to SPARQL queries.

Most Integrated Development Environments (IDEs) provide feedback when code contains syntax errors (i.e. *syntax checking*). Feedback is immediate, which means the user can spot syntax errors in the code without having to execute it. Again, such functionality is useful for SPARQL editing as well. Immediate feedback on a SPARQL syntax means the user can spot invalid queries without having to execute it on a SPARQL endpoint. The FLINT SPARQL editor supports syntax checking by means a JavaScript SPARQL grammar and parser.

2.2 Applicability Features

There are only few clients who allows access to multiple endpoints. Most triplestores provide a client interface, linking to that specific endpoint. They are *endpoint dependent*. Examples are 4Store [6], OpenLink Virtuoso [7], OpenRDF Sesame Workbench [4] and SPARQLer⁵. More generic clients are the Sesame2 Windows Client [4], Glint⁶, Twinkle⁷ and SparqlGUI⁸. Other applications fall somewhere in between. The FLINT SPARQL Editor only connects to endpoints which support cross-domain JavaScript (i.e. CORS enabled). This is a problem because not all endpoints are CORS enabled, such as FactForge, CKAN, Mondeca or data.gov. Other editors support only XML or JSON as query results, such as SNORQL⁹ (part of D2RQ [3]), which only support query results in SPARQL-JSON format.

Platform (In)dependence increases the accessibility of a SPARQL client. The user can access the client on any operating system. Web interfaces are a good example: a site should work on any major browser (Internet Explorer/Firefox/Chrome), and at least one of these browsers is available for any type of common operating system. Examples are Virtuoso, 4Store and the Flint SPARQL Editor. Another example of multi-platform support is the use of a .jar file (e.g. Twinkle), as any major operating system supports Java and executing Java archives. Examples of single-platform applications are Sesame2 Windows Client and SparqlGUI: they require Windows.

2.3 Usability Features

Query retention allows for easy re-use of important or often used queries. This allows the user to close the application, and resume working on the query later. An example is the ‘Query Book’ functionality of the Sesame Windows Client.

⁴ See <http://codemirror.net/>

⁵ See <http://www.sparql.org/>

⁶ See <https://github.com/MikeJ1971/Glint>

⁷ See <http://www.ldodds.com/projects/twinkle/>

⁸ See <http://www.dotnetrdf.org/content.asp?pageID=SparqlGUI>

⁹ See <https://github.com/kurtjx/SNORQL/>

Quick evaluation or testing of a graph generated by the user should not require the hassle of installing a local triplestore. Ideally, this functionality would be embedded in the SPARQL client application itself. Most applications requiring a local installation on the users computer support this feature, such as Twinkle. The Sesame Windows Client supports file uploads as well, though it requires a local triplestore which implements the OpenRDF SAIL API.

Query results (such as JSON or XML) for SELECT queries are often relatively difficult to read and interpret, especially for a novice. A *rendering* method which is easy to interpret and understand is a table. All applications except 4Store support the rendering of query results into a table. Because of the use of persistent URIs, we would expect navigatable results for resources, e.g. in the form of drawing the URIs as hyperlinks. This feature is not supported by some applications, such as Virtuoso, Twinkle or SparqlGUI. SNORQL is an application with an elaborate way of visualizing the query results. Besides allowing the user to navigate to the page of the URI, the user can click on a link to browse the current endpoint for resources relevant to that URI.

Downloading the results as a file allows for better re-use of these results. A user might want to avoid running the same heavy query more than once, and use the results stored as a file instead. Additionally, the results of CONSTRUCT queries are often used in other applications or triplestores. Saving the user from needing to copy & paste query results clearly improves user experience as well. The only application that does not support the downloading of results, is the FLINT SPARQL editor.

Most of the clients described above are restricted to one simple task: accessing information behind a SPARQL endpoint. However, equally important to this task is assisting the user in doing so. This is something where all but one applications fail. Regrettably, the one interface with a user-friendly interface (FLINT SPARQL editor) falls short in the important feature of accessing all endpoints. We conclude that currently no single endpoint independent, accessible, user-friendly SPARQL client exists.

3 The YASGUI SPARQL Client

In this rest of this section we discuss the architecture, features and design considerations of YASGUI (Figure 1) and compare them to other clients.

3.1 Architecture

YASGUI is built using SmartGWT toolkit¹⁰, jQuery¹¹, and uses new HTML5 functionalities such as local storage and client-side generation of files. Some of the newest HTML5 functionalities are not supported by outdated browsers and Internet Explorer. This degradation is handled gracefully: access via an incompatible browser results in a notification to the user and disabled features (such

¹⁰ See <http://www.smartclient.com/product/smartgwt.jsp>

¹¹ See <http://jquery.com/>

Table 1. SPARQL client feature matrix

Feature	4Store	OpenLink Virtuoso	SNORQL	SPARQLer	Sesame Workbench	Sesame2 Windows Client	Glint	Twinkle	SparqlGUI	Flint SPARQL Editor	YASGUI
Autocompletion	-	-	-	-	-	-	-	-	-	+ ^a	+ ^b
Syntax Highlighting	-	-	-	-	-	-	-	-	-	+	+
Syntax Checking	-	-	-	-	-	-	-	-	-	+	+
Multiple Endpoints	-	-	-	-	-	+	+	+	+ ^c	+/ ^c	+
Query retention	-	-	-	-	-	+	+	-	+	-	+
File upload	-	-	-	-	+	+/ ^d	-	+	+	-	- ^e
Platform independent	+	+	+	+	+	-	-	+	-	+	+
Results rendering	-	+/ ^f	+	+/ ^f	+	+/ ^f	+/ ^f	+/ ^f	+/ ^f	+	+
Results download	+	+	+	+	+	+	+	+	+	-	+

^a Autocompletion of properties and classes available in the triple store

^b Autocompletion of prefixes/namespaces, support for properties and classes is a planned feature.

^c Can deal with a limited number of endpoints, e.g. only CORS enabled ones.

^d File upload requires a local triple store that implements the OpenRDF SAIL API, e.g. OpenRDF Sesame or OpenLink Virtuoso.

^e File upload is a planned feature, using the rdfstore-js client side triple store.

^f The rendering does not use hyperlinks for URI resources.

as downloading of files, or client-side caching of large objects). The decision to use HTML5 is motivated by the increasing support of the standard by major browsers. The server-side part of YASGUI is responsible for some of the communication with external services and endpoints. Communication with SPARQL endpoints is done using the Jena library [5]. External services used by YASGUI are CKAN¹², Mondeca¹³ and Prefix.cc¹⁴ (see section 3.2), and bitly¹⁵ (see section 3.4).

3.2 Syntactic Features

Two existing libraries provide support for syntax *highlighting* and *checking* in YASGUI: The CodeMirror JavaScript library, which is an extensive JavaScript library for highlighting code, and a JavaScript SPARQL grammar of the FLINT SPARQL Editor. Given this grammar, CodeMirror applies the highlighting to the SPARQL query. Additionally, CodeMirror provides a well documented API

¹² See <http://semantic.ckan.net/sparql>

¹³ See <http://labs.mondeca.com/endpoint/ends>

¹⁴ See <http://prefix.cc/>

¹⁵ See <http://bitly.com>

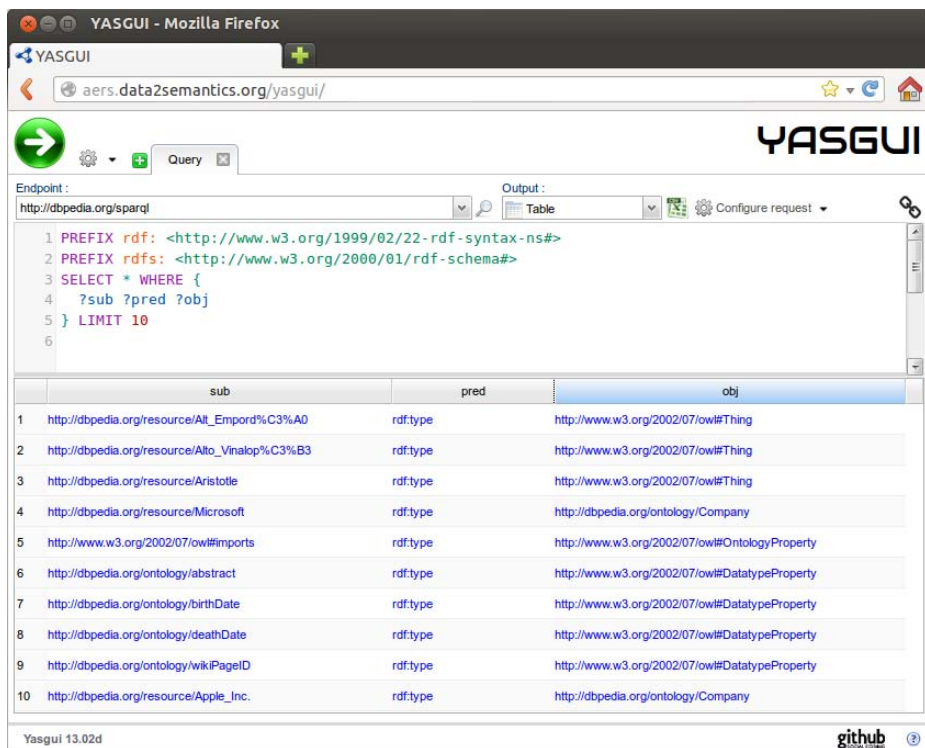


Fig. 1. Screenshot of the YASGUI interface

to parse and dissect the SPARQL query, useful for other YASGUI features such as prefix autocompletion. Both libraries are well documented, well maintained, extendible and easy to use. The existence of both libraries illustrate the availability of elaborate open source project, and the small amount of effort it takes to integrated them into an application.

YASGUI uses the RESTful API of Prefix.cc to perform *autocompletion* of namespace prefixes: full namespace URIs are completed as you type. We furthermore rely on the CKAN SPARQL endpoint for endpoint URL autocompletion and search. This autocompletion feature functions by matching the partially typed endpoint, with the list of endpoints (and their descriptions). The CKAN endpoint provides access to the CKAN datahub.io¹⁶ catalogue of datasets. Users can either use a simple autocompletion combobox, or browse through a list of endpoints in a table. Our fallback option for CKAN is the endpoint provided by Mondeca. Mondeca hosts a project where the availability (up-time) of these endpoints is published. Both endpoints have proven to be difficult to use and access for our purposes. The CKAN endpoint is rather unreliable in up-time, and the Mondeca endpoint often return syntactically invalid XML. In the

¹⁶ See <http://datahub.io>

implementation of YASGUI we try to handle both issues as gracefully as possible. The list of endpoints provided by CKAN or Mondeca is cached on the YASGUI server. If YASGUI fails to retrieve the list in real time from either of the endpoints, we fall back to the cached results.

Another issue with CKAN (and to a lesser extent Mondeca) is the reward model for adding and maintaining the catalogue: there is little incentive for owners of a dataset to add it to CKAN, and even less incentive keep the information up to date (e.g. when the endpoint is down or moved). As a result, CKAN is cluttered with outdated information, and some endpoints are missing. This is partly compensated by Mondeca, which allows filtering by endpoints which are up, though incorrect or missing information still persists. The reward model employed by Prefix.cc is the opposite: the content is crowd-sourced (anybody can add prefixes), and voting is used to deal with conflicting prefix definitions. Users of prefix.cc have an incentive to keep the information up to date and as correct as possible. As a result, the information retrieved from prefix.cc is more reliable and usable than information from CKAN and Mondeca.

3.3 Applicability Features

As mentioned in section 2, client-side web applications such as the FLINT SPARQL Editor are *endpoint independent*, but only work on CORS-enabled endpoints. To overcome this limitation, YASGUI includes a server-side proxy for accessing endpoints that do not support CORS. For endpoints which do support cross domain JavaScript, YASGUI executes the queries solely from the clients side via JavaScript. The only scenario where YASGUI fails to connect to an endpoint is where a locally installed endpoint is unreachable from the web, operating on a different port than YASGUI, and CORS-disabled. Here, the YASGUI proxy is not able to access the client. Because of the CORS restriction, YASGUI is not able to access the endpoint via JavaScript as well, as it is operating on a different port. We consider this issue to be minor: because the endpoint is installed locally, the user will have access to change its CORS settings, or even run the endpoint via a different port.

Other than dealing with the accessibility issues of CORS disabled sites, endpoint independent clients should support configurable requests. For instance, some endpoints may only support the XML results format, or allow the use of additional request parameters, such as the ‘soft-limit’ of 4Store. Such endpoints can only be used to their full potential if users are able to specify these additional arguments manually. Therefore, YASGUI supports the specification of an arbitrary number of request parameters for every endpoint.

Finally, we had to add quite some code to deal with possible errors returned by endpoints. The SPARQL protocol specifies what the endpoint request and response should look like, but leaves error handling unspecified: what HTTP error code should be sent by an endpoint, and how should error messages be communicated. As a result, triple stores come with various ways of conveying errors. Some endpoints return the error as part of an HTML page (with the regular 200 HTTP code), or as a SPARQL query result. Others only return an

HTTP error code, where only some include a reason phrase together with the error code. The latter is a best practice for RESTful services. The absence of a standard, and the failure to adhere to best practices, makes a generic robust error handling solution messy and difficult to implement. Developing such a solution requires coding and testing by trial and error, and test queries on as many different endpoints as possible.

3.4 Usability Features

As Table 1 shows, most SPARQL clients support both *rendering* and *downloading* of query results to some extent. YASGUI does both as well. Users can render results either as a lightweight HTML table (for large numbers results), an elaborate sortable/groupable table, or show the raw query results with syntax highlighting. Tables can be downloaded as CSV, where raw query results are available for download ‘as is’.

YASGUI stores the application state, making this application state persistent between browser/user sessions: a returning user will see the screen as it was when she last closed the YASGUI browser page. We elaborate on this feature by providing query permalink functionality. For a given query and endpoint combination, YASGUI creates a link. Opening the link in a browser will open YASGUI with the specified query, endpoint and request arguments filled in. We believe this is a welcome feature for people working together with a need to exchange queries. Other than regular query permalinks, YASGUI supports the generation of shortlinks as well using the Bitly URL shortener service. Using an external service such as bitly poses some limitations: there is a fair use policy, limiting the scalability of this solution. Additionally, there is a danger of link-rot as well. An alternative to an external URL shortener service is implementation of such a service as part of YASGUI. This deals with both the issues of scalability as well as link-rot. We opted for the external bitly service, as the danger of link-rot is low with a popular service such as bitly. Whenever scalability becomes an issue, the alternative of implementing our own service would still be a viable option.

4 Discussion

In the preceding sections we described YASGUI and compared it to other clients. YASGUI shows how straightforward it is combining Web APIs, libraries and new Web technologies. Compared to other clients, YASGUI is the first client that really leverages the tools and services we as a community have developed for ourselves.

In the process, we encountered 4 challenges in using WEB APIs and Linked Data together. First, maintenance of Web 2.0 content is a challenge. External services such as CKAN contain outdated, incorrect or incomplete information (section 3.2). The main challenge here is how to manage this information, e.g. using a reward model where users have an incentive to update the information, or

active curation by the service manager. Secondly, standard adherence (or lack of standards) is a challenge (section 3.3), such as the different error handling approaches implemented by endpoints. Additionally, graceful degradation (section 3.2) is an issue. The before-mentioned challenges, as well as issues such as external services breaking down, should not break the application. In the worst case, it should only break application features. Finally, it is worth considering whether or not to use online web services. Use of online web services such as url-shorteners or endpoint catalogues create external dependencies. For some services such as CKAN, graceful degradation is possible. However, for other services such as url-shorteners, this is not the case. This is a trade-off. Instead of using online web services, an application specific implementation may be preferable and more robust.

In general, our experience is that there is an abundance of libraries, new Web technologies, services, and APIs. The use of these tools increases the feature set of your application, and decreases the number of lines you have to write.

References

1. Ankolekar, A., Krötzsch, M., Tran, T., Vrandečić, D.: The two cultures: Mashing up Web 2.0 and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web* 6(1), 70–75 (2008)
2. Battle, R., Benson, E.: Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). *Web Semantics: Science, Services and Agents on the World Wide Web* 6(1), 61–69 (2008)
3. Bizer, C., Seaborne, A.: D2rq - treating non-rdf databases as virtual rdf graphs. In: *World Wide Web Internet and Web Information Systems*, p. 26 (2004)
4. Broekstra, J., Kampman, A., Van Harmelen, F.: Sesame: An architecture for storing and querying RDF data and schema information (2001)
5. Grobe, M.: Rdf, jena, sparql and the ‘semantic web’. In: *Proceedings of the 37th Annual ACM SIGUCCS Fall Conference, SIGUCCS 2009*, pp. 131–138. ACM, New York (2009)
6. Harris, S., Lamb, N., Shadbolt, N.: 4store: The design and implementation of a clustered RDF store. In: *5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2009)*, pp. 94–109 (2009)
7. Openlink Virtuoso: Universal server platform for the real-time enterprise (2009), <http://www.openlinksw.com/>

A Case-Study of Ontology-Driven Semantic Mediation of Flower-Visiting Data from Heterogeneous Data-Stores in Three South African Natural History Collections

Willem Coetzer, Deshendran Moodley, and AURONA GERBER

CAIR (Centre for Artificial Intelligence Research)

University of KwaZulu-Natal (Durban) and

CSIR Meraka (Pretoria), South Africa

w.coetzer@saiab.ac.za, moodleyd37@ukzn.ac.za, agerber@csir.co.za

<http://www.cair.za.net/>

Abstract. The domain complexity and structural- and semantic heterogeneity of biodiversity data, as well as idiosyncratic legacy data-creation processes, present significant integration and interoperability challenges. In this paper we describe a case-study of ontology-driven semantic mediation using records of flower-visiting insects from three natural history collections in South Africa. We establish a conceptual domain model for flower-visiting, expressed in an OWL ontology, and use it to semantically enrich the three data-stores. We show how this enrichment allows for the creation of an integrated flower-visiting dataset. We discuss how the ontology captures both implicit and explicit knowledge, and we show how the ontology can be used to identify and analyze high-level flower-visiting behaviour. We propose that a system that employs this ontology for semantic enrichment and semantic mediation may be used to automatically construct flower-visiting and pollination networks, the manually constructed equivalents of which are routinely used by domain scientists to analyze their data.

Keywords: biodiversity information, semantic mediation, ontology, plant-insect interactions, pollination.

1 Introduction

The challenges of integrating, or making interoperable, distributed, heterogeneous sources of biodiversity- and ecological data have been described [1,2]. Biodiversity is a complex domain and is no different from other domains in that users encode different definitions of the same concepts [3], which frustrates efforts to integrate data.

We present a case-study of three data-stores of flower-visiting insect specimens. All three data-stores consistently contained the names of the plant species, termed *host-plants*, with which both flower-visiting and non-flower-visiting insect specimens were associated. Whereas flower-visiting records were

not explicit in most records of two data-stores, most records of the third data-store contained explicit, easily distinguishable flower-visiting data. To develop a semantic mediation solution, we created the first version of an OWL ontology containing concepts related to flower-visiting and the utilization of flower products, as well as the bearing of pollen by insect vectors. Our work will facilitate the construction of a system to bring about interoperability between distributed and heterogeneous biodiversity data-stores and systems. This will enable biodiversity scientists to more easily extract and analyze the behaviour of flower-visiting insects. Such a system would allow flower-visiting and pollination networks to be automatically assembled and compared.

Outline. In Section 2 we sketch the background against which the need for our study emerged, discuss previous work in biodiversity semantics, and introduce our case-study of interoperability of flower-visiting data. Section 3 begins by describing the domain of flower-visiting and pollination, including our scope, before explaining the process of ontology construction. Expert- and implicit knowledge is highlighted. The usefulness of the concepts in the ontology is discussed in Section 4, by linking data from the data-stores to classes in the ontology. Finally we discuss our approach to a potential solution, including areas where future work is required, and conclude.

2 Background

2.1 Semantics in Biodiversity Informatics

The importance of verifiable specimen-vouchers (i.e. physical preparations such as pinned insects) in museum collections has caused attention to be focused on such specimen information [4]. In recent years *observations* of biodiversity have become important, including observations made by citizen scientists [5]. Both voucher records and observations (collectively termed occurrences) have been subject to the development and adoption of useful standards for publishing and exchanging biodiversity information (the group known as Biodiversity Information Standards (BIS), formerly called the Taxonomic Databases Working Group or TDWG) [6]. One of the BIS standards is the set of terms named the Darwin Core, which contain ‘clearly defined semantics that can be understood by people or interpreted by machines, making it possible to determine appropriate uses of the data encoded therein’ [7]. The purpose of the Darwin Core terms is to allow biodiversity data to be published and integrated [7].

Biodiversity data are commonly formatted according to the Darwin Core standard and then uploaded to a Global Biodiversity Information Facility (GBIF) participant node (such as the South African Biodiversity Information Facility, SABIF). The data then become discoverable via the GBIF Data Portal, and may be downloaded upon acceptance of conditions. Whereas such database federation has been successful for the sharing of core data attributes (e.g. the Darwin Core categorizes terms as relating to Occurrence, Event, Location, Identification, Taxon), more specialized data, for example data that record biotic

interactions such as parasitism or pollination, are typically omitted because standard terms to describe specific instances of ecological interactions do not yet exist. Currently, shared data therefore fall short of the common phrase ‘who did what to whom, where, when, how and why?’ because the ‘what’, ‘how’ and ‘why’ are still missing.

The ‘Who’ and ‘To Whom’. The Taxon Concept Schema (TCS) [8,9], is a standard model to exchange taxonomic information (hence the alternative name ‘Taxonomic Concept Transfer Schema’). The TCS is written in XML. More specifically, the TCS allows ‘explicit communication of information about Taxon Concepts and their associated names’ [8]. A Taxon Concept is a concept or definition of a group, such as a new beetle species, in a taxonomist’s mind, which may become published in an article. Several collaborative initiatives aim to define standardized concepts to describe the anatomy and morphology of animals e.g. Hymenoptera [10] or plants [11].

The ‘Where’ and ‘When’. The Darwin-SW Ontology is described as ‘an ontology using Darwin Core terms to make it possible to describe biodiversity resources in the Semantic Web’ [12]. This is seen as particularly useful for publishing, as Linked Open Data, datasets consisting of Darwin Core terms.

Ecological Semantics. Much work has been done to define concepts used in ecology. Ecological Metadata Language (EML) has a long history of practical application [13,14], and much work has advanced the use of ontologies [15,16] to create interoperable systems and to enable the execution of scientific workflows [17,18].

The Need for Defining the ‘What’, ‘How’ and ‘Why’ of Biodiversity Information. While the Ecology Ontology and Ecological Networks Ontology [15] contain useful constructs, we found no published, formal definitions of biotic interactions, i.e. concepts that describe specific behaviours representing interactions between individual animals, or between plants and animals. Some preliminary work has been done to extend the Darwin Core standard to broadly include interactions [19] by using terms e.g. `VisitedFlowerOf`, `FlowerVisitedBy`, `NestedIn`, `UsedAsNestBy`. A short list of standard terms was proposed [20] specifically for the interaction, `VisitedFlowerOf`. This list contains the elements: `PollinationEvidence`, `PollenRemoval`, `NectarRemoval`, `OilRemoval` and `FlowerPredation`. Doubt has been expressed as to whether this approach will result in the adequate expression of relationships between specimens or observations.

Semantic Mediation in Biodiversity Informatics. An underlying ontology was used to integrate cereals data from public web databases with data from a local database, allowing molecular characteristics and phenotypic expression to be correlated [37]. While the subject of semantic mediation in biodiversity informatics has been addressed as an architecture component (e.g. [17,18]), few examples of practical applications exist.

2.2 Background to the Case-Study

The Quality of Biodiversity Data in South African Museums. South African natural history museums participated in a programme [21] to cleanse and migrate their data to a standard relational database schema and application (Specify Collections Management Software, University of Kansas Biodiversity Institute). Despite having general data of a higher quality, and consistency in schema and syntax, participating researchers of flower-visiting were still unable to easily extract meaningful summaries across data-stores because semantic heterogeneity remained an unresolved challenge. Further work was therefore undertaken with three data-stores that contained data related to collections of flower-visiting insects, namely those of the Albany Museum (AM) in Grahamstown, Iziko Museum (SAM) in Cape Town and the Plant Protection Research Institute (SANC) in Pretoria. Table 1 summarizes the data attributes that characterized the data-stores and shows how the word *flower(s)* could be used to distinguish flower-visiting records. The heterogeneity of biodiversity information is evident in Table 1. For example, AM is a specialized flower-visiting data-store because it includes even the colours of visited flowers, and almost all the records are marked with the words ‘visit’ and ‘flower’ (also Table 2). On the other hand, SANC contains less-meaningful information for a flower-visiting researcher.

Table 1. Data attributes from the three data-stores. FV = percentage explicit flower-visiting records. Flower-visiting records were distinguished by the *Sampling Method* and *Insect Behaviour* attributes.

	SAM sample data (n=2 094) 3% FV	SANC sample data (n=219) 4% FV	AM sample data (n=21 159) 97% FV
Host Type	host-plant	host-plant	host-plant
Host Taxon	Diascia capensis	Ruschia indecora	Indigofera nigromontana
Sampling Method	flowers	swept from flowering Acacia albida	hand net
Insect Behaviour	foraging on nectar	[no data]	visiting flowers
Flower Colour	[no data]	[no data]	deep pink

3 Ontology Construction in the Domain of Flower-Visiting and Pollination

Various kinds of animals, including arthropods (e.g. insects), birds (e.g. humming-birds and sunbirds) and mammals (e.g. bats) are well-known *flower-visitors* because they live a life of actively, frequently and consistently seeking

out flowers in order to utilize the flowers themselves or their products. The most important flower products are nectar, pollen and oil, which are ingested or collected by the flower-visitors. Insects are important flower-visitors and many insect groups have co-evolved as pollinators of plants.

Pollination is defined with varying granularity. A simple definition reads: ‘The transfer of pollen from an anther to a stigma’ [22]. Some definitions emphasize that all pollination is ultimately an event (one-step process) because it consists of the act by which pollen is deposited on the pollen-receptive surfaces of a flower (or other reproductive structure such as a cone). In the typical case, pollination (cross-pollination) is a two-step process whereby a vector (‘carrier’) transfers pollen from the anther of one flower to the stigma of another flower [22]. This is the definition that formed the basis of our domain model, though we did not model the process or event of pollination.

In the study of flower-visiting ecology, pollination may or may not be confirmed in a field setting. Confirmation of pollination requires closely following the flower-visitor and recording its behaviour to see whether it actually transfers pollen onto the stigma. Thus, when ecologists refer to ‘pollination’ or a ‘pollinator’, unless otherwise stated, the word is usually used loosely to mean ‘inferred pollination’ or ‘potential pollinator’/‘pollen vector’ (an organism that carries or transports pollen). Flower-visiting records are the basic currency of pollination ecologists because flower-visiting is easier to observe with high confidence.

Scope. We limited our modelling to angiosperms (flowering plants) that are pollinated by vectors i.e. not by an abiotic medium such as wind or water. We circumscribed as flower-visitors those taxa that belong to the phylum Arthropoda i.e. including the terrestrial groups represented broadly by spiders, millipedes (which mostly inhabit the soil) and insects. Plant galls caused by developing insect larvae, including larvae developing in flower-galls, were excluded from the domain. There was no geographic limitation to our study.

3.1 Concepts Used in Domain Modelling: Flower-Visiting and Pollen-Bearing

For the purpose of ontology construction we chose to define the concept of a *flower-visitor* broadly, by interpreting a review of flower-visiting insects [23]. This review clearly included in the concept insects that hid in flowers (e.g. thrips), camouflaged themselves against flowers in order to ambush prey (e.g. mantids) or laid eggs in flowers (e.g. fruit flies). An insect can be a flower-visitor even if it does not ingest or collect nectar, pollen, oil (with or without terpene fragrance), resin, gum, anthers, ovules, seeds, petals or some other part of the flower or the entire flower.

It is generally accepted that pollen-transfer, both from the anther to a flower-visitor and from the flower-visitor to the stigma is an accidental process¹. A flower-visitor can become more-or-less covered in pollen, which it may then

¹ Fig-wasps seem to undertake an intentional pollination ritual [36].

groom off the surfaces of its body using its tarsi (feet) and mouthparts, and pack into the scopa (hairy patch) on the hind leg, or store on the abdomen or in the crop. The pollen is then taken back to the nest and fed to the young (e.g. social bees) or deposited as nest provision for future young (e.g. solitary bees). Some plants, e.g. orchids and milkweeds, produce a pollinium (plural pollinia), or pollen-mass, borne on a sticky stalk that adheres to the flower-visitor's body. The whole complex including the pollinium and the stalk is called a pollinarium (plural pollinaria).

3.2 Expert- and Implicit Knowledge

Students of flower-visiting and pollination know implicitly that e.g. an adult beetle or fly or wasp of a certain taxonomic group (e.g. monkey beetles of the tribe Hopliini), or any bee (superfamily Apoidea) has only one reason to be associated with a plant, and that is to visit the plant's flowers, usually to ingest or collect nectar or pollen or other flower products. Many publications list known flower-visiting groups [23].

The importance of implicit knowledge is even more pronounced in the particular case of bees of the genus *Rediviva*, consisting of 26 species that are endemic to South Africa, Lesotho and Swaziland. The females only visit a small number of plant species (about 140 species in 14 genera) whose flowers produce oil to attract these particular bees, or they will visit any number of other plant species whose flowers produce nectar instead of oil [24]. The female bees collect and carry the oil using hairs on their especially-adapted, long front legs, and take the oil back to their nests as provision (i.e. the egg is laid on the oil in the nest and the female that laid the egg then abandons the nest while the larva develops by feeding on the oil). Male *Rediviva* bees only visit flowers that produce nectar, which, like the females that visit 'nectar plants', they ingest to sustain themselves. A 'nectar-plant' could be any flowering plant species, in the area that the bee frequents, that happens to have nectar in its flowers at the time. Among all the specimen records in the SANC data-store that were created during the course of preparing two seminal articles on the famous *Rediviva* oil-collecting bees of southern Africa, the words 'visit', 'flower' or 'oil' do not occur once. The reason for this was probably related to the need for critical information to fit onto a small specimen label. No information was lost within the museum because an expert only needs to know the sex of the adult bee specimen and the plant species name to know whether a *Rediviva* bee was collecting nectar or oil, and that it was visiting flowers [25,26]).

3.3 The Flower-Visiting and Pollen-Bearer Ontology

In this section we describe the semantic analysis and ontology construction process we followed to create the OWL ontology using Protégé [27]. Both bottom-up (i.e. from the data) and top-down ontology construction approaches (i.e. from literature and discussions with experts) were employed. We re-used

concepts from the Plant Ontology [11] where possible. In modelling flower-visiting we made extensive use of the `Role` concept as defined in BFO (the Basic Formal Ontology) [28]. Examples of roles include the role of a person as a surgeon or the role of a chemical compound in an experiment. We created `-Role` concepts for the activities associated with flower-visitors, and created an Object Property `participates_in` (inverse: `participated_in_by`), thus a `FlowerVisitor` `participates_in` some `FlowerVisitorRole`. The `-Role` taxonomy is depicted in Figure 1.



Fig. 1. The roles (concepts) in the asserted class hierarchy as displayed in Protégé 4.2

3.4 The `FlowerVisitorRole`

Our objective was to make interoperable heterogeneous records of *flower-visitors*, which are generally organisms that utilize flowers. We therefore created the object property, `utilizes` (inverse: `utilized_by`), and defined the necessary condition for the class `FlowerVisitorRole`: `utilizes some WholePlant`

This means that an organism on a severed flower lying on the ground, or in a flower arrangement, cannot be a `FlowerVisitor`.

The necessary and sufficient conditions for the class, `FlowerVisitorRole`, are either:

```

A: (utilizes some FlowerMechanicalSupport)
   or (utilizes some FlowerSpace)
   or (utilizes some FlowerTissue)
   or (utilizes some FlowerProduct)
    
```

or

B: (participates_in some PlantVisitorRole)
and (member_of some FlowerVisitingGroup)

or

C: (bears some Pollen) or (bears some Pollinarium)

In Section A above, `utilizes some FlowerMechanicalSupport` could mean alighting on a flower, `utilizes some FlowerSpace` could mean inserting the proboscis into the flower or hiding in the flower, `utilizes some FlowerTissue` could mean laying an egg inside the tissue or eating the tissue, and `utilizes some FlowerProduct` could mean ingesting or collecting nectar or pollen. This class will therefore include individuals that are incidental flower-visitors (e.g. spiders) as well as highly specialized pollen-collectors (e.g. bees).

Section B in the above class definition states that a condition for an organism that `participates_in` the `FlowerVisitorRole` is that it `utilizes some WholePlant` and is a `(member_of some FlowerVisitingGroup)`.

We created the object property, `bears` (inverse: `borne_by`), meaning to ‘have on (the outside of the body)’, as in ‘the bee’s abdomen bears pollen’. This object property was used, in Section C above, to assert that a condition for an organism that `participates_in` the `FlowerVisitorRole` is that it `bears Pollen` or `bears` at least one `Pollinarium`.

3.5 The FlowerUtilizerRole and Descendent Classes, Including Implicit Knowledge of *Rediviva* Bees

It was asserted that a condition for the `FlowerUtilizerRole` is `((utilizes some FlowerMechanicalSupport) or (utilizes some FlowerSpace) or (utilizes some FlowerTissue) or (utilizes some FlowerProduct))`. This means that `FlowerUtilizerRole` is equivalent to `FlowerVisitorRole`.

We specialized the object property, `utilizes`, into the object properties, `ingests` (inverse: `ingested_by`) and `collects` (inverse: `collected_by`).

We defined a `FlowerProduct` to be the class subsuming the class `(FlowerSecretion or Pollen or Pollinarium)`. The class `FlowerSecretion` subsumed the class `(FlowerGum or FlowerNectar or FlowerOil or Flower-Resin)`.

The `FlowerUtilizerRole` was specialized into `FlowerProductUtilizerRole` and `FlowerPollenBearerRole`. More specifically, if an individual `utilizes (ingests or collects) some FlowerProduct`, that is sufficient to mean that it `participates_in` the `FlowerProductUtilizerRole`.

An individual that `(bears some Pollen) or (bears some Pollinarium)` sufficiently meets the condition for the `FlowerPollenBearerRole`. If an organism actively ingests or collects pollen, some pollen will invariably remain on its body after grooming and packing into the scopa. A necessary condition

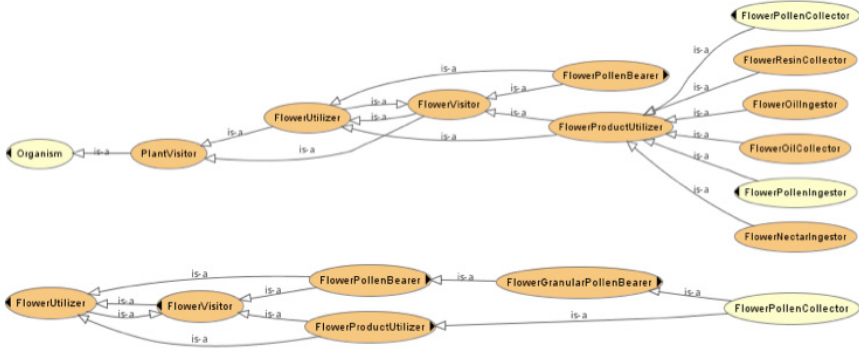


Fig. 2. It is asserted that a *FlowerPollenBearer* need not be a *FlowerProduct Utilizer*, but an organism may be both a *FlowerPollenBearer* and a *FlowerProduct Utilizer* because these classes are not disjoint. This successfully models active pollen-collecting and pollen-ingesting, which necessarily result in passively bearing pollen.

of the *FlowerPollenIngestorRole* and the *FlowerPollenCollectorRole* is therefore: *bears some Pollen*. Figure 2 depicts two parts of the inferred class hierarchy: *FlowerProductUtilizer* and sub-classes, as well as detail of the *FlowerPollenCollector* class hierarchy. The classes in Figure 2 are sub-classes of *Organism*. These classes *participate_in* the *-Role* classes depicted in the taxonomy in Figure 1.

The conditions that are sufficient for membership in the *FlowerOilCollector* class are as follows: ((*participates_in some FlowerOilCollectorRole*)) or ((*participates_in some OilPlantVisitorRole*) and (*member_of some FlowerVisitingGroup*) and (*has_sex only Female*) and (*part_of some RedivivaGenus*)).

This means that a *FlowerOilCollector* can either be observed directly (*collects some FlowerOil*) or its presence can be inferred (e.g. in the SAM data-store) from the facts that an ‘oil plant’ (with flowers that secrete oil, not nectar) was visited, the insect was a female and it was a species in the genus *Rediviva*.

3.6 The *IllegitimateFlowerVisitorRole* and Sub-classes

With reference to Figure 1, the concept of ‘illegitimately’ visiting flowers (i.e. by definitely avoiding coming into contact with the anthers, and therefore never becoming a *FlowerPollenBearer*) is frequently encountered in the flower-visiting literature, and we therefore included this in our ontology. Robbers, which damage the petals (e.g. by biting a hole in the petal to access the nectar), are distinguished from thieves, which inflict no petal damage. A secondary robber obtains nectar through the hole made by a primary robber [29].

4 Linking the Ontology to Existing and Future Data

The class, `FlowerUtilizer` (Section A of the definition of the `FlowerVisitorRole`) therefore represents records resulting from the observations of a generalist scientist who may record an organism generally utilizing a flower by e.g. sitting on, or flying around and feeding from (visiting), a flower. In the AM data-store a small number of records were classified as members of the class `FlowerUtilizer` (Table 2).

Table 2. Examples of the class `FlowerProductUtilizer` in the AM data-store

# records	Behaviour	Class
137	Visiting extrafloral nectaries	<code>PlantVisitor</code>
95	On foliage	<code>PlantVisitor</code>
8	On stem of plant	<code>PlantVisitor</code>
20135	Visiting flowers	<code>FlowerProductUtilizer</code>
380	In flowers	<code>FlowerUtilizer</code>
22	On flowers	<code>FlowerUtilizer</code>
16	Sheltering in flower	<code>FlowerUtilizer</code>
8	In copula on flowers	<code>FlowerUtilizer</code>

The vast majority of records, however, were instances of the class, `FlowerProductUtilizer`. An expert in the study of flower-visitors would record a flower-visitor to be an instance of the class `FlowerProductUtilizer` (i.e. specifically ingesting or collecting nectar or pollen). Importantly, this observation can be made by an expert observing an insect that has not even touched a flower. The expert is able to classify the organism into a specific taxonomic group, and to remember how previous individuals in this specific group have behaved (i.e. they *visited* flowers, which is a shorter way of recording that they ingested or collected nectar or pollen), and to know that newly observed individuals of the same group are unlikely to behave differently. The predominance of records of the `FlowerProductUtilizer` class therefore reflects the predominance of bees and pollen wasps in this data-store, which is due, in turn, to the development of the careers of the specialists who built the specimen collection. It is therefore not surprising that the biodiversity information in the AM data-store is richer than the information in the other data-stores.

4.1 Data in the SAM and SANC Data-Stores

Ninety-seven per cent of the records in the SAM data-store, and 96% of the records in the SANC data-store, were instances of the class `FlowerVisitor`, a term that is less meaningful than `FlowerUtilizer` or `FlowerProductUtilizer`. A small number of records in the SAM data-store were instances of sub-classes of the class `FlowerProductUtilizer`. Some of these are shown in Table 3.

Table 3. Examples of the class `FlowerProductUtilizer` in the SAM data-store

# records	Behaviour	Class
1	Collecting pollen on yellow flowers.	<code>FlowerPollenCollector</code>
1	Patrolling <i>Corymbium</i> . With pollenaria.	<code>FlowerPollinariumBearer</code>
1	Feeding on <i>Brunia laevis</i> pollen.	<code>FlowerPollenIngestor</code>
1	Foraging on nectar of <i>Euphorbia</i> flowers.	<code>FlowerNectarIngestor</code>
1	Taking resin from <i>Dalechampia capensis</i> .	<code>FlowerResinCollector</code>

Section C of the definition of the `FlowerVisitorRole` (i.e. a `FlowerPollenBearer`) is of particular, current interest. If an organism is seen to bear pollen or a pollinarium, DNA barcoding can be used to identify [30] the plant species that produced the pollen. This is a very important step in the study of flower-visiting because it means that it will no longer be necessary to observe a `FlowerPollenBearer`, either in any physical association with a plant or flower, or actually ingesting or collecting pollen, to know:

1. That it must be a `FlowerUtilizer` (but not necessarily a `FlowerProductUtilizer`) and therefore a `FlowerVisitor`;
2. The list of plant species, which it has recently visited, utilized and borne pollen from.

5 Discussion and Conclusion

We have shown how implicit domain knowledge about flower-visitors can be represented in an ontology for use in semantic enrichment of, and semantic mediation between, heterogeneous data sources.

Researchers of flower-visiting need to summarize data into lists of insect species and the plant species whose flowers those insects visit, and which they probably pollinate. These lists usually form the basis of further work involving the modelling of flower-visiting networks (which are useful in community ecology), and, more specifically, pollination networks (e.g. [31]). In an applied study the ultimate objective may be to compare the characteristics [32] of pollination networks across space or through time e.g. to estimate the effect, on pollination, of habitat transformation [33] or global change.

Clearly, systems used to capture and manage specimen data are not designed to capture the background knowledge required to access the rich, and often implicit, information associated with these records. This knowledge is usually held by the curator or scientists who generated the records. This becomes more pronounced for biodiversity researchers accessing a network of locally controlled and heterogeneous biodiversity databases. A significant barrier to data integration and analysis will therefore be removed if knowledge can be explicitly represented within the system. For example, illegitimate flower-visitor species must be excluded from the process of assembling a pollination network.

In our current ontology we assumed that there are no exceptions of a `KnownFlowerVisitingGroup`. This is an area where future work is needed

because the semantic representation of exceptions, or defeasibility with current OWL ontologies, is problematic. One of these exceptions is a particular Afrotropical bee species, which is an obligate raider of other bees' nests and therefore has no need to, and never does, visit flowers. Yet bees are the most important group of flower-visiting insects. Such exceptions will need to be carefully modelled to prevent the possibility of drawing incorrect inferences.

While the ontology described above can certainly facilitate the creation of a semantically rich flower-visiting data set, it still falls short of capturing uncertain and vague biotic interactions associated with flower-visiting occurrences. Probabilistic graphs such as Bayesian Networks are better able to deal with uncertain or vague causal relations [34]. In the earth observation domain, the combination of ontologies and Bayesian networks has recently been explored in the Sensor Web Agent Platform (SWAP) [35]. In SWAP sensor observations from heterogeneous sensor data-stores are semantically enriched with OWL ontologies and used to populate Bayesian networks to determine the probability of the occurrence of abstract physical earth observation phenomena.

The next step in our semantic mediation system will be to adapt the SWAP [35] approach and construct a Bayesian network that describes the causal relations between plant-visiting events, flower-visiting events, pollen transfer events and pollination events. These events will be defined using concepts from the flower-visiting ontology. In this way semantically enriched observations from the three data-stores can be used as proxies to determine the probabilities of the occurrence of flower-visiting and pollination events.

Acknowledgement. With gratitude we acknowledge the JRS Biodiversity Foundation (<http://www.jrsbdf.org/>) for financial support of the research presented in this paper through a 2011 grant for *Improvement and Integration of Pollinator Biodiversity Information in Africa*.

References

1. Johnson, N.F.: Biodiversity Informatics. *Annual Review of Entomology* 52, 421–438 (2007)
2. Jones, M.B., Schildhauer, M.P., Reichman, O.J., Bowers, S.: The New Bioinformatics: Integrating Ecological Data From the Gene to the Biosphere. *Annual Review of Ecology Evolution and Systematics* 37, 519–544 (2006)
3. Deans, A.R., Yoder, M.J., Balhoff, J.P.: Time to Change How We Describe Biodiversity. *Trends in Ecology & Evolution* 27, 78–84 (2011)
4. Bisby, F.A.: The Quiet Revolution: Biodiversity Informatics and the Internet. *Science* 289, 2309–2312 (2000)
5. Silvertown, J.: A New Dawn For Citizen Science. *Trends in Ecology & Evolution* 24, 467–471 (2009)
6. Biodiversity Information Standards, <http://www.tdwg.org/>
7. Wieczorek, J., Bloom, D., Guralnick, R., Blum, S., Döring, M., Giovanni, R., Robertson, T., Vieglais, D.: Darwin Core: An Evolving Community-Developed Biodiversity Data Standard. *PLoS ONE* 7, e29715 (2012)

8. Kennedy, J., Hyam, R., Kukla, R., Paterson, T.: A Standard Data Model Representation for Taxonomic Information. *Omics, A Journal of Integrative Biology* 10, 220–230 (2006)
9. Hyam, R., Kennedy, J.: Taxon Concept Schema – User Guide. Unpublished Report, 28 p. (2005)
10. Yoder, M.J., Mikó, I., Seltmann, K.C., Bertone, M.A., Deans, A.R.: A Gross Anatomy Ontology For Hymenoptera. *PloS One* 5, e15991 (2010)
11. The Plant Ontology Consortium: The Plant Ontology™ Consortium and Plant Ontologies. *Comparative and Functional Genomics* 3, 137–142 (2002)
12. Webb, C., Baskauf, S.: Darwin-SW: Darwin Core Data for the Semantic Web
13. Michener, W.K., Brunt, J.W., Helly, J.J., Kirchner, T.B., Stafford, S.G.: Nongeospatial Metadata for the Ecological Sciences. *Ecological Applications* 7, 330–342 (1997)
14. Johnson, J.C., Christian, R.R., Brunt, J.W., Hickman, C.R., Waide, R.B.: Evolution of Collaboration within the US Long Term Ecological Research Network. *BioScience* 60, 931–940 (2010)
15. Williams, J.R., Martinez, N.D., Golbeck, J.: Ontologies for Ecoinformatics. *Web Semantics: Science, Services and Agents on the World Wide Web* 4, 237–276 (2006)
16. Madin, J., Bowers, S., Schildhauer, M., Krivov, S., Pennington, D., Villa, F.: An Ontology for Describing and Synthesizing Ecological Observation Data. *Ecological Informatics* 2, 279–296 (2007)
17. Michener, W.K., Beach, J.H., Jones, M.B., Ludäscher, B., Pennington, D.D., Pereira, R.S., Rajasekar, A., Schildhauer, M.: A Knowledge Environment for the Biodiversity and Ecological Sciences. *Journal of Intelligent Information Systems* 29, 111–126 (2007)
18. Michener, W.K., Jones, M.B.: Ecoinformatics: Supporting Ecology as a Data-Intensive Science. *Trends in Ecology & Evolution* 27, 85–93 (2012)
19. De Giovanni, R., Cartolano, E., Giannini, T., Saraiva, A., Pizzigatti, P.: Darwin Core Interaction Extension Concept List, <http://wiki.tdwg.org/twiki/bin/view/DarwinCore/InteractionExtension>
20. De Giovanni, R., Cartolano, E., Giannini, T., Saraiva, A., Pizzigatti, P.: Darwin Core Interaction Extension: Pollination Extension Concept List, <http://wiki.tdwg.org/twiki/bin/view/DarwinCore/PollinationExtension>
21. Coetzer, W., Gon, O., Hamer, M., Parker-Allie, F.: A New Era for Specimen Databases and Biodiversity Information Management in South Africa. *Biodiversity Informatics* 8, 1–11 (2012)
22. Raven, P.H., Evert, R.F., Eichhorn, S.E.: *Biology of Plants*. Worth Publishers, Inc., New York (1986)
23. Kevan, P.G., Baker, H.G.: Insects as Flower Visitors and Pollinators. *Annual Review of Entomology* 28, 407–453 (1983)
24. Pauw, A.: Floral Syndromes Accurately Predict Pollination by a Specialized Oil-Collecting Bee (*Rediviva peringueyi*, Melittidae) in a Guild of South African Orchids (*Coryciinae*). *American Journal of Botany* 93, 917–926 (2006) ST – Floral syndromes accurately predict
25. Whitehead, V.B., Steiner, K.E.: Oil-collecting Bees of the Winter Rainfall Area of South Africa. *Annals of The South African Museum* 108, 143–277 (2000)
26. Whitehead, V.B., Steiner, K.E., Eardley, C.D.: Oil Collecting Bees Mostly of the Summer Rainfall area of Southern Africa (Hymenoptera: Melittidae: *Rediviva*). *Journal of the Kansas Entomological Society* 81, 122–141 (2008)
27. Horridge, M.: *A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3* (2011)

28. Arp, R., Smith, B.: Function, Role, and Disposition in Basic Formal Ontology. *Nature* 2, 1–4 (2008)
29. Murphy, C.M., Breed, M.D.: Nectar and Resin Robbing in Stingless Bees. *American Entomologist* 36–44 (Spring 2008)
30. Hebert, P.D.N., Cywinska, A., Ball, S.L., DeWaard, J.R.: Biological identifications through DNA Barcodes. *Proceedings of the Royal Society B: Biological Sciences* 270, 313–321 (2003)
31. Dupont, Y.L., Padron, B., Olesen, J.M., Petanidou, T.: Spatio-Temporal Variation in the Structure of Pollination Networks. *Oikos* 118, 1261–1269 (2009)
32. Kaiser-Bunbury, C.N., Muff, S., Memmott, J., Müller, C.B., Caffisch, A.: The Robustness of Pollination Networks to the Loss of Species and Interactions: A Quantitative Approach Incorporating Pollinator Behaviour. *Ecology Letters* 13, 442–452 (2010)
33. Valdovinos, F.S., Ramos-Jiliberto, R., Flores, J.D., Espinoza, C., López, G.: Structure and Dynamics of Pollination Networks: The Role of Alien Plants. *Oikos* 118, 1190–1200 (2009)
34. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs (2003)
35. Moodley, D., Simonis, I., Tapamo, J.: An Architecture for Managing Knowledge and System Dynamism in the Worldwide Sensor Web. *International Journal of Semantic Web and Information Systems: Special issue on Semantics-enhanced Sensor Networks. Internet of Things and Smart Devices* 8, 64–88 (2012)
36. Wiebes, J.T.: Co-evolution of Figs and Their Insect Pollinators. *Annual Review of Ecology and Systematics* 10, 1–12 (1979)
37. Sala, A., Bergamaschi, S.: A Mediator Based Approach to Ontology Generation and Querying of Molecular and Phenotypic Cereals Data. *International Journal of Metadata, Semantics and Ontologies* 4(1/2), 85–92 (2009)

Mapping Keywords to Linked Data Resources for Automatic Query Expansion

Isabelle Augenstein¹, Anna Lisa Gentile¹, Barry Norton²,
Ziqi Zhang¹, and Fabio Ciravegna¹

¹ Department of Computer Science, University of Sheffield, UK

² Ontotext, UK

{i.augenstein,a.l.gentile,z.zhang,f.ciravegna}@dcs.shef.ac.uk,
barry.norton@ontotext.com

Abstract. Linked Data is a gigantic, constantly growing and extremely valuable resource, but its usage is still heavily dependent on (i) the familiarity of end users with RDF's graph data model and its query language, SPARQL, and (ii) knowledge about available datasets and their contents. Intelligent keyword search over Linked Data is currently being investigated as a means to overcome these barriers to entry in a number of different approaches, including semantic search engines and the automatic conversion of natural language questions into structured queries. Our work addresses the specific challenge of mapping keywords to Linked Data resources, and proposes a novel method for this task. By exploiting the graph structure within Linked Data we determine which properties between resources are useful to discover, or directly express, semantic similarity. We also propose a novel scoring function to rank results. Experiments on a publicly available dataset show a 17% improvement in Mean Reciprocal Rank over the state of the art.

1 Introduction

Linked Open Data grows at an astonishing rate, and is becoming a gigantic data source that can be harnessed to power various applications. One of the current challenges for accessing and consuming Linked Data on the Web is dealing with heterogeneity [8], i.e. that data can be distributed and duplicated in different data stores, and described by different vocabularies. As a result, accessing and consuming Linked Data on the Web requires end users to be aware of where to find the data (which datasets) and how it is represented (which vocabularies). Additionally, users need to have working knowledge of formal query languages such as SPARQL to access the datasets. This creates high barriers of entry to creating applications that use Linked Data.

A simple and convenient way for users to express their information needs [8] is the usage of natural language queries (*NLQs*). Therefore, we assume that enabling a user to use natural language to query Linked Data would offer a conducive alternative to formal query languages. The first step in translating a NLQ to SPARQL is to map the keywords in the query to Linked Data resources. Most current semantic search approaches offer limited abstraction from the keywords contained in the query, i.e. the mappings are based merely on keywords and their lexical derivatives ([14], also [18], [17]).

Approaches that just consider variations on a lexicographic level often fail entirely to map keywords to Linked Data resources [17]. More advanced approaches therefore enrich their query with semantically similar words, for example by looking up synonyms in a dictionary, often WordNet ([18], [17], [12]), but dictionaries in general contain a very limited vocabulary and even WordNet has very few named entities.

Freitas et al. [7] argue that matching natural language terms to resources in Linked datasets can easily cross taxonomic boundaries and suggest that expanding the query with a more general semantically similar class of terms produces better results than either matching keywords on the mere base of lexicographic similarity or by WordNet-based expansion. Semantic similarity describes the strength of semantic association between two words. It includes the classical relations hypernymy, hyponymy, meronymy and synonymy. Semantic relatedness is more general than similarity and also covers antonymy and other semantic associations [2].

This idea is further supported by recent studies on exploratory query expansion, where users are presented with not only semantically similar, but also generally related keywords [9]. However, the information about related keywords is usually presented to the user but not directly integrated into the ranking of results [4,16]. Furthermore, existing approaches are typically based on a single knowledge source such as WordNet [16], or Wikipedia [7], and are bound to the specific structure of the knowledge source assumed to be known a-priori. We argue that such methods are highly restricted by the scope of the knowledge source and difficult to adapt across domains.

The major contribution of this paper is the basis for a highly generic, adaptable *automatic query expansion* approach that (i) automatically finds and ranks URIs of related resources for keywords, and (ii) that does not impose a-priori knowledge about data sources or knowledge bases. Compared to the foregoing work of Freitas et al. our contribution is to (i) offer a means to benefit from the multitude of properties between resources in Linked Data, not just the use `wikiPageWikiLink` property, representing the internal Wikipedia links as used in that foregoing work; (ii) offer means to automatically rank the effectiveness of such properties in expressing a useful notion of semantic similarity; (iii) to move beyond simply considering Wikipedia/DBpedia and demonstrate effectiveness across large datasets.

Our approach hinges on the ability to take a keyword from the query, w and expand this into a larger set of keywords, E_w , based on the labellings of ‘neighbours’ in whole graph of Linked Open Data. In the current work this expanded keyword set is then used to find candidates within a fixed target vocabulary (i.e. we use the whole Linking Open Data Cloud to help find alternative keywords via similar resources but evaluate these by using them to find terms in the DBpedia ontology for the purposes of cross-evaluation). The training phase of our approach is intended to apply supervised learning to identify which relationships (properties) among those available in Linked Open Data express useful semantic similarities.

Compared to the state of the art, this approach is completely independent of the structure of underlying datasets and vocabularies. It is therefore highly generic and adaptable across different, and growing, domains covered by Linked Open Data. It is

also highly flexible; as we show that it can be used to expand keywords based on any generalised or specialised semantic relations given suitable training data. We report experiments on the gold standard used in [10]. Results show a 17% improvement in Mean Reciprocal Rank with respect to figures reported in [10].

The paper is structured as follows: Section 2 discusses relevant related work, Section 3 describes our query expansion method and Section 4 presents an evaluation of our method on the DBpedia ontology dataset. We discuss future work directions and conclude in Section 5.

2 Related Work

Freitas et al. [8] discuss the challenges and limitations of querying Linked Data. They identify the main streams of works addressing them, including those based on Information Retrieval (*IR*) methods and those based on Natural Language Processing (*NLP*).

IR based approaches provide, at different levels, an exploration of the underlying datasets in a Web search style, i.e. one which treats documents as *bags of words*, where each word is a mere token. The basic idea behind IR methods is to apply keyword search directly over Linked Data triples. To achieve this goal the triples are tokenised and indexed according to one of several possible strategies. *Watson* [6] is among the class of Semantic Search engines that crawl, analyse and (token) index semantic data and ontologies. *Sindice* [14] also provides a token index for textual search over Semantic Web resources, as well as a SPARQL endpoint over the merged graph-indexed data it has crawled, and allows (cached) retrieval of resources whether found by textual or graph (SPARQL) matching. The graph processor extracts and indexes all textual property values — including, but not limited to labels — for each resource in the graph and tokenises these. In the case that the property attaching a resource to a textual value is through an inverse functional property, this allows searching for blank nodes (which do not have an explicit identifier). *SearchWebDB* [15,16] allows users to formulate queries in terms of keywords, which are translated to structured queries representing possible interpretations of the user's information needs. Returned results are the triples satisfying those queries, ranked with decreasing syntactic similarity from the original query keywords. The keyword matching process is limited to the literals associated with each objects, but neighbour resources are also retrieved to help the user select the best answer. In a subsequent work, semantically similar entries from WordNet are also used for the keyword matching process [16]. *Falcons Object Search* [4] performs the keyword matching process against labels, literals and textual descriptions of each object and immediately linked objects. A drawback of the search is the assumption that keywords in the query directly match labels for resources, although users can iteratively refine the query with additional keywords. Ranking of results is performed according to text similarity (tf-idf), weighted by the popularity of each object. The popularity is a score calculated at index time, as a logarithmic function of the number of documents where the object is mentioned.

NLP based methods attempt to derive a SPARQL query from a natural language question (NLQ), returning a single answer or the set of results compatible with the user question. These approaches mainly rely on linguistic resources to expand and disambiguate keywords, with the drawbacks of coverage and lack of generality in the relationships considered mentioned earlier. *FREyA* [5] is a natural language interface for querying ontologies. It performs syntactic parsing of the NLQ and involves the user in the translation process if necessary. The key feature of the approach is that the user's choices are used for training the system in order to improve its performance over time. *Gingseng* [1] and *NLP reduce* [11] are two natural language interface question answering systems. *NLP reduce* allows the user to enter either full natural language questions, sentence fragments, or just keywords. The question is reduced to a set of query terms, which are matched against a precomputed index of the ontology lexicon, where for all lexicon terms stemming and synonym expansion have been performed. *Gingseng* does not provide full natural language interface, but rather guides user input with respect to the considered ontologies, suggesting term and relations present in the underlying ontologies. *PowerAqua* [12] implements a strategy for (i) combining and (ii) ranking answers from distributed heterogeneous data sources in the context of a multi-ontology question answering task. The result list integrates triples coming from different data stores, ranked using a multi-level ranking algorithm, which combines three different criteria (popularity, confidence and semantic interpretation). Two preprocessing steps are performed to translate the natural language question. A linguistic component extracts relevant terms from the question, then a mapping component identifies potentially suitable ontologies for the terms searching for occurrences of the terms, but also synonyms, hypernyms and hyponyms.

Freitas et al. [7] argue that combining NLP and IR techniques with a richer exploitation of the graph structure beyond involved entities, would significantly improve performance of the overall process. They propose a method which allows end users to query Linked Data using natural language queries. The search is performed in three steps. First, they recognise key entities in the query (Named Entities and lexicon terms); then they retrieve potential DBpedia entries matching each entity. The evaluation is performed against the DBpedia portion of the QALD evaluation query set ¹. As a general approach, we are inclined to follow directions in [7], as we will discuss in Section 5, but the aim of this paper is a focused evaluation of the preprocessing step which translates and expands terms in the NLQ in a set of Linked Data resources. Although both NLP-centric works and IR based methods point out the importance of such a preprocessing step, they do not provide an in vitro evaluation of such task. Keyword expansion in queries has been extensively addressed in the traditional IR settings [3], but it is still mostly untapped in the context of Linked Data, as mentioned in [9]. We follow the evaluation proposed by Freitas et al. [9], who created a test set for this specific task on Linked Data and elaborate on their proposed method, by providing a richer exploitation of the graph structure around involved concepts.

¹ <http://www.sc.cit-ec.uni-bielefeld.de/sites/www.sc.cit-ec.uni-bielefeld.de/files/sharedtask.pdf>

Table 1. *Labelling* properties

rdfs:label
foaf:name
dc:title
skos:prefLabel
skos:altLabel
fb:type.object.name

Table 2. *Labelling*⁺ properties

foaf:name	0.26666668
rdfs:label	0.18666667
dc:title	0.16867469
sindice:label	0.16842106
skos:prefLabel	0.15730338
commontag:label	0.13333334
skos:altLabel	–
fb:type.object.name	–

3 Method

At the start of our approach we have a keyword w for which the user wants to find representative resources in Linked Data. The dataset² we operate on consists of a set of resources, R , including a set of properties used to denote specific relationships between resources, $P \subset R$. Literals are a further subset of resources $L \subset R$ with a lexical representation, which are disjoint from P (i.e. $P \cap L = \emptyset$). As usual, for RDF, the data consists of a set of statements, $S \subset (R \setminus L) \times P \times R$, each being a 3-tuple (triple) consisting of a ‘subject’, ‘predicate’ and an ‘object’. In the current application, and the method described here, this is applied in a more focussed setting, where the intention is to find concepts (classes and properties) in a fixed vocabulary; we call the target classes $TC \subset R$, where for each $c \in TC$ we have $(c, \text{rdf:type}, \text{rdfs:Class}) \in S$ and target properties $TP \subset P$, where for each $p \in TP$ we have $(p, \text{rdf:type}, \text{rdf:Property}) \in S$.

We have chosen a set of labelling properties, i.e. properties whose values are expected to be literals which might be worthwhile in identifying distinct concepts, $Labelling \subset P$, shown in Table 1. We make claims neither that this is a complete nor, in each part, useful; indeed Table 2 shows that the method itself reveals two further properties we had not considered (in bold) and that, according to the precision metric we explain below, two that we did include were not useful in the evaluation.

In order to find representative concepts we construct from w an expanded set of keywords, E_w , that improve the chance of finding the most fitting concept in the target vocabulary according to its labelling (under the properties *Labelling*). These expanded sets are more general than ‘synsets’ (sets of synonyms within dictionary-oriented terms) in both scope, including a huge potential range of named entities, and in the flexibility of the semantic relationships covered. A distinctive feature of our method is that the data is not presumed or intended to be governed by the vocabulary targetted but rather the results improve as the dataset considered increases beyond the target (as we shall show in the next section, we cross-evaluate against a method choosing terms from the DBpedia ontology, but use a much larger portion of the Sindice cache to find the expanded set).

As well as exploiting the labels of resources across the whole dataset, our algorithm learns a rank over all properties, not just those used for labelling, according to how

² N.B. in terms of the Linking Open Data Cloud this merged graph can, and does in our evaluation, span several individual datasets in the sense they are registered at the Data Hub.

useful they are in expressing useful notions of semantic similarity. This is achieved by a supervised training period, described in Section 3.2, in which the first stages of the algorithm, forming the expanded set and then using this to find candidates in the target vocabulary, is followed by a manual choice between candidates. As a result of this training, we learn a precision for the properties and rank then subset these to create an ordered list $\vec{Rel} \subset P$ that the final part of the algorithm can then use to automatically distinguish between the fitness of candidates, as described in Section 3.3.

3.1 Candidate Identification

In the first stage of the algorithm, i.e. lines 1-7 in Algorithm 1, triples in the dataset where the keyword w is used as the label of a resource, r , are identified, via any labelling property $labelling1 \in Labelling$. Next the neighbours of r , who are the objects of any relationship $p \in Q$ (where $Q \subseteq P$ is a parameter) from this subject resource, are found. These neighbours fall into two groups: the first are themselves literals; the second are not literals, but may have literals attached via a labelling property $labelling2 \in Labelling$. The resulting set of labels is called an ‘expanded set’, E_w , relative the original w .

Algorithm 1. identifyCandidates(w, Q)

```

1: for all  $r$  such that  $(r, labelling1, w) \in S$  and  $labelling1 \in Labelling$  do
2:   for all  $s$  such that  $(r, p, s) \in S$  and  $p \in Q$  do
3:     if  $s \in L$  then
4:        $E_w \leftarrow s$ 
5:     else
6:        $E_w \leftarrow \{lit \mid (s, labelling2, lit) \in S \text{ and } labelling2 \in Labelling\}$ 
7:     end if
8:      $Cand_w \leftarrow Cand_w \cup \{(p, findTargetsFromExpandedLabels(E_w))\}$ 
9:   end for
10: end for

```

This expanded set is used, for each examined property p , to find a list of terms from the target ontology by Algorithm 2.

Algorithm 2. findTargetsFromExpandedLabels(Labels)

```

for all  $label \in Labels$  do
   $Tokens = \{label\} \cup tokenise(label)$ 
end for
for all  $token \in Tokens$  do
  for all  $t$  such that  $(t, labelling3, token) \in S$  and  $labelling3 \in Labelling$  and  $t \in TC \cup TP$  do
     $Targets \leftarrow Targets \cup \{t\}$ 
  end for
end for
return  $Targets$ 

```

Algorithm 2 relies on a function *tokenise* which splits multi-word labels into a set of words, i.e. $\text{tokenise}(\text{"programming language"}) = \{\text{"programming"}, \text{"language"}\}$.

Finally, having obtained the targets according to the given property, Algorithm 1 pairs that property and the target resources found in the set $\text{Cand}_w \subset P \times \mathcal{P}(R)$, or properly $Q \times \mathcal{P}(R)$. At this stage the method branches according to whether we are training, or running automatically with respect to an existing training period, which has defined a ranked short-list $\vec{Rel} \subset P$ that express a useful notion of semantic similarity.

3.2 Training

In order to train, we select a list of keywords to stand for w , with P (i.e. the whole set of properties in the dataset) standing for Q , in several iterations of the algorithm presented in Section 3.1. This training set, $\vec{W}^{\text{train}} = \{w_1^{\text{train}}, w_2^{\text{train}}, \dots, w_n^{\text{train}}\}$, consists of nouns and compound nouns in general use; our only constraint in the evaluation was that this set is disjoint from the keywords used in the test set, as described in Section 4.

The aim of training is to produce a precision measure for every property in P^{train} , that is every property exercised in finding candidates across \vec{W}^{train} , i.e.:

$$P^{\text{train}} = \{p \mid (p, T) \in \text{Cand}_w \text{ where } w \in \vec{W}^{\text{train}}\}$$

The first stage is to manually choose the best (not necessarily unique) among the candidates under each p that occurs in Cand_w , for each $w \in \vec{W}^{\text{match}}$. We therefore first generalise Cand_w to $\vec{\text{Cand}}^{\text{train}}$, that is an ordered set matching \vec{W}^{train} such each $\text{Cand}_i^{\text{train}} = \text{Cand}_{w_i^{\text{train}}}$.

Based on the manual selection we then also have an ordered set \vec{Best} such that $\text{Best}_{w,p} \subset T_{w,p}$ where $(p, T_{w,p}) \in \text{Cand}_w$ for each $w \in \vec{W}^{\text{train}}$.

In order to compute precision for each p , we now define two functions $\text{hits}(p, w)$ and $\text{candidates}(p, w)$, both defined for each $p \in P^{\text{match}}$ and $w \in \vec{W}^{\text{train}}$, as follows:

$$\begin{aligned} \text{hits}(p, w_i^{\text{train}}) &= \{t \mid t \in T \text{ where } (p, T) \in \text{Cand}_i^{\text{train}} \text{ and } t \in \text{Best}_{w_i^{\text{train}}, p}\} \\ \text{candidates}(p, w_i^{\text{train}}) &= \{t \mid t \in T \text{ where } (p, T) \in \text{Cand}_i^{\text{train}}\} \end{aligned}$$

We can then apply the following simple calculation:

$$\text{prec}(p) = \frac{\sum_{w \in \vec{W}^{\text{train}}} \text{hits}(p, w)}{\sum_{w \in \vec{W}^{\text{train}}} \text{candidates}(p, w)}$$

We then define some threshold $0 \leq \theta \leq 1$, and use prec to define an ordered (ranked) subset of properties shown to encode semantic relatedness, \vec{Rel} , as the greatest set with the following definition:

$$\text{rel}_i \in \vec{Rel} \text{ iff } \text{rel}_i \in P^{\text{match}} \wedge \text{prec}(\text{rel}_i) > \theta \wedge \nexists j \text{ such that } j > i \wedge \text{prec}(j) < \text{prec}(i)$$

3.3 Judging Candidate Fitness

Having obtained the ranked list of properties for semantic relatedness by training, we can now take any set of candidates over a test set of keywords, $\overrightarrow{W}^{test}$, and apply Algorithm 1 to obtain candidates. In this case however we take only properties from $\overrightarrow{R}el$ to form the other parameter, Q . As a result we assume $\overrightarrow{Cand}^{test}$ matching the definition in Section 3.2, over properties P^{test} , again assuming a matching definition (note, by design: $P^{test} \subseteq \overrightarrow{R}el$).

In order to judge fitness of each candidate resource r in $Cand_i^{test}$ (i.e. where $r \in R$ and $(p, R) \in Cand_i^{test}$ for some p) to the corresponding keyword, w_i^{test} , we combine, as a numerical product, the precision of the property, p , used to find that candidate and a tf-idf score for r , computed as follows:

$$tf(r, w_i^{test}) = \frac{\sum_{p \in \overrightarrow{R}el} \begin{matrix} 1 & \text{if } r \in R \\ 0 & \text{if } r \notin R \end{matrix} \text{ where } (p, R) \in \overrightarrow{Cand}_i^{test}}{1 + \sum_{v \in \overrightarrow{W}^{train}} \sum_{p \in \overrightarrow{R}el} \begin{matrix} 1 & \text{if } r \in R \\ 0 & \text{if } r \notin R \end{matrix} \text{ where } (p, R) \in \overrightarrow{Cand}^{train}}$$

And the ‘inverse’ (i.e., logarithmic) document frequency as:

$$idf(r, \overrightarrow{W}^{train}) = \log \frac{|\overrightarrow{W}^{train}|}{1 + \sum_{v \in \overrightarrow{W}^{train}} \begin{matrix} 1 & \text{if } \exists p, R \text{ such that } r \in R \text{ and } (p, R) \in \overrightarrow{Cand}^{train} \\ 0 & \text{otherwise} \end{matrix}}$$

4 Evaluation

For evaluation purposes we used the DBpedia ontology as the target vocabulary (i.e. TC were DBpedia ontology classes and TP were DBpedia ontology properties), and a dataset of the Sindice cache (therefore S consists of many statements from datasets registered at the Data Hub³, as shown in the Linking Open Data Cloud, together with further semantic data found by crawling).

4.1 Gold Standard and Evaluation Metric

We derived a gold standard, containing 178 keywords of which 134 have at least one representation in the DBpedia ontology, from the experiment described in [9]. When examining this paper’s set up and results, we detected some minor errors which we have corrected in our cross-evaluation. In re-evaluating the approach used by Freitas et al. [9] the actual figures for their approach did not change significantly with respect to our improvements (0.6 vs. 0.64 in the original evaluation).

The first class of correction actually rates Freitas et al.’s results higher than their own evaluation as the keyword *beatles* was marked as not available in the DBpedia ontology (N/A), although their results list contained the DBpedia class `dbpedia-owl:Artist`, which we score as a hit and credit them for this in all such cases. Another group of errors concern keywords that are marked as N/A , where the correct result is not contained in

³ <http://datahub.io/>

Table 3. Top 23 properties used and their precision

foaf:name	0.267	owl:sameAs	0.121
fb-common:topic	0.189	wn20schema:gloss	0.1
rdfs:label	0.187	opencyc:seeAlsoURI	0.093
dc:subject	0.182	rdfs:seeAlso	0.082
dc:title	0.169	dbpedia-owl:abstract	0.0774
sindice:label	0.168	rdfs:comment	0.0676
rdfs:subClassOf	0.168	rdfs:range	0.0667
skos:prefLabel	0.157	rdfs:subClassOf	0.0656
fb-type:object	0.143	fb:documented_object	0.0619
wn20schema:derivationallyRelated	0.143	dbpedia-owl:wikiPageWikiLink	0.0487
wn20schema:containsWordSense	0.138	dc:description	0.0471
commenttag:label	0.133		

the results set but actually available in DBpedia (e.g. `dbpedia-owl:manufacturer` and `dbpedia-owl:Automobile` for the keyword *honda*).

Additionally, some results were marked as correct matches although on closer inspection there are better candidates; for example, for the keyword *feline*, the match `dbpedia-owl:genus` was accepted, although a less general result is `dbpedia-owl:Mammal`.

Finally, there were some inconsistencies between Freitas et al.'s self evaluation when comparing the result table and their downloaded gold standard (i.e., the correct answer was marked in the gold standard, but marked as *N/A* in the result table).

The updated dataset and the results of all evaluations are available for download⁴.

Freitas et al. [7] evaluate query expansion as an information retrieval task, where expansion candidates are ranked according to their relevance. The measure used for evaluation is *Mean Reciprocal Rank* (MRR) which measures the quality of the ranking by calculating the inverse rank of the best result. In this study, we follow the same approach. That is, we apply MRR to the resulting ranked list of candidates given by our algorithm.

4.2 Approach

Since our approach requires a supervised training phase, we manually created a training set following the same principles outlined in [7]. The training set contains 40 keywords. After applying the training phase, 194 candidate relations are learnt. We used a precision threshold of 0.045 to cut off the candidate relation list. This resulted in 23 relations which can be found in Table 3.

As well as producing, when projected over labelling properties (those with plain literal values), an interesting comparison (Table 2) with our original set of labelling properties (Table 1), as observed earlier, this list also brings to light a common but useful error in the Sindice-crawled data, the misspelling `subClassOf`, which turns out to be higher precision (0.168) than the correct predicate `subClassOf` (0.0656).

⁴ <http://oak.dcs.shef.ac.uk/LODIE/know-a-1od-2013>

Table 4. Set of example results

Query: [spacecraft]	Query: [engine]	Query: [factory]
dbpedia-owl:Spacecraft dbpedia-owl:spacecraft dbpedia-owl:satellite dbpedia-owl:missions dbpedia-owl:launches dbpedia-owl:closed dbpedia-owl:vehicle dbpedia-owl:Rocket	dbpedia-owl:engine dbpedia-owl:gameEngine dbpedia-owl:Artwork dbpedia-owl:AutomobileEngine dbpedia-owl:Locomotive dbpedia-owl:fuel dbpedia-owl:added dbpedia-owl:Musical	dbpedia-owl:manufacturer dbpedia-owl:plant dbpedia-owl:Canal dbpedia-owl:engine dbpedia-owl:class dbpedia-owl:Album dbpedia-owl:product dbpedia-owl:assembly
Query: [bass]	Query: [wife]	Query: [honda]
dbpedia-owl:Fish dbpedia-owl:Instrument dbpedia-owl:instrument dbpedia-owl:voice dbpedia-owl:partner dbpedia-owl:note dbpedia-owl:Musical dbpedia-owl:lowest	dbpedia-owl:spouse dbpedia-owl:Criminal dbpedia-owl:person dbpedia-owl:status dbpedia-owl:education dbpedia-owl:Language dbpedia-owl:sex dbpedia-owl:family	dbpedia-owl:manufacturer dbpedia-owl:discovered dbpedia-owl:Asteroid dbpedia-owl:engine dbpedia-owl:season dbpedia-owl:vehicle dbpedia-owl:Automobile dbpedia-owl:participant

4.3 Results

In order to illustrate the output of our method, Table 4 lists keyword queries and their ranked lists. The correct results the user chose in the evaluation are marked, as in the gold standard, in bold. These example queries illustrate that our approach is able to find resources of extended keywords based on different kinds of semantic relationships. These include identity (e.g. spacecraft - `dbpedia-owl:spacecraft`), hyponymy (e.g. engine - `dbpedia-owl:AutomobileEngine`), hypernymy (e.g. wife - `dbpedia-owl:spouse`, `dbpedia-owl:person`, `dbpedia-owl:family`), meronymy (e.g. honda - `dbpedia-owl:engine`), and synonymy (e.g. factory - `dbpedia-owl:plant`). Our approach is also able to different senses for ambiguous keywords (e.g. bass - `dbpedia-owl:Fish`, `dbpedia-owl:Instrument`, `dbpedia-owl:instrument`). The authors leave it to the reader’s imagination to explain why the class `Criminal` is ranked so highly in the matches for the keyword “wife”.

The results of our evaluation are shown in Tables 5 and 6. Our MRR (*LOD Keyword Expansion*) is 0.77, which means most of the best results are located on the first position. We can further show 17% improvement in MRR in comparison with the Wikipedia based approach (*ESA*) Freitas et al. [9] followed. Overall, *LOD Keyword Expansion* is able to answer 90% of the keyword queries (only taking into account keywords that are represented in DBpedia), which is 3% more than *ESA*. We also compare our approach to a simple *String match* baseline and a *String match* baseline enriched with WordNet synonyms (*String match + WordNet*) [9]. Our approach is able to answer twice as many keyword queries as *String match* and 38% more than *String match + WordNet*.

Table 5. Mean reciprocal rank (MRR)

Model	MRR
ESA	0.6
LOD Keyword Expansion	0.77

Table 6. Percentage queries answered (Recall)

Model	Recall
String match	0.45
String match + WordNet	0.52
ESA	0.87
LOD Keyword Expansion	0.90

5 Conclusion and Outlook

In this paper, we have described a method for automatic query expansion for Linked Data resources which is based on using properties between resources within the Linking Open Data cloud. In our evaluation, we examined how useful these different properties are for finding semantic similarities and thereby finding alternative (expanded) keywords, and applied our method to the DBpedia ontology as a target. Compared to the state of the art [9], we show an improvement of 17% in Mean Reciprocal Rank.

The approach described currently bases query expansion on semantic similarity. Related work discussed above [17] follows a multi-strategy approach; first using string similarity then, if the desired result is not achieved, lexical expansion with WordNet, and finally, where these simpler methods have failed, using ESA (as in [9]). They show that the best result is achieved by combining all these approaches. We would speculate, therefore, that our results could be achieved with such a hybrid approach, bringing to bear our general means for semantic similarity only after the lower-hanging fruit has been picked off.

Future work will also re-apply some feedback based on the approach documented here. Firstly, as shown in Tables 1 versus 2, our set of labelling properties could be adjusted, extending the reach of the method. Secondly, once this set is fixed we could fine-tune the algorithm by learning over the set of properties that express semantic similarity separately from the labelling properties (which are currently discarded from Algorithm 2).

While this work focused on the evaluation of the keyword expansion process per se, we intend to perform an in vivo evaluation of the task, by performing a semantic search evaluation. One way we foresee for privileging one concept rather than the other in the simultaneous matching is exploiting Encyclopaedic Knowledge patterns (EKPs) [13], to exploit not only available relations between candidate concepts, but also statistical evidence in the data about connections which have been actually used in instance data.

References

1. Bernstein, A., Kaufmann, E., Kaiser, C.: Querying the semantic web with ginseng: A guided input natural language search engine. In: 15th Workshop on Information Technologies and Systems, Las Vegas, NV, pp. 112–126 (2005)
2. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Journal of Computational Linguistics* 32(1), 13–47 (2006)

3. Carpineto, C., Romano, G.: A Survey of Automatic Query Expansion in Information Retrieval. *ACM Comput. Surv.* 44(1), 1 (2012)
4. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: *Proceedings of the 17th International Conference on World Wide Web*, pp. 1101–1102. ACM (2008)
5. Damljanovic, D., Agatonovic, M., Cunningham, H.: FREyA: An interactive way of querying Linked Data using natural language. In: García-Castro, R., Fensel, D., Antoniou, G. (eds.) *ESWC 2011. LNCS*, vol. 7117, pp. 125–138. Springer, Heidelberg (2012)
6. d’Aquin, M., Baldassarre, C., Gridinoc, L., Sabou, M., Angeletou, S., Motta, E.: Watson: Supporting next generation semantic web applications. In: *Proceedings of the WWW/Internet Conference*, Vila real, Spain (2007)
7. Freitas, A., Curry, E., Oliveira, J.G., O’Riain, S.: A Distributional Structured Semantic Space for Querying RDF Graph Data. *International Journal of Semantic Computing* 5(04), 433–462 (2011)
8. Freitas, A., Curry, E., Oliveira, J.G., O’Riain, S.: Querying heterogeneous datasets on the linked data web: Challenges, approaches, and trends. *IEEE Internet Computing* 16(1), 24–33 (2012)
9. Freitas, A., Curry, E., O’Riain, S.: A distributional approach for terminological semantic search on the linked data web. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 384–391. ACM (2012)
10. Freitas, A., Oliveira, J., O’Riain, S., Curry, E., Pereira da Silva, J.: Querying Linked Data using semantic relatedness: A vocabulary independent approach. *Natural Language Processing and Information Systems*, 40–51 (2011)
11. Kaufmann, E., Bernstein, A., Fischer, L.: NLP-Reduce: A “naive” but Domain-independent Natural Language Interface for Querying Ontologies. In: *4th European Semantic Web Conference* (2007)
12. Lopez, V., Nikolov, A., Fernandez, M., Sabou, M., Uren, V., Motta, E.: Merging and ranking answers in the semantic web: The wisdom of crowds. *The Semantic Web*, 135–152 (2009)
13. Nuzzolese, A.G., Gangemi, A., Presutti, V., Ciancarini, P.: Encyclopedic knowledge patterns from wikipedia links. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) *ISWC 2011, Part I. LNCS*, vol. 7031, pp. 520–536. Springer, Heidelberg (2011)
14. 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)
15. Tran, T., Wang, H., Haase, P.: SearchWebDB: Data web search on a pay-as-you-go integration infrastructure. Tech. rep., Technical report, University of Karlsruhe (2008)
16. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: *IEEE 25th International Conference on Data Engineering, ICDE 2009*, pp. 405–416. IEEE (2009)
17. Walter, S., Unger, C., Cimiano, P., Bär, D.: Evaluation of a layered approach to question answering over linked data. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) *ISWC 2012, Part II. LNCS*, vol. 7650, pp. 362–374. Springer, Heidelberg (2012)
18. Wang, H., Tran, T., Haase, P., Penin, T., Liu, Q., Fu, L., Yu, Y.: Searchwebdb: Searching the billion triples. In: *Billion Triple Challenge at the International Semantic Web Conference* (2008)

Finding Fault: Detecting Issues in a Versioned Ontology

Maria Copeland, Rafael S. Gonçalves, Bijan Parsia, Uli Sattler, and Robert Stevens

School of Computer Science, University of Manchester, Manchester, UK

Abstract. Understanding ontology evolution is becoming an active topic of interest for ontology engineers, e.g., there exist large collaboratively-developed ontologies but, unlike in software engineering, comparatively little is understood about the dynamics of historical changes, especially at a fine level of granularity. Only recently has there been a systematic analysis of changes across ontology versions, but still at a coarse-grained level. The National Cancer Institute (NCI) Thesaurus (NCIt) is a large, collaboratively-developed ontology, used for various Web and research-related purposes, e.g., as a medical research controlled vocabulary. The NCI has published ten years worth of monthly versions of the NCIt as Web Ontology Language (OWL) documents, and has also published reports on the content of, development methodology for, and applications of the NCIt. In this paper, we carry out a fine-grained analysis of asserted axiom dynamics throughout the evolution of the NCIt from 2003 to 2012. From this, we are able to identify axiomatic editing patterns that suggest significant regression editing events in the development history of the NCIt.

1 Introduction

This paper is part of a series of analyses of the NCIt corpus [1,2], the earlier of which focus on changes to asserted and inferred axioms. The current analysis extends previous work by tracing editing events at the individual axiom level, as opposed to the ontology level. That is, instead of analysing the total number of axioms added or removed between versions, we track the appearance and disappearance of individual axioms across the corpus. As a result, we are able to identify a number of regressions (i.e., inadvertent introduction of error) which occur over the last ten years of development of the NCIt ontology, as well as a number of event sequences that, while not necessarily introducing errors, indicate issues with the editing process. We are able to do this analytically from the editing patterns alone.

2 Preliminaries

We assume that the reader is familiar with OWL 2 [3], at least from a modeller perspective. An OWL ontology \mathcal{O} is a set of axioms, containing logical and non-logical (e.g., annotation) axioms. The latter are analogous to comments in conventional programming languages, while the former describe classes or individuals and the relations between these via properties. The *signature* of an ontology \mathcal{O} (the set of individuals, class and property names used in \mathcal{O}) is denoted $\tilde{\mathcal{O}}$.

We use the standard notion of first order logic *entailment*, and denote an axiom α entailed by an ontology \mathcal{O} as $\mathcal{O} \models \alpha$.

Finally, we can determine whether there exist logical faults in such sequence based on the following definition:

Definition 1 ([4]). Let \mathcal{O}_i and \mathcal{O}_j be two versions of an ontology \mathcal{O} where $i < j$.

- An axiom α is an addition (removal) if $\alpha \in \mathcal{O}_j \setminus \mathcal{O}_i$ ($\alpha \in \mathcal{O}_i \setminus \mathcal{O}_j$).
- An addition α is effectual if $\mathcal{O}_i \not\models \alpha$, denoted $\alpha \in \text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j)$, and ineffectual otherwise, denoted as $\alpha \in \text{IneffAdd}(\mathcal{O}_i, \mathcal{O}_j)$.
- A removal α is effectual if $\mathcal{O}_j \not\models \alpha$, denoted $\alpha \in \text{EffRem}(\mathcal{O}_i, \mathcal{O}_j)$, and ineffectual otherwise, denoted as $\alpha \in \text{IneffRem}(\mathcal{O}_i, \mathcal{O}_j)$.

3 Conceptual Foundations

Prior to the study of fault detection techniques, we establish a clear notion of the type of faults we are trying to isolate. In all cases, we define a *fault* as deviation from the required behaviour. In software engineering, software faults are commonly divided into functional and non-functional depending on whether the fault is in the required functional behaviour (e.g., whether the system is acting correctly with respect to its input, behaviour, and output), or whether the fault is in the expected service the system needs to provide (i.e., whether the (correct) behaviour is performed *well*). Functional and non-functional faults can be further subdivided based on the impact to the system and/or to the requirements specification. For example, functional faults can be divided into fatal and non-fatal errors depending on whether the fault crashes the system. Generally, crashing behaviour is always a fatal fault, however it might be preferable to encounter a system crash instead of a non-fatal fault manifested in some other, harder to detect, manner. Faults that impact the requirements may be implicit, indeterminate (i.e., the behaviour might be *underspecified*), or shifting. A shifting specification can render previously correct behaviour faulty (or the reverse), as faults are defined as deviations from the “governing” specification. For convenience, we presume throughout this study that the specification is stable over the lifetime of the examined ontology, i.e., we expect the notation of ‘acceptable model’ or ‘acceptable entailment’ to be stable throughout the lifetime of the ontology.

We also restrict our attention to the *logical* behaviour of the ontology, and we approximate this by sets of desired entailments from the analysis of effectual and ineffectual changes in consecutive sets of ontologies. This restriction might not reflect the full behaviour of an ontology in some application as 1) many entailments might be irrelevant to the application (e.g., non-atomic subsumptions for a terminologically oriented application) or 2) the application might be highly sensitive to other aspects of the ontology, including, but not limited to, annotations, axiom shape, and naming patterns. However, these other aspects are less standardised from application to application, so are rather more difficult to study externally to a given project. Furthermore, faults in the logical portion of an ontology both can be rather difficult to deal with and affect these other aspects. With this in mind, we define a logical bug as follows:

Definition 2. An ontology \mathcal{O} contains a (logical) bug if $\mathcal{O} \models \alpha$ and α is not a desired entailment or $\mathcal{O} \not\models \alpha$ and α is a desired entailment.

Of course, whether a (non)entailment is desired or not is not determinable by a reasoner — a reasoner can only confirm that some axiom is or is not an entailment. Generally, certain classes of (non)entailments are always regarded as errors. Particularly, and analogously to crashing bugs in software engineering, the following are all standard errors:

- \mathcal{O} is inconsistent, i.e., $\mathcal{O} \models \top \sqsubseteq \perp$
- $A \in \tilde{\mathcal{O}}$ is unsatisfiable in \mathcal{O} , i.e., $\mathcal{O} \models A \sqsubseteq \perp$
- $A \in \tilde{\mathcal{O}}$ is tautological in \mathcal{O} , i.e., $\mathcal{O} \models \top \sqsubseteq A$

In each of these cases, the “worthlessness” of the entailment is straightforward¹ and we will not justify it further here. That these entailments are bugs in and of themselves makes it easy to detect them, so the entire challenge of coping with such bugs is in explaining and repairing them.

Of course, not all errors will be of these forms. For example, in most cases, the subsumption $Tree \sqsubseteq Animal$ would be an undesired entailment. Detecting this, however, requires domain knowledge; specifically, the denotation of `Tree` and `Animal`, the relation between them, and the intent of the ontology. If there is an explicit specification, such as a finite list of desired entailments, then checking for correctness of the ontology would be straightforward. Typically, however, the specification is implicit and, indeed, may be inchoate, only emerging via the ontology development process. Consequently, it would seem that automatic detection of such faults is impossible. This is certainly true when considering a single version of an ontology. The case is different when multiple versions are compared. Crucially, if an entailment *fluctuates* between versions, that is, if it is the case that $\mathcal{O}_i \models \alpha$ and $\mathcal{O}_j \not\models \alpha$ where $i < j$, then we can infer that *one* of those cases is erroneous. However, it is evident that $\mathcal{O}_i \not\models \alpha$ but $\mathcal{O}_j \models \alpha$ may not be, as the fact that $\mathcal{O}_i \not\models \alpha$ might just indicate that a desired entailment had not been introduced yet.

In what follows, we consider a sequence $\mathcal{O}_i, \mathcal{O}_j, \mathcal{O}_k, \mathcal{O}_l, \mathcal{O}_m, \mathcal{O}_n, \mathcal{O}_o, \mathcal{O}_p$ of ontologies, and use i, j, k, l, m, n, o, p as indices for these ontologies with $i < j < k < l < m < n < o < p$.

Definition 3. Given two sequence of ontologies $\mathcal{O}_i, \mathcal{O}_j$, and $\mathcal{O}_k, \mathcal{O}_l$, where $i < j < k < l$, a fault **indicating** set of changes, denoted $\text{FISoC}((i, j), (k, l))$, is:

$$\text{FISoC}((i, j), (k, l)) := \text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j) \cap \text{EffRem}(\mathcal{O}_k, \mathcal{O}_l)$$

Note that, if $\alpha \in \text{FISoC}((i, j), (k, l))$, either the entailment $\mathcal{O}_j \models \alpha$, thus $\alpha \in \mathcal{O}_j$, or the non-entailment $\mathcal{O}_l \not\models \alpha$ is a logical bug, but $\text{FISoC}((i, j), (k, l))$ does not identify which of these two is the logical bug. Instead the fault indicating set tells us that *one* of the changes introduces a bug. As mentioned earlier, the set shows the existence of a bug assuming a stable specification. Any subsequent findings of the same

¹ There is, at least in the OWL community, reasonable consensus that these are all bugs in the sort of ontologies we build for the infrastructure we use.

$\alpha \in \text{FISoC}((m, n), (o, p))$ where $l < m$ is a fault indicate content regression. It is not surprising to find reoccurring content regressions due to the absence of content regression testing.

We can have a similar set of changes wherein the removal is *ineffectual* i.e., $\alpha \in \mathcal{O}_i$, $\alpha \notin \mathcal{O}_j$, but $\mathcal{O}_j \models \alpha$. Though in this case, since the functionality of the ontology is not changed by an ineffectual removal, such a set does not indicate a regression at the ontology level. Indeed, this would be indicative of a *refactoring*. Of course, if the refactoring is the bug, then the ineffectual removal from \mathcal{O}_i to \mathcal{O}_j would be a failed attempt to remove the bug. Without access to developer intentions or other external information, we cannot distinguish between these two situations. However, we can conclude that an iterated pattern of ineffectual changes is problematic. That is, even if the set of changes $\{\text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j) \cap \text{IneffRem}(\mathcal{O}_k, \mathcal{O}_l)\}$ is a refactoring, a subsequent ineffectual addition, $\text{IneffAdd}(\mathcal{O}_m, \mathcal{O}_n)$, would indicate a sort of thrashing. Meaning, if the original refactoring was correct, then “refactoring back” is a mistake (and if the “refactoring back” is correct, then the original refactoring is a mistake).

Definition 4. Given two sets of ontologies, $\mathcal{O}_i, \mathcal{O}_j$ and $\mathcal{O}_k, \mathcal{O}_l$ where $i < j < k < l$, then any of the following sets

$$\begin{aligned} \text{F1SSoC}((i, j), (k, l)) &:= \text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j) \cap \text{IneffRem}(\mathcal{O}_k, \mathcal{O}_l) \\ \text{F2SSoC}((i, j), (k, l)) &:= \text{IneffAdd}(\mathcal{O}_i, \mathcal{O}_j) \cap \text{IneffRem}(\mathcal{O}_k, \mathcal{O}_l) \\ \text{F3SSoC}((i, j), (k, l)) &:= \text{IneffRem}(\mathcal{O}_i, \mathcal{O}_j) \cap \text{IneffAdd}(\mathcal{O}_k, \mathcal{O}_l) \end{aligned}$$

are fault **suggesting** set of changes denoted as $\text{FSSoC}((i, j), (k, l))$.

There is a large gap in the strength of the suggestiveness between sets of kind *F1SSoC* and the sets of kind *F2SSoC* and *F3SSoC*. Sets of kind *F1SSoC* can be completely benign, indicating only that additional information has been added to the axiom (e.g., that the axiom was strengthened), whereas there is no sensible scenario for the occurrence of sets of kind *F2SSoC* and *F3SSoC*. In all cases, much depends on whether the lack of effect of the change is known to the ontology modeller. For instance, if a set of type $\text{F1SSoC}((i, j), (k, l))$ was an attempt to repair α , then α is a logical bug if α is an undesired entailment that was meant to have been repaired in \mathcal{O}_l , then this repair failed.

A combination of *FSSoC* and *FiSoC* sets may be embedded in larger sets. Consider the case $\alpha \in \text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j) \cap \text{IneffRem}(\mathcal{O}_k, \mathcal{O}_l) \cap \text{IneffAdd}(\mathcal{O}_m, \mathcal{O}_n) \cap \text{EffRem}(\mathcal{O}_o, \mathcal{O}_p)$. From this we have an indicative fault in the set $\text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j)$, $\text{EffRem}(\mathcal{O}_o, \mathcal{O}_p)$ and two suggestive faults in the sets, $\text{EffAdd}(\mathcal{O}_i, \mathcal{O}_j)$, $\text{IneffRem}(\mathcal{O}_k, \mathcal{O}_l)$ and $\text{IneffRem}(\mathcal{O}_k, \mathcal{O}_l)$, $\text{IneffAdd}(\mathcal{O}_m, \mathcal{O}_n)$. The latter two seem to be subsumed by the encompassing former. The analysis presented here does not, at this time, cover all paired possibilities. This is partly due to the fact that some are impossible on their own (e.g., two additions or two removals in a row) and partly due to the fact that some are subsumed by others.

Of course, as we noted, all these observations only hold if the requirements have been stable over the examined period. If requirements fluctuate over a set of changes, then the changes might just track the requirements and the ontology might never be in a pathological state.

4 Methods and Materials

The verification of the concepts defined in Section 3 is carried out via a detailed analysis of the National Cancer Institute (NCI) Thesaurus (NCIt) ontology. The NCI is a U.S. government funded organisation for the research of causes, treatment, and prevention of cancer [5]. The NCIt is an ontology written in the Web Ontology Language (OWL) which supports the development and maintenance of a controlled vocabulary about cancer research. Reports on the collaboration process between the NCIt and its contributors have been published in 2005 and 2009 (see [6,7,8]), which provide a view of the procedural practices adopted to support domain experts and users in the introduction of new classes in the ontology. These publications, together with the publicly available monthly releases and class-based change logs, form the basis of the corpus used in this study.

We gathered 105 versions of the NCIt, from release 02.00 (October 2003) through to 12.08d (August 2012), from the NCI’s public archive.² Two versions are unparseable using the OWL API v3.4.1 [9], and were therefore discarded, leaving 103 versions. The ontologies were parsed and individual axioms and entities were extracted and inserted into a MySQL v5.1.63 database. All gathered data and SQL queries used in this study are available online.³ The database stores the following data for each NCIt release, O_i (where i is the version identifier):

1. Ontology O_i : Each ontology’s release name is stored in a table “Ontology” with a generated integer identifier i .
2. Axioms $\alpha_j \in O_i$: Each structurally distinct axiom α_j is stored in an “Axioms” table with identifier j , and a tuple (j, i) is stored in a table “Is In” (that is, axiom j is asserted in ontology i).
3. Effectual changes: Each added (removed) axiom $\alpha_j \in \text{EffAdd}(O_i, O_{i+1})$ ($\alpha_j \in \text{EffRem}(O_i, O_{i+1})$), with identifier j , is stored in table “Effectual Additions” (“Effectual Removals”) as a tuple $(j, i + 1)$.
4. Ineffectual changes: Each added (removed) axiom $\alpha_j \in \text{IneffAdd}(O_i, O_{i+1})$ ($\alpha_j \in \text{IneffRem}(O_i, O_{i+1})$), with identifier j , is stored in table “Ineffectual Additions” (“Ineffectual Removals”) as a tuple (j, i) .

All subsequent analysis is performed by means of SQL queries against this database, to determine suitable test areas for fault detection. For test area identification, we select test sets based on the outcome of 1) Frequency distribution analysis of the set of asserted axioms (i.e., in how many versions each axiom appears or follows from), and 2) Consecutivity analysis of asserted axioms (whether an axiom’s occurrence pattern has “gaps” throughout its “lifetime”). For fault detection, we conduct SQL driven data count analysis between the selected test cases and the effectual and ineffectual changes’ SQL tables, in order to categorise logical bugs as *FISoCs* or *FSSoCs*.

The machine used for data collection has an Intel Xeon Quad-Core 3.2GHz processor, 32GB of memory, and runs Java 7 on Mac OS X 10.8.3. The machine where

² ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Thesaurus/archive/

³ <http://owl.cs.manchester.ac.uk/research/topics/ncit/regression-analysis/>

the database is hosted, and where all SQL queries are executed has an Intel Dual-Core 3.0GHz processor, 8GB of memory, and runs Java 7 on Ubuntu Linux 11.10.

For the purpose of distinguishing effectual and ineffectual changes we use the Pellet description logic reasoner v2.3.0 [10].

5 Results

5.1 Test Area Selection

The test area selection for this study is determined by conducting analyses on axioms' frequency distribution and consecutivity evaluation. We start our frequency distribution analysis by determining in which (and consequently the number of) versions an axiom occurs in. From this we categorise their consecutivity based on the type of occurrence in the corpus, such as: continual occurrence, interrupted occurrence, and single occurrence. The analysis of axioms with continual occurrence provides knowledge about the stability of the ontology, since it helps with the identification of axioms that, due to their consistent presence throughout the ontology's versions, can be associated with the 'core' of the represented knowledge. As described in Section 3, the axioms' occurrence can be correlated with *FISoCs* or *FSSoCs* depending on the effect of the associated changes.

We found that the highest number of 20,520 asserted axioms correspond to frequency 11, as shown in Table 1 and Figure 1 (the former containing the most highly populated frequencies, while the latter showing all frequencies). This means that 20,520 axioms appear in the NCI ontology for exactly 11 versions. Of these asserted axioms, 20,453 asserted axioms (99.67%), appear in 11 consecutive versions. The distribution of these axioms across the corpus is concentrated between version 6 to 16 with 19,384 asserted axioms (the majority of these additions took place in version 6, with 13,715 added axioms), between versions 1 to 52 with 593 asserted axioms, and 187 asserted axioms for the remaining versions. These numbers do not account for the 358 new axioms added in version 93 that are still in the corpus in version 103 with 11 occurrences.

The next highest frequency is 5 with 14,586 asserted axioms, where 14,585 of those occur consecutively. Only the axiom `Extravasation \sqsubseteq BiologicalProcess` is present from version 20 to 23, removed in version 24, and re-added in version 45 before being removed in version 46.

The next two rows in Table 1 show the results for frequency distribution 2 and 3 with 13,680 and 12,806 asserted axioms respectively. For frequency distribution 2, there are 10,506 asserted axioms with consecutive occurrence. Of these axioms, 445 entered the corpus in version 102 and remain in the corpus until version 103. The total number of axioms with non-consecutive occurrences is 3,174 asserted axioms. However, only 8 axioms are not included in the set of axioms that are part of the modification event taking place between versions 93 and 94. In this event 3,166 axioms with non-consecutive occurrences were added in version 93, removed (or possibly refactored) in version 94, and re-entered the ontology in version 103. This editing event is discussed in Section 6. Of the 12,806 asserted axioms with frequency distribution 3, 12,804 asserted axioms occur in consecutive versions (99.98%) and 644 asserted axioms are present in the last studied version of the corpus.

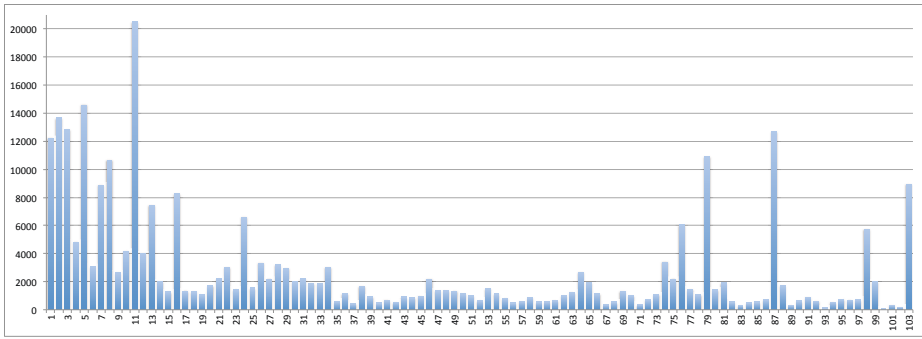


Fig. 1. Distribution of asserted axioms based on the number of versions they are present in (x-axis: frequency, y-axis: number of asserted axioms)

Our results show that three high frequency distributions are observed in the top ten distributions, with axioms occurring in 87, 79 and 103 versions. There are 12,689 asserted axioms present in 87 versions, with 99.86% of those occurring consecutively. From these axioms, 12,669 asserted axioms appear in the last version of the ontology, out of which 12,651 were added in version 17 and remained consecutively until version 103. For frequency distribution 79, there exist 10,910 asserted axioms that appear in 79 versions, with 10,866 still present in version 103. Out of these, 10,861 axioms were added in version 25 and remained until version 103 consecutively. Finally, there are 8,933 asserted axioms that appear in 103 versions of the NCIt. This means that 8,933 axioms were added in the first studied version of the NCIt and remain until the last studied version. That is, of the 132,784 asserted axioms present in version 103, 6.73% of the axioms were present since version 1. From this we can infer that 6.73% of the asserted axiom population found in the last version of the NCIt represents a stable backbone of axioms present in all versions.

As shown in Table 1, 12,219 asserted axioms occur in only 1 version of the NCIt. Out of these, 2,084 axioms appear in version 103 and may remain in future versions.

Table 1. Frequency distribution trends

Frequency	Axiom Count	Occurring in Version 103	Consecutive Occurrence	Non-consecutive Occurrence
11	20,520	358	99.67%	0.33%
5	14,586	831	99.99%	0.01%
2	13,680	445	76.80%	23.20%
3	12,806	664	99.98%	0.02%
87	12,689	12,669	99.86%	0.14%
1	12,219	47 in v102 and 2,084 in v103	–	–
79	10,910	10,866	99.93%	0.07%
8	10,662	599	99.93%	0.07%
103	8,933	8,933	100.00%	0.00%

Taking this into account, we observe that a total of 10,135 asserted axioms with single occurrence appear in the remaining 102 versions. From the 103 studied versions, 98 versions have asserted axioms that only appear in those versions; and versions 45, 54, 88, 92, and 100 do not. A detailed representation of this distribution across the NCIt corpus demonstrates that the first three years of the studied NCIt versions show the highest rate of single occurrences in the corpus with three identifiable high periods of single occurrences around versions 1 to 5, versions 16 to 18, and versions 21 to 25.

5.2 Fault Detection Analysis

In this study, we limit Fault Detection Analysis to the finite set of asserted axioms with non-consecutive occurrence for the top ten frequencies identified in the previous section. It is important to note that at this point this study does not examine the set of all *FISoC* and *FSSoC* for the collected versions of NCIt. Instead we focus our attention on the identified 53 asserted axioms that occur in non-consecutive versions for the top ten distributions, excluding all axioms that were part of the renaming events identified between versions 91 to 103 of the NCIt. Of these 53 examined axioms, 32 asserted axioms have logical bugs of type *FISoC*. Further examination of the change sets of these *FISoCs* indicate that 27 axioms conform directly with Definition 3 because all of their additions and removals are effectual; that is, the set of changes is $(\text{EffAdd}(O_i, O_{i+1}) \cap \text{EffRem}(O_j, O_{j+1}))$. The remaining 5 axioms have change sets of type $(\text{EffAdd}(O_i, O_{i+1}) \cap \text{IneffRem}(O_j, O_{j+1}) \cap \text{EffRem}(O_k, O_{k+1}))$. Although in this set there is an ineffectual removal prior to the effectual removal, from this change set we may conclude that ineffectual removal is “fixed” when the effectual removal takes place.

We also identified the asserted axiom

`Benign_Peritoneal_Neoplasm ⊆ Disease_Has_Primary_Anatomic_Site`
 only `Peritoneum` with *axiom_id* 159025 as a logical bug of type *FISoC* for the first removal $(\text{EffAdd}(O_{20}, O_{21}) \cap \text{EffRem}(O_{21}, O_{22}))$, and a second logical bug type *FISSoC* for the second removal $(\text{EffAdd}(O_{26}, O_{27}) \cap \text{IneffRem}(O_{28}, O_{29}))$. The presence of both logical bugs, *FISoC* and *FISSoC*, in this axiom suggests that the re-introduction of the axiom to the ontology in version 27 after being removed in version 22 may correspond to content regression, and the second ineffectual removal in version 29 to refactoring.

The remaining 21 asserted axioms have logical bugs of type *FSSoC*. Seventeen of these axioms conform with *FISSoC* set, thus suggesting possible refactoring. To confirm these refactoring events, additional axiomatic difference analysis needs to be carried out on these axioms, as suggested in [4]. Four axioms (*axiom_ids* 110594, 153578, 157661, and 127241) have the change sets identified for *F2SSoC*. Two of these axioms (*axiom_ids* 157661, and 127241) suggest refactoring for the first change set (the set is of type *FISSoC*), and are later re-introduced in the ontology with logical bugs of type *FISoC*.

As mentioned earlier, the analysis conducted in this section excludes fault detection for the set of axioms affected by the renaming event that took place between versions 91 and 103. We provide more information about this renaming event and the impact to our results in Section 6. However, it is important to mentioned that our analysis is

Table 2. *FISoC* - Fault indicating set of changes (Effectual Addition abbrv. to “Eff. Add.,” Effectual Removal abbrv. to “Eff. Re.,” Ineffectual Addition abbrv. to “Ineff. Add.,” and Ineffectual Removal abbrv. to “Ineff. Re.”)

Frequency Rate	Axiom ID	Versions for <Eff. Add., Eff. Re.>	Versions for <Eff. Add.>	Versions for <Eff. Add., Ineff. Re., Eff. Re.>	Versions for <Ineff. Add.>	First NCI Version	Last NCI Version
11	57506	<4,5>, <7,17>				4	16
	58364	<4,5>, <7,17>				4	16
	103206			<7,17,26>		7	25
	105069			<7,17,26>		7	25
	210295	<40,47>, <51,55>				40	54
2	49544	<2,3>, <4,5>				2	4
	50602	<2,3>, <4,5>				2	4
	50858	<2,3>, <18,19>				2	18
	120551	<12,13>, <16,17>				12	16
	172613	<25,26>, <62,63>				25	62
	172917	<25,26>, <62,63>				25	62
3	159025	<21,22>				21	28
	257839	<83,84>, <93,94>	<103>			83	103
87	30433	<1,12>, <14,75>	<89>			1	103
	39267	<1,2>	<18>			1	103
	68617	<5,6>	<18>			1	103
	118516	<12,74>	<79>			12	103
	119326	<12,74>	<79>			12	103
	121919	<13,47>	<51>			13	103
	122832	<13,47>	<51>			13	103
79	6838			<1,17,86>	<23>	1	85
	8905	<1,6>	<30>			1	103
	44135			<1,17,86>	<23>	1	85
	125718	<15,19>	<29>			15	103
	125895	<15,19>	<29>			15	103
	162303	<23,93>, <94,103>				23	103
	162304	<23,34>	<34>			23	103
8	22465			<1,2,52>	<45>	1	51
	67505	<5,6>, <10,17>				5	16
	238416	<72,79>	<103>			72	103
	238488	<72,79>	<103>			72	103
	262226	<87,93>, <94,96>				87	95

sensitive to cosmetic changes to axioms, e.g., axiom renaming, and does not treat them as logical bugs due to the superfluous nature of these changes.

6 Discussion

In general, the historical analysis of the NCI, as recorded in their monthly releases from 2003 to 2012, show that the ontology is consistently active and the evolution management process in place for NCI’s maintenance (as described in [11] and [7]) may be positive contributors to the overall steady growth of the NCI ontology.

The growth of the ontology is mostly driven by the asserted ontology where high levels of editing activity took place in the first three years of the analysed population. The change dynamics observed in this period suggest a trial and error phase, where editing and modelling activities are taking place until reaching a level of stability, possibly related to reaching maturity, for the remainder of the observed versions.

Table 3. FSSoC - Fault suggesting set of changes (Effectual Addition abbrv. to “Eff. Add.,” Effectual Removal abbrv. to “Eff. Re.,” Ineffectual Addition abbrv. to “Ineff. Add.,” and Ineffectual Removal abbrv. to “Ineff. Re.”)

Frequency Rate	Axiom ID	Versions for <Eff. Add., Ineff. Re.>	Versions for <Ineff. Add., Ineff. Re.>	Versions for <Ineff. Add.>	Versions for <Ineff. Add., Eff. Re.>	First NCIt Version	Last NCIt Version	Refactoring
11	110594		<10, 20>, <31, 32>			10	31	F2SSoC
	215592	<50, 55>		<98>		50	103	Refactoring
	215897	<50, 55>		<98>		50	103	Refactoring
5	157661	<20, 24>	<45, 46>			20	45	F2SSoC
2	99659	<6, 7>			<16, 17>	6	16	Refactoring
	127241	<16, 17>	<21, 22>			16	21	F2SSoC
3	159025	<27, 29>				21	28	Refactoring
87	3241	<1, 7>		<23>		1	103	Refactoring
	12085	<1, 17>		<33>		1	103	Refactoring
	106537	<9, 17>		<25>		9	103	Refactoring
	106569	<9, 17>		<25>		9	103	Refactoring
	106878	<9, 17>		<25>		9	103	Refactoring
	107407	<9, 17>		<25>		9	103	Refactoring
	107860	<9, 17>		<25>		9	103	Refactoring
	107952	<9, 17>		<25>		9	103	Refactoring
	108468	<9, 17>		<25>		9	103	Refactoring
	111380	<10, 17>		<24>		10	103	Refactoring
114579	<10, 17>		<24>		10	103	Refactoring	
79	42533	<1, 17>		<41>		1	103	Refactoring
8	153578		<17, 18>, <20, 27>			17	26	F2SSoC
	215709	<50, 53>		<99>		50	103	Refactoring

Although the chronological analysis primarily points to the first three years as a phase of rapid change, a more in-depth study of the diachronic data set revealed that content regression takes place throughout all versions of the NCIt. A detailed study of the ‘life of axioms’ in the ontology from the Frequency Distribution Analysis shows that the evolution of the NCIt is marked by logical bugs of either *FISoC* and/or *FSSoC* types. As a result, we found that asserted axioms with logical bugs enter the ontology in a version, are removed in a different version, and later re-entered the ontology unchanged. Only 6.73% of the asserted axioms in version 103 correspond to axioms that have been present and unchanged from the first version analysed until this last version.

Our study revealed that most asserted axioms appear in two versions of the ontology. However, in this finding we identified 125,294 axioms are affected by the renaming event that took place between versions 93 and 94. In a preliminary study conducted for this paper, we found these asserted axioms first appear in version 93, are removed in version 94, and then re-enter the NCIt unchanged in version 103. We have confirmed with the NCI that this editing event corresponds to the renaming of terms that took place in version 93, where every term name was replaced from its natural language name to its NCIt code. This renaming event also affects the set of asserted axioms with frequency distribution 11. The non-consecutive version occurrences for 1,186 axioms show that they first occur consecutively from versions 91 and 92, are removed in version 93, and then re-enter the ontology in version 94. These axioms remain consecutively until version 102 before they are removed again in version 103. The identification of this renaming event does not affect the information content dynamics of the ontology; however, it does affect the overall change dynamics. This renaming event is important to our analysis because it shows major editing periods are still part of the NCIt.

Taking into account these renaming events, the study found that the NCI overall ‘survival’ rate for asserted axioms is 5 versions. Axioms with non-consecutive presence in the ontology are directly linked to logical bugs that either indicate content regressions or suggest axiom refactoring. Information content is not as permanent as the managerial and maintenance processes indicate, but logical bugs for unmodified axioms are more predominant than expected. The analysis conducted in this paper identifies specific sets of axioms that are part of this group of regression cycles, and it is able to provide in detail the type of faulty editing patterns for these axioms and the location of these errors. We argue that the identification axioms with re-occurring logical bugs is a crucial step towards the identification of test cases and test areas that can be used systematically in Ontology Regression Testing.

7 Limitations

This study has taken under consideration the following limitations: (i) The NCI evolution analysis and asserted axiom dynamics correspond to the publicly available OWL versions of the NCI from release 02.00 (October 2003) to 12.08d (August 2012). Historical records of NCI prior to OWL are not taken into consideration in this study. (ii) The presented results and analysis is limited in scope to the set of asserted axioms only. The inclusion of entailment analysis is only conducted in regards to the computation of logical differences to categorise the asserted axioms’ regression events into logical bugs of types *FISoC* or *FSSoC*. (iii) Test area selection for the set of axioms with presence in non-consecutive versions is derived by selecting all axioms with non-consecutive presence based on their ranking in the high frequency analysis for all asserted axioms. The selected test area should be viewed as a snapshot of the whole population of axioms with non-consecutive presence, since the set of 53 analysed axioms correspond only to the top 10 high frequency distribution as described in Section 5.1. In addition, for the 53 analysed axioms we conducted consecutive versions pair analysis for effectual and ineffectual changes. Analysis of the whole corpus, including non-consecutive versions analysis for effectual and ineffectual changes, is planned for future research. (iv) This study primarily corresponds to Functional Requirement Test Impact Analysis since it deals directly with the ontology. Non-functional Requirements are linked to entailment analysis such as subsumption hierarchy study, which is excluded in this work.

8 Conclusion

Large collaborative ontologies such as the NCI need robust change analysis in conjunction with maintenance processes in order to continue to effectively support the ontology. The work presented in this paper shows that a detailed study of axioms with logical bugs need to be part of ontology evaluation and evolution analysis techniques due to its significant contribution to regression testing in ontologies. Although the study presented here is limited in that it is only evaluating unchanged asserted axioms, it still shows that a great portion of the editing efforts taking place in the NCI is in the unmodified content. Regression analysis of this unmodified content can target specific changes in the

modelling and representation approaches which can potential safe effort and increase productivity in the maintenance of the ontology.

Regression testing in Ontology Engineering is still a growing area of research, and the work presented here shows that a step towards achieving regression analysis in ontologies is by providing quantitative measurements of axiom change dynamics, identification of logical bugs, and the study of ontology evolutionary trends, all of which can be extracted efficiently by looking at versions of an ontology.

References

1. Gonçalves, R.S., Parsia, B., Sattler, U.: Analysing the evolution of the NCI thesaurus. In: Proc. of CBMS 2011 (2011)
2. Gonçalves, R.S., Parsia, B., Sattler, U.: Analysing multiple versions of an ontology: A study of the NCI Thesaurus. In: Proc. of DL 2011 (2011)
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. *J. of Web Semantics* (2008)
4. Gonçalves, R.S., Parsia, B., Sattler, U.: Categorising logical differences between OWL ontologies. In: Proc. of CIKM 2011 (2011)
5. de Coronado, S., Haber, M.W., Sioutos, N., Tuttle, M.S., Wright, L.W.: NCI Thesaurus: Using science-based terminology to integrate cancer research results. *Studies in Health Technology and Informatics* 107(1) (2004)
6. Hartel, F.W., de Coronado, S., Dionne, R., Fragoso, G., Golbeck, J.: Modeling a description logic vocabulary for cancer research. *J. of Biomedical Informatics* 38(2), 114–129 (2005)
7. Thomas, N.: NCI Thesaurus - Apelon TDE Editing Procedures and Style Guide. National Cancer Institute (2007)
8. Noy, N.F., de Coronado, S., Solbrig, H., Fragoso, G., Hartel, F.W., Musen, M.A.: Representing the NCI Thesaurus in OWL: Modeling tools help modeling languages. *Applied Ontology* 3(3), 173–190 (2008)
9. Horridge, M., Bechhofer, S.: The OWL API: A Java API for working with OWL 2 ontologies. In: Proc. of OWLED 2009 (2009)
10. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *J. of Web Semantics* 5(2), 51–53 (2007)
11. de Coronado, S., Wright, L.W., Fragoso, G., Haber, M.W., Hahn-Dantona, E.A., Hartel, F.W., Quan, S.L., Safran, T., Thomas, N., Whiteman, L.: The NCI Thesaurus quality assurance life cycle. *Journal of Biomedical Informatics* 42(3) (2009)

Optique: Towards OBDA Systems for Industry

Evgeny Kharlamov¹, Ernesto Jiménez-Ruiz¹, Dmitriy Zheleznyakov¹,
Dimitris Bilidas⁶, Martin Giese², Peter Haase⁵, Ian Horrocks¹, Herald Kllapi⁶,
Manolis Koubarakis⁶, Özgür Özçep⁴, Mariano Rodríguez-Muro³,
Riccardo Rosati⁷, Michael Schmidt⁵, Rudolf Schlatte²,
Ahmet Soylu², and Arild Waaler²

¹ University of Oxford, UK

`fname.lname@cs.ox.ac.uk`

² University of Oslo, Norway

`{martingi,rudi,ahmets,arild}@ifi.uio.no`

³ Free University of Bozen-Bolzano, Italy

`rodriguez@inf.unibz.it`

⁴ Hamburg University of Technology, Germany

`oezguer.oezcep@tu-harburg.de`

⁵ fluid Operations AG, Walldorf, Germany

`fname.lame@fluidops.com`

⁶ University of Athens, Greece

`{d.bilidas,herald,koubarak}@di.uoa.gr`

⁷ Sapienza Università di Roma, Italy

`lname@dis.uniroma1.it`

Abstract. The recently started EU FP7-funded project Optique will develop an end-to-end OBDA system providing scalable end-user access to industrial Big Data stores. This paper presents an initial architectural specification for the Optique system along with the individual system components.

Keywords: OBDA, ontologies, OWL 2, Big Data, System Architecture.

1 Introduction

A typical problem that end-users face when dealing with Big Data is that of data access, which arises due to the three dimensions of Big Data: *volume*, since massive amounts of data have been accumulated over the decades, *velocity*, since the amounts may be rapidly increasing, and *variety*, since the data are spread over different formats. In this context accessing the *relevant* information is an increasingly difficult task. The Optique project¹ [1] on ‘Scalable End-user Access to Big Data’ advocates a next generation of the well known *Ontology-Based Data Access* (OBDA) approach to address the Big Data dimensions and in particular the data access problem.

The project is focused around two demanding use cases that provide it with motivation, guidance, and realistic evaluation settings. The first use case is provided by Siemens (<http://www.siemens.com>) and encompasses several terabytes of temporal data coming from sensors, with a growth rate of about 30

¹ <http://www.optique-project.eu/>

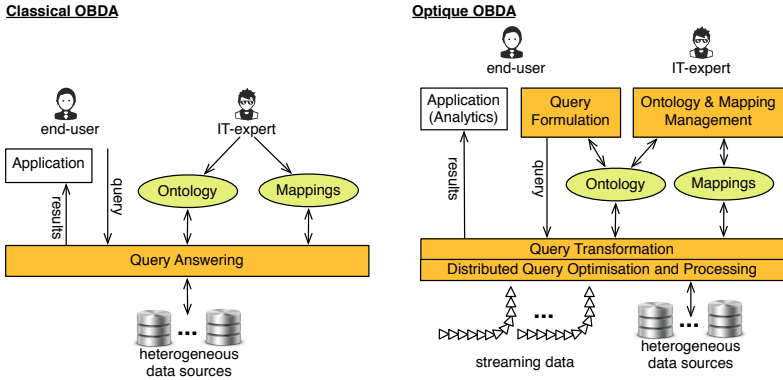


Fig. 1. Left: classical OBDA approach. Right: the Optique OBDA system

gigabytes per day. Users need to query this data in combination with many gigabytes of other relational data that describe events. The second use case is provided by Statoil (<http://www.statoil.com>) and concerns more than one petabyte of geological data. The data are stored in multiple databases which have different schemata, and—for the time being—the users have to manually combine information from many databases, including temporal, in order to get the results for a single query.

Accessing the *relevant* data in this context is becoming increasingly difficult for end-users. E.g., in large enterprises, such as Statoil, end-users work with applications that allow accessing data through a limited set of predefined queries. If an end-user needs data that these predefined queries do not provide, then the help of IT-experts (e.g., database managers) is required to translate the information need of end-users to specialised queries and optimise them for efficient execution. This process is the main bottleneck in the Optique use cases since it may require several iterations and may take several days. In particular in the oil and gas industry, IT-experts spend 30–70% of their time gathering and assessing the quality of data [2]. This is clearly very expensive in terms of both time and money.

The Optique has the potential of reducing the cost of data access dramatically, by automating the process of going from an information requirement to the retrieval of the relevant data, and to reduce the time needed for this process from days to hours, or even to minutes. A bigger goal of the project is to provide a platform with a generic architecture that can be adapted to any domain that requires scalable data access and efficient query execution.

The key to this automated translation is the “Ontology-Based Data Access” (OBDA) [3,4] approach. The idea is to use an ontology, which presents to users a semantically rich conceptual model of the problem domain. The users formulate their information requirements (that is, queries) in terms of the ontology, and then receive the answers in the same intelligible form. These requests should be executed over the data automatically, without an IT-expert’s intervention. To this end, a set of mappings is maintained which describes the relationship between the terms in the ontology and the corresponding terminology in the

data source specifications, e.g., table and column names in relational database schemas.

State-of-the-art OBDA systems that are based on classical OBDA architecture (see Figure 1), however, have shown among others the following limitations.

1. The *usability* of OBDA systems regarding the user interface is still an open issue. Even if the vocabulary provided by the ontology is familiar to end-users, the user may find difficult to formulate complex queries when several concepts and relationships are involved.
2. OBDA systems critically depend on a *suitable ontology* and the corresponding *set of mappings*, which are in practice expensive to obtain. Even if we assume that the ontology and mappings are given, they are not static artefacts and should evolve according to the new end-users' information requirements.
3. Treatment of query answering is usually limited to query rewriting and there is little support of *distributed* query optimisation and processing in OBDA systems. Moreover, even in the scenarios without data distribution current state of the art implementations of rewriting-based query answering have shown serious limitations in *scalability* [3].
4. *Temporal* data, *streaming*, e.g., sensor, data, and corresponding *analytical tools* are generally ignored by OBDA systems, which seriously limits their applicability in enterprises such as Siemens where one has to deal with large amounts of streaming data from many turbines and diagnostic centres, in combination with historical, that is, temporal relational data sources.

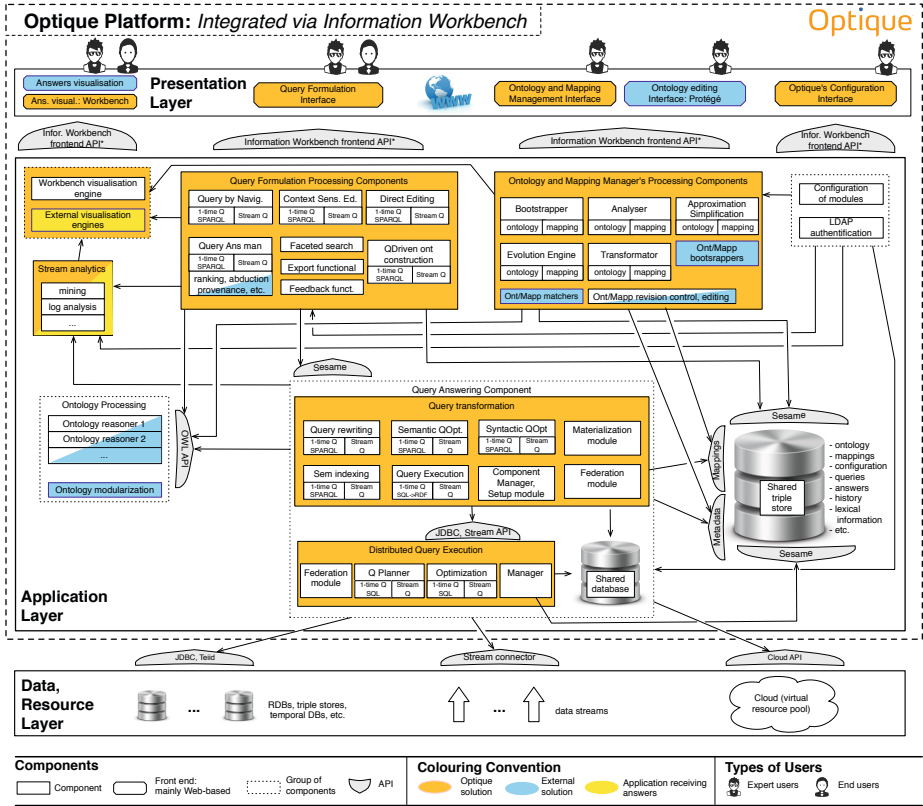
In the Optique project, we aim at developing a next generation OBDA system (cf. Figure 1, right) that overcomes these limitations. More precisely, the project aims at a cost-effective approach that requires to revise existing OBDA components and develop new ones, in particular, to support *(i)* novel ontology and mapping management components, *(ii)* user-friendly query formulation interface(s), *(iii)* automated query translation, *(iv)* distributed query optimisation and execution exploiting private and public cloud resources for scale-out, and *(v)* rigorous treatment of temporal and streaming data.

In this work we present an initial specification of the architecture of the Optique system describing the individual system components, their interplay and interfaces, and establishes agreement on system-wide conventions and standards. This architecture will serve as a guideline for the modules and components developed in the technical work packages, and will evolve according to new requirements. The final architecture remains for future work.

2 Optique Architecture

This section presents the designed Optique OBDA solution [5] which aims at overcoming the limitations of current OBDA systems. Figure 2 shows an overview of the Optique's solution architecture and its components. The architecture is developed using the three-tier approach and has three layers:

- The *presentation layer* consists of four main user interfaces, mainly Web based: *(i)* to configure and log-in into the system, *(ii)* to compose queries,



* E.g., widget development, Java, REST

Fig. 2. The general architecture of the Optique OBDA system

(iii) to visualise answers to queries, and (iv) to maintain the system by managing ontologies and mappings. The first three interfaces are for both end-users and IT-experts, while the fourth one is meant for IT-experts only.

- The *application layer* consists of several main components of the Optique's system, supports its machinery, and provides the following functionality: (i) *query formulation*, (ii) *ontology and mapping management*, (iii) *query answering*, and (iv) *processing and analytics of streaming and temporal data*. Additionally, the Optique system will include an *LDAP authentication* module, to assign different roles to Optique users, and a *Configuration module*, to provide a custom initial set-up to the Optique components.
- The *data and resource layer* consists of the data sources that the system provides access to, i.e., relational, semistructured, temporal databases and data streams. It also includes capabilities to exploit virtual resources such as storage and compute infrastructure available through cloud offerings.

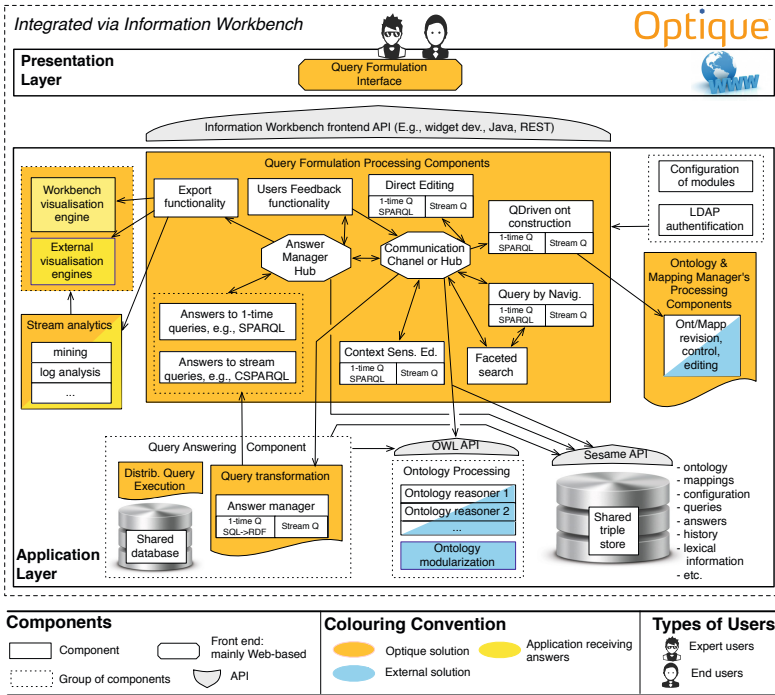


Fig. 3. Query Formulation components of the Optique OBDA system

The entire Optique system will be integrated via the Information Workbench (IWB) platform² [6]. The IWB is a generic platform for semantic data management, which provides a *shared triple store* for managing the OBDA system assets (such as ontologies, mappings, query logs, (excerpts of) query answers, database metadata, etc.), generic interfaces and APIs for semantic data management (e.g. *ontology processing APIs*). Moreover, the IWB provides general functionality for user management, fine-grained access control and configuration file management.

In addition to these backend data management capabilities, the Information Workbench comes with a flexible user interface that will be used for implementing the query formulation components. The user interface follows a semantic wiki approach, based on a rich, extensible pool of widgets for visualization, interaction, mashup, and collaboration, which can be flexibly integrated into semantic wiki pages, allowing developers to compose comprehensive, actionable user interfaces without any programming efforts.

In the following sections we describe in detail the main four application layer components together with their respective sub-architectures.

² www.fluidops.com/information-workbench/

2.1 Query Formulation Component

The query formulation component aims at providing a user-friendly interface for non-technical users combining multiple representation paradigms [7]. We intend to design and implement novel techniques to exploit the semantics of the underlying ontology in order to formulate both complex and valid queries, as well as to use already existing techniques, e.g., [8,9]. Furthermore, this component will also integrate a query-driven ontology extension subcomponent to insert new end-users' information requirements in the ontology. Figure 3 shows the main query formulation subcomponents for the Optique OBDA solution and their interaction with other components of the system. Next we give a brief overview of each of them. Note that many subcomponents deal with both one-time queries, e.g., SPARQL queries, and continuous queries, e.g., CSPARQL queries.

1. *Editing components.* Different users may cooperate on the same query or set of queries, thus, the Optique solution aims at providing (at least) three kinds of interfaces to formulate the query: *direct editing*, *context sensitive editing*, and *query by navigation* exploiting *faceted search* and other navigation paradigms. IT-experts may prefer the direct editing of the query using a formal language (e.g. SPARQL), while other users will be provided with a less technical interface, e.g., query by navigation. Additionally, direct editing should also allow the possibility of exploiting the ontology, and provide context sensitive completion. All interfaces should provide views on the partially constructed query, and users should be able to switch between views at will.
2. *Query-driven ontology construction component.* The ontology may not include all vocabulary needed by end-users. Moreover, the vocabulary is to a certain extent specific to individuals, projects, departments, etc., and subject to change. We consider it crucial to keep the ontology up-to-date w.r.t. end-user needs. Thus, the Query-driven ontology construction component will deal with four different changing scenarios driven by end-user requirements:
 - (a) Adding new synonyms. Concept synonyms (e.g. annotation labels) do not represent new logical extension of the ontology, and hence end-users will be able to add them to the ontology with no (logical) harm. For example, the concept `WellBore` can be extended with the labels “drill hole” or “borehole”. To avoid an overloading of the ontology with synonyms, we advocate a separation between the ontology (e.g. logical axioms) and the terminological information (e.g. synonyms, descriptions, related terms, etc.) as proposed in [10].
 - (b) Adding basic extensions. End-user queries may also require basic extension of the ontology hierarchy, such as adding a new concept, say, `GeologicalWellBore`, under `WellBore` (i.e. `GeologicalWellBore` \sqsubseteq `WellBore`). These types of additions can be considered *safe* [11] since they represent a *conservative extension* of the ontology. However, other additions to the ontology may require further analysis by the IT-expert if they are not conservative extensions (e.g. reclassifying the concept `WellBore` under the new concept `PlannedSideTrack`).
 - (c) “On the fly” extensions. This represents the more challenging scenario where we intend to exploit ontology learning techniques in order to

mine formulated queries and identify new relevant concepts and relations (e.g., [12,13]). Ontology alignment techniques (e.g. LogMap [14]) will also be required in order to relate the new vocabulary to the existing ontology concepts.

- (d) IT-expert assistance. In the cases where the manual or on-the-fly extensions are insufficient, the assistance of the IT-expert will be required to extend the ontology accordingly.
- 3. *The Answer Manager component.* This component should deal with the (basic) visualization of the query results and their transformation into the required output formats (e.g. input formats of external tools).
- 4. *The User Feedback component.* This component is intended to allow the user to semi-automatically refine a query if the (partially) obtained results are not the expected ones. Furthermore, similar or related queries to the partially constructed query will also be suggested in order to help end-users in the refinement.

This component interacts with other components of the Optique OBDA system:

- 1. *The Ontology Revision Control system.* Different versions of the ontology may exist concurrently (e.g. extensions driven by different formulated queries or query requirements). These versions will be managed by the IT-experts through a revision control system (from the Ontology and Mapping management system) in order to detect logical defects (e.g. unsatisfiabilities), logical conflicts among versions as in [15], and OWL 2 profile violations (e.g. a new version is outside the OWL 2 QL profile).
- 2. *The Ontology Processing component.* The ontology will be a key element for the query formulation component and thus, the ontology processing component (e.g. OWL API) will also play an important role. Furthermore, logic-based ontology modularization techniques [16] will also be exploited to achieve a good balance between overview and focus when dealing with large ontologies. The properties of such modules guarantee that the semantics of the concepts of interest is preserved while providing (in general) a much smaller fragment of the ontology.
- 3. *Shared triple store.* The Query formulation component accesses the Shared triple store where, among others, the ontology, query logs, (excerpts of) query answers and the lexical information are physically stored.
- 4. *The Query Answering component* will transform the formulated queries into executable and optimized queries with respect to the data sources (e.g. streaming data, relational databases).
- 5. *Stream Analytics.* By exporting answers to this component one can do, e.g., data mining in answers for continuous queries.
- 6. The *Visualisation Engines* allow to visualise both queries and query answers.

2.2 Ontology and Mapping Management Component

The ontology and mapping management component [17] will provide tools and methodologies to (i) semi-automatically bootstrap an initial ontology and mappings and (ii) maintain the consistency between the evolving mappings and the evolving ontology.

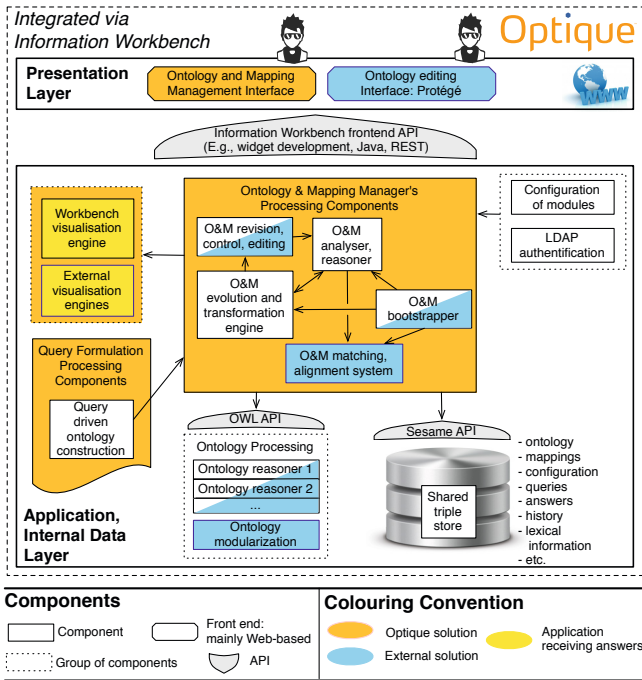


Fig. 4. Ontology and Mapping Management component of the Optique OBDA system

In Figure 4 we present the *Ontology and Mapping Management* component (the O&M manager) of the Optique OBDA system. The O&M manager has a Web interface at the presentation layer which will also include the well known ontology editor Protégé for sophisticated extensions over the ontology. Functionalities of the O&M manager are intended for IT-experts rather than end-users. The manager includes five subcomponents:

1. The *O&M Bootstrapper*. OBDA systems crucially depend on the existence of suitable ontologies and mappings. Developing them from scratch is likely to be expensive and a practical OBDA system should support a (semi-)automatic bootstrapping of an initial ontology and set of mappings. Thus, the Optique system will be equipped with a *bootstrapper*, a routine that takes a set of database schemata and possibly instances over these schemata as an input, and returns an ontology and a set of mappings connecting the ontology entities to the elements of the input schemata.
2. The *O&M Matching and alignment system*. The bootstrapped ontologies will be aligned with domain ontologies using state of the art ontology alignment techniques³ (e.g. LogMap [14]).
3. The *O&M Analyser and reasoner*. The ontologies and the mappings are not static objects and are subject to frequent changes. The O&M analyser will check ontologies and mappings for defects caused by these changes.

³ <http://www.ontologymatching.org/>

We distinguish between logical and modelling defects. The three types of *logical* defects that are usually discussed in the literature (see, for example, [18,19]) are *inconsistency*, *unsatisfiability*, and *incoherency*. In OBDA scenarios *empty mappings* will also be an important logical defect. A mapping of an OBDA setting is *empty* if it does not propagate any individuals (resp. pairs of individuals) into any concept (resp. property) in the ontology. *Modelling defects* are less intuitive and less formally defined than logical ones. Typical modelling defects are *redundancy* [20] (e.g. redundant axioms, concepts, and mappings) and *unexpected entailments* (e.g. those entailments that do not correspond to the expected by domain experts [21]).

4. The *O&M Evolution and transformation engine*. The functionality of this component is twofold. On the one hand, it performs debugging on defects found by the analyser, and returns a debugged version of the ontology and the mappings. In the development of the Optique OBDA system we plan to exploit existing techniques from ontology evolution, e.g. [22,23]. On the other hand, it may perform several kinds of transformations of the ontology and the mappings: for instance, it should transform an input ontology which is in a language not supported by the OBDA systems (e.g., OWL 2) and return a version of the ontology into the supported language (e.g., OWL 2 QL profile). Also, it may perform transformations of the mappings related to formal properties of the mappings or to optimisation strategies [24]. This may involve changes in the syntax and/or semantics of the ontology.
5. The *O&M Revision, control and editing system* will support versioning and editorial processes for both ontologies and mappings. It will also act as a hub, coordinating interoperation between the analyser and evolution engine.

The O&M manager interacts with other components of the Optique system:

1. It accesses the *Shared triple store*, where the ontology and mappings are stored. It both reads and updates the ontology and mappings.
2. The analyser, alignment system, and evolution engine access to reasoning capabilities, e.g., ontology reasoners, ontology modularisation engines, etc.
3. The *Query Formulation Component* can call the O&M manager when a user decides to extend the ontology. We refer to this as *query-driven ontology construction*.
4. The O&M manager is connected to a *Visualisation engine* to visualise ontology and mappings.

2.3 Query Answering Component

Covering the backend functionalities (query planning, optimization, execution, and use of cloud resources), the query answering component is compound of two large subcomponents: (i) *query transformation* and (ii) *distributed query processing* components. The query transformation component is responsible for translating, also known as *rewriting*, queries received from the query formulation component, e.g., SPARQL queries, into an optimised executable code that is evaluated over the data sources and streams in the data layer. The Quest system [25] implements sophisticated rewriting and optimization techniques and is going to be the core part of the Optique's query transformation.

The distributed query processing subcomponent provides query planning and execution. It distributes queries to individual servers and uses massively

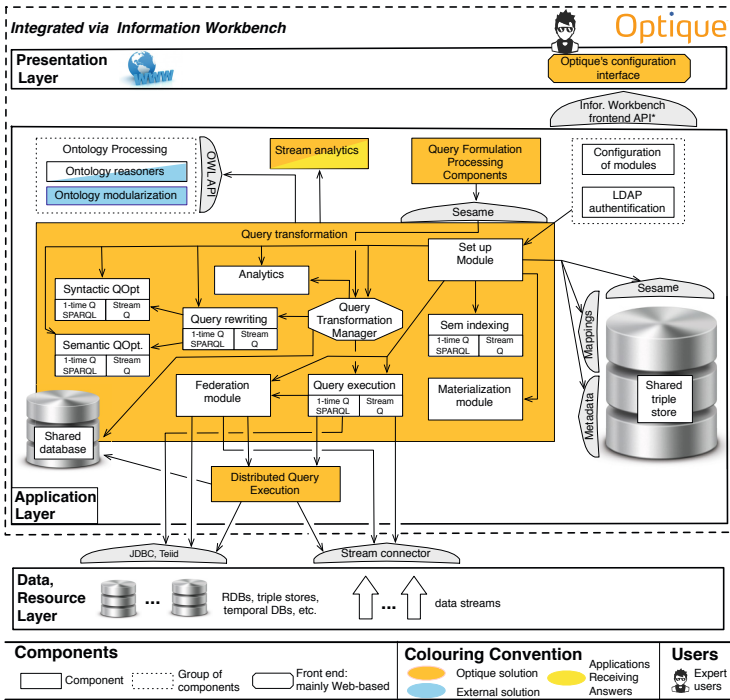


Fig. 5. Query transformation component of the Optique OBDA system

parallelised (cloud) computing. The ADP [26] for complex dataflow processing in the cloud is going to be the core part of Optique’s distributed query processing.

Query Transformation Component. Figure 5 presents the architecture of the *Query transformation* (QT) component of the Optique system [27]. The QT component interacts with other components of the Optique OBDA system:

1. The *query formulation component* provides a query interface for end-users. This component receives queries from an end-user and send them to the QT component, e.g., via Sesame API.
2. The *configuration* module provides the configuration for the QT module that is required for query transformation performance.
3. The *ontology processing* module (a group of components such as ontology reasoners, modularisation systems, etc.) is called by the QT module to perform semantic optimisation.
4. The *distributed query processing* component receives rewritten queries from the QT module and performs their evaluation over data sources.
5. The (*internal*) *shared database* contains the technical data required for data answering process such as semantic indices, answers to queries, etc. This database will be only shared by the QT component and the distributed query processing component.

6. The *shared triple store* contains the data that can be used by (the most of) the components of the Optique OBDA system. E.g., it contains the ontology, the mappings, the query history, etc.
7. The *stream analytics module* analyzes answers to the stream queries.
8. *Data sources* (RDBs, triple stores, data streams) can be also accessed by the QT module during the query execution.

The Query Transformation Manager (QTM) is the principal component of the QT component and will orchestrate the QT process. The QT process is triggered when a query is received from the query formulation component. Next we describe the role of each QT subcomponent and their interplay in the query transformation process. The process can be divided into several stages:

1. *Initialisation.* At this stage the *Configuration* module sends the configuration to the *Set-up module*, which in turn configure the other QT submodules. The initialisation includes several steps in which the input ontology and mappings get analysed and optimised so as to allow the rewriting and optimisation algorithms to be fast, and the query evaluation over the data sources to be more efficient. The *semantic indexing* and *materialisation* modules are in charge of creation and maintenance of so-called semantic index.
2. *Query rewriting.* The QTM sends the (SPARQL) query Q received from the query formulation component to the *query rewriting* module, which is in charge of rewriting the query in the required format; for example, SQL if it is supposed to be evaluated over RDBs or Streaming SPARQL for querying data streams. Further, for the sake of simplicity, we will assume that the target query format is SQL. Along with the rewriting, this module optimises the rewritten query both *syntactically* and *semantically* as described below:
 - During the transformation process, the initial query (e.g., SPARQL) might be turned into a number of SQL queries Q_1, \dots, Q_n such that their union is equal to Q . In the Syntactic and Semantic optimisation stages, these queries get optimised to improve the performance, e.g., some joints, conditions, filters within this SQL queries are deleted. In the former stage, the methods that are used to detect what parts of the queries have to be optimised are syntactical, that is, they are based only on the shape of a query and do not involve any reasoning. In the latter stage, the methods take into account semantic information such as query containment, integrity constraints of the data sources, etc.
3. *Query execution.* After rewriting and optimization, the queries $Q'_{i_1}, \dots, Q'_{i_m}$ are sent to the *query execution* module. This module decides which of these queries, if any, need to be evaluated using distributing query execution, and which can be evaluated by the standard query answering facilities. In the former case, the corresponding queries are sent to the *distributed query processing* component. In the latter case, the corresponding queries are evaluated directly over the *data sources* by standard means. If needed, some of the queries can be evaluated over a federated database system, by the *Federation module*.

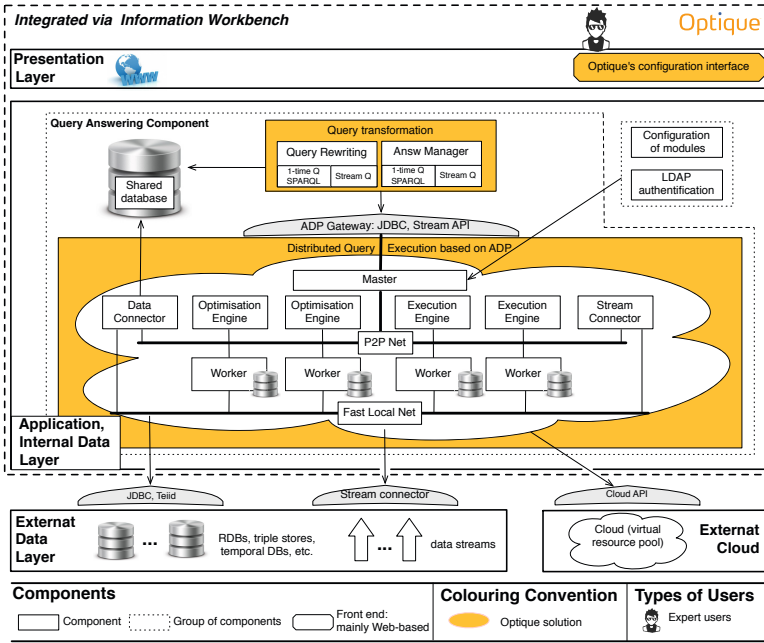


Fig. 6. General architecture of the ADP component within the Optique System

4. *Query answer management.* After the query evaluation process, the answers to the queries that have been sent directly to the data sources are returned to the QTM module. The manager transforms them into the required format and sends them back to the query formulation component, which represents the answers to end-users. The queries that has been sent to the *distributed query processing* component do not necessarily go directly to the query answer manager, but to a *shared database*. The reason is that the answers can be up to several GBs, so sending them directly to the QTM may hang the system. The QTM receives the signal that the answers are in the shared database and metadata about the answers. Then, together with the *analytics module*, it decides how to proceed further. The answers to one-time queries, e.g. SQL queries over RDBs, eventually go to the query formulation component, while the answers to stream queries go to the *stream analytics module*.

Distributed Query Processing Component. The Distributed query processing component [28] aims at achieving the efficiency in Big Data scenarios by both *massive parallelism*, i.e., running queries with the maximum amount of parallelism at each stage of execution, and *elasticity*, i.e., by allowing a flexibility to execute the same query with the use of resources that depends on the the resource availability for this particular query, and the execution time goals. The distributed query execution is based on the ADP (Athena Distributed Processing) [26] a system for complex dataflow processing in the cloud. ADP has been

developed and used successfully in several European projects such as Health-e-Child [29].

The general architecture of the distributed query processing component within the Optique platform is shown in Figure 6. The query is received through the gateway using JDBC. This communication mainly involves interaction with the QT component. The *Master* node is responsible for initialization and coordination of the process. The *optimization engine* produces the execution plan for the query. Next, the execution plan is given to the *execution engine* which is responsible for reserving the necessary resources, sending the operators of the graph to the appropriate *workers* and monitor the execution. Next we describe in more detail the distribution process:

- *Language and Optimization*: A query Q , expressed in SQL, are issued to the system through the gateway. Q is transformed to a data flow language allowing complex graphs with operators as nodes and with edges representing producer-consumer relationships. The first level of optimization is planning, which results in an SQL query script. We enhanced SQL by adding the table partition as a first class citizen of the language; it is defined as a set of tuples having a particular property (e.g., the tuples in the same partition share the value of a hash function applied on one column). A table is defined as a set of partitions. The optimizer produces an execution plan in the form of a directed acyclic graph with all the information needed to execute Q .
- *Execution Engine*: ADP relies on an asynchronous execution engine. As soon as a worker node completes one job, it is sending a corresponding signal to the execution engine. The execution engine uses an asynchronous event based execution manager, which records the jobs that have been executed and assigns new jobs when all the prerequisite jobs have finished.
- *Worker Pool*: The resources needed to execute Q (machines, network, etc.) are allocated automatically. Those resources are wrapped into containers, which are used to abstract from the details of a physical machine in a cluster or a virtual machine in a cloud. Workers run Q using a python wrapper of SQLite (<http://www.sqlite.org>). This part of the system can also be used as a standalone single node DB (<https://code.google.com/p/madis/>). Queries are expressed in a declarative language which is an extension of SQL. This language facilitates considerably the use of user-defined functions (UDFs). The system supports row, aggregate, and virtual table functions.
- *Data/Stream Connector*: The Data and Stream Connectors (DC and SC) are responsible for handling and dispatching the relational and stream data through the network respectively. They are used when the system receives a request for collecting the results of executed queries. SC uses an asynchronous stream event listener to be notified of incoming stream data, whereas DC utilizes a table transfer scheduler to receive partitions of relational tables from the worker nodes.

2.4 Streaming and Temporal Data

Processing and Analytics of Streaming and Temporal Data is primarily motivated by the need of large industries. For example, Siemens encompasses several

terabytes of temporal data coming from sensors, with an increase rate of about 30 gigabytes per day [30]. Addressing this challenge requires a number of techniques and tools which should be integrated in several modules of the Optique OBDA solution. For example, the query formulation module should support window queries and the query answering module should support rewriting and optimised execution of such queries. Additionally, the ontology and mapping management component is also required to design appropriate formalisms to support ontological modelling of streaming and temporal data.

The query language that the system should provide to end-users should combine (i) *temporal operators*, that address the time dimension of data and allow to retrieve data which was true “always” in the past or “sometimes” in the last X months, etc., (ii) *time series analysis operators*, such as mean, variance, confidence intervals, standard deviation, as well as trends, regression, correlation, etc., and (iii) *stream oriented operators*, such as sliding windows.

Given a query, mapping languages, and ontology, the Optique system should be able to translate queries into highly optimised executable code over the underlying temporal and streaming data. This requires techniques for automated query translation continuous and temporal queries. Existing translation techniques are limited and they do not address query optimisation and distributed query processing. Thus, novel approaches should be developed.

3 Conclusions

The Optique system will provide an end-to-end OBDA solution for Big Data access addressing a number of important industry requirements. The technology and system will be developed in a close cooperation of six universities, two industrial partners, and two use cases: Statoil and Siemens. The system will be deployed and evaluated in our use cases. It will provide valuable insights for the application of semantic technologies to Big Data integration problems in industry.

Acknowledgements. We are thankful to D. Calvanese, T. Hubauer, M. Meier, R. Möller, M. Roshchin, and D. Fabio Savo for insightful discussions. The authors are supported by the EU project Optique (FP7-IP-318338).

References

1. Giese, M., Calvanese, D., Haase, P., Horrocks, I., Ioannidis, Y., Kllapi, H., Koubarakis, M., Lenzerini, M., Möller, R., Özçep, O., Rodriguez Muro, M., Rosati, R., Schlatte, R., Schmidt, M., Soylyu, A., Waaler, A.: Scalable End-user Access to Big Data. In: Akerkar, R. (ed.) Big Data Computing. Chapman and Hall/CRC, Florida (to appear, 2013)
2. Crompton, J.: Keynote talk, the W3C Workshop on Semantic Web in Oil & Gas Industry, Houston, TX, USA, December 9-10 (2008), <http://www.w3.org/2008/12/ogws-slides/Crompton.pdf>
3. Rodriguez-Muro, M., Calvanese, D.: High Performance Query Answering over DL-Lite Ontologies. In: Knowledge Representation and Reasoning, KR (2012)

4. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. *Semantic Web* 2(1), 43–53 (2011)
5. Calvanese, D., Giese, M., Haase, P., Horrocks, I., Hubauer, T., Ioannidis, Y., Jiménez-Ruiz, E., Kharlamov, E., Kllapi, H., Klüwer, J., Koubarakis, M., Lamparter, S., Möller, R., Neuenstadt, C., Nordtveit, T., Özcep, O., Rodriguez-Muro, M., Roshchin, M., Ruzzi, M., Savo, F., Schmidt, M., Soyly, A., Waaler, A., Zheleznyakov, D.: The Optique Project: Towards OBDA Systems for Industry. In: OWLED (2013)
6. Haase, P., Hütter, C., Schmidt, M., Schwarte, A.: The Information Workbench as a Self-Service Platform for Linked Data Applications. In: Proc. of WWW (2012)
7. Cuenca Grau, B., Giese, M., Horrocks, I., Hubauer, T., Jiménez-Ruiz, E., Kharlamov, E., Schmidt, M., Soyly, A., Zheleznyakov, D.: Towards Query Formulation and Query-Driven Ontology Extensions in OBDA. In: OWLED (2013)
8. Bechhofer, S., Horrocks, I.: Driving User Interfaces from FaCT. In: Proceedings of the 2000 International Workshop on Description Logics, pp. 45–54 (2000)
9. Catarci, T., Dongilli, P., Mascio, T.D., Franconi, E., Santucci, G., Tessaris, S.: An ontology based visual tool for query formulation support. In: European Conf. on Artif. Intell. (ECAI), pp. 308–312 (2004)
10. Jimeno-Yepes, A., Jiménez-Ruiz, E., Llavori, R.B., Rebholz-Schuhmann, D.: Reuse of terminological resources for efficient ontological engineering in life sciences. *BMC Bioinformatics* 10(S-10) (2009)
11. Jiménez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., Berlanga, R.: Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 185–199. Springer, Heidelberg (2008)
12. Zhang, J., Xiong, M., Yu, Y.: Mining query log to assist ontology learning from relational database. In: Zhou, X., Li, J., Shen, H.T., Kitsuregawa, M., Zhang, Y. (eds.) APWeb 2006. LNCS, vol. 3841, pp. 437–448. Springer, Heidelberg (2006)
13. Kotis, K., Papasalouros, A., Maragoudakis, M.: Mining query-logs towards learning useful kick-off ontologies: an incentive to semantic web content creation. *IJKEDM* 1(4) (2011)
14. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 273–288. Springer, Heidelberg (2011)
15. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Supporting concurrent ontology development: Framework, algorithms and tool. *Data Knowl. Eng.* 70(1), 146–164 (2011)
16. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res.* 31, 273–318 (2008)
17. Haase, P., Horrocks, I., Hovland, D., Hubauer, T., Jiménez-Ruiz, E., Kharlamov, E., Klüwer, J., Pinkel, C., Rosati, R., Santarelli, V., Soyly, A., Zheleznyakov, D.: Optique System: Towards Ontology and Mapping Management in OBDA Solutions. In: WoDOOM (2013)
18. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.A.: Debugging unsatisfiable classes in OWL ontologies. *J. Web Sem.* 3(4), 268–293 (2005)
19. Shchekotykhin, K., Friedrich, G., Fleiss, P., Rodler, P.: Interactive Ontology Debugging: Two Query Strategies for Efficient Fault Localization. *Web Semantics: Science, Services and Agents on the World Wide* 12 (2012)

20. Grimm, S., Wissmann, J.: Elimination of Redundancy in Ontologies. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part I*. LNCS, vol. 6643, pp. 260–274. Springer, Heidelberg (2011)
21. Horrocks, I.: Tool Support for Ontology Engineering. In: *Foundations for the Web of Information and Services*, pp. 103–112 (2011)
22. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of *DL – Lite* Knowledge Bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I*. LNCS, vol. 6496, pp. 112–128. Springer, Heidelberg (2010)
23. Cuenca Grau, B., Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D.: Ontology Evolution Under Semantic Constraints. In: *Knowledge Representation and Reasoning, KR* (2012)
24. Pinto, F.D., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: *Int’l Conference on Extending Database Technology (EDBT)*, pp. 561–572 (2013)
25. Rodríguez-Muro, M., Calvanese, D.: Quest, an OWL 2 QL Reasoner for Ontology-based Data Access. In: *OWLED* (2012)
26. Kllapi, H., Sitaridi, E., Tsangaris, M.M., Ioannidis, Y.E.: Schedule optimization for data processing flows on the cloud. In: *Proc. of SIGMOD*, pp. 289–300 (2011)
27. Calvanese, D., Horrocks, I., Jiménez-Ruiz, E., Kharlamov, E., Meier, M., Rodríguez-Muro, M., Zheleznyakov, D.: On Rewriting and Answering Queries in OBDA Systems for Big Data. In: *OWLED* (2013)
28. Kllapi, H., Bilidas, D., Horrocks, I., Ioannidis, Y., Jiménez-Ruiz, E., Kharlamov, E., Koubarakis, M., Zheleznyakov, D.: Distributed Query Processing on the Cloud: the Optique Point of View. In: *OWLED* (2013)
29. Health-e-Child: Integrated healthcare platform for european paediatrics (2006), <http://www.health-e-child.org/>
30. Horrocks, I., Hubauer, T., Jiménez-Ruiz, E., Kharlamov, E., Koubarakis, M., Möller, R., Bereta, K., Neuenstadt, C., Özçep, Ö., Roshchin, M., Smeros, P., Zheleznyakov, D.: Addressing Streaming and Historical Data in OBDA Systems: Optique’s Approach. In: *Know@LOD* (2013)

Summary of the Demonstration and Poster Track

Miriam Fernández¹ and Vanessa López²

¹ Knowledge Media Institute
The Open University, UK
² IBM Dublin

The demonstration and poster track is an opportunity for researchers and practitioners to present their innovative prototypes, practical developments, on-going projects, lessons learned and late-breaking results. This year we had a very exciting track with thirty-five poster and thirty-two demo submissions. All poster and demonstration papers were peer reviewed by at least 2 reviewers, resulting in twenty accepted posters and twenty-four accepted demos.

Posters and demos give a glimpse to where the Semantic Web is heading. This year they spanned a broad set of topics over which we can highlight the topic of semantic search: linked data indexing, exploratory search over linked data, spoken search over linked data, cross-lingual querying over linked data, federated SPARQL queries and visual query editors. Other less frequent topics presented this year are relation discovery over linked data, facts discovery over linked data, ontologies for the academic domain, ontologies for the music domain, solutions for big data and semantic analyses of the social web.

All posters and demos were presented and voted on by the ESWC audience. The best ESWC 2013 poster was awarded to Tobias Käfer and colleagues for their work “Exploring the dynamics of Linked Data”. In this work the authors proposed the Linked Data Observatory, 8 months of weekly collected linked data documents used to study the dynamics and evolution of linked data. The best ESWC 2013 demo award was given to two different works: “Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data” by Charalampos Nikolaou et al. as well as “Exploratory search on the top of DBpedia chapters with the discovery hub application” by Nicolas Marie et al. The first demo is a web-based tool that enables exploration of linked geospatial data and supports the creation, sharing, and collaborative editing of thematic maps. The second demo is an exploratory search engine that helps users to explore topics of interest.

All posters and demonstrations are included in this volume. Demonstration papers are up to 5 pages long, poster papers are shorter with 2 pages maximum.

We hope you enjoy each and one of them!

LODatio: A Schema-Based Retrieval System for Linked Open Data at Web-Scale

Thomas Gottron¹, Ansgar Scherp^{2,1}, Bastian Kraye¹, and Arne Peters¹

¹ Institute for Web Science and Technologies, University of Koblenz, Germany

² Data and Web Science, University of Mannheim, Germany

{gottron, scherp, bastiankramer, arne}@uni-koblenz.de

Abstract The Linked Open Data (LOD) cloud has grown to an enormous source for semantic data. Its distributed and decentralized approach is one reason for its success but also poses challenges. A main difficulty is to identify those data sources on the LOD cloud which provide the information a user is actually interested in. With LODatio, we have developed a prototype retrieval system to support users in finding the right data sources for a given schema-oriented information need. Beyond classical search system functions such as retrieval, ranking, result set size estimation and providing result snippets, LODatio provides sophisticated support for the users in refining and expanding their information need.

1 Introduction

The aim of Linked Open Data (LOD) [4] is to publish and interlink open data of different quality and of different purpose on the web using the Resource Description Framework (RDF). This data forms a huge web-scale RDF graph—the so-called LOD cloud—which represents an enormous source of structured information. Since its advent in 2007, LOD has gained widespread popularity and is supported by non-profit organizations such as libraries as well as major industries. It lies in the nature of the data to be distributed across several de-centralized data sources. As on the classical web of documents, there is no central instance managing or indexing all the information. With the growing number of data sources and the tremendous increase of pure amount of data published, it becomes more and more important to identify sources in the LOD cloud which provide information satisfying schema-oriented information needs. Schema-oriented information needs comprise the search for certain types of resources or resources having certain properties. Existing Semantic Web search engines such as Swoogle [2], SemSearch [7], Sig.ma [11], and Sindice [8] typically focus on the search for specific resources, though. They usually allow for a keyword-based querying approach of traditional search or to enter an URI to start browsing. Thus, schema-oriented information needs as indicated above are not supported by these systems.

In order to provide for a system addressing schema-oriented information needs, we have designed the retrieval system LODatio. LODatio allows for identifying those LOD data sources that provide RDF resources satisfying a schema-oriented information need specified via a SPARQL query. The advantage of using SPARQL as query language is that it allows for a precise definition of the schema-oriented constraints on the data

and that the same query can subsequently be used to query the relevant data sources. However, for future versions of LODatio we intend to implement an additional interface for non-expert users allowing for the formulation of keyword-based queries which will then be translated to SPARQL automatically [9]. LODatio uses a schema-based index [6] to identify for given SPARQL query patterns the relevant sources for LOD. The index contains meta data about these data sources such as their URI, the number of compliant resources and a limited number of examples. Making use of this meta data in an nifty way, we implemented several services to actively support the user in his information seeking task. These services entail relevance based ranking of data sources, estimation of the overall number of relevant resources, provision of example data in the form of result snippets, as well as user support for refining or generalizing the query. The latter two services support typical search tactics to overcome situations in which the user finds too few or too many results [1]. Inspired by web search engines, we refer to this services as *Did you mean?* and *Related queries*.

The motivation for all of these features are the positive effects similar features have had on classical web retrieval. Query related result snippets, for instance, allow users to interact faster and more successful with a retrieval system [10]. Query recommendations, instead, have been found to be accepted frequently and to help the users in overcoming problems in formulating their information need [5]. In general, such active user support features have been called for and proposed as a research agenda for retrieval systems [1].

The live demo of the LODatio prototype (cf. Figure 1) is available online at <http://lodatio.west.uni-koblenz.de:8080>. It is accessible via a web interface allowing for end user access to the system. Example queries provide an easy entrance point to the system and allow for a quick formulation of a first query.

2 Main Features of LODatio

Motivated by successful features of modern web search engines, we have implemented a range of services that form the backbone of the LODatio system:

Ranked Retrieval. Based on the lookup capabilities of the underlying index, we can identify relevant data sources. To provide a ranking, we use a naive ranking function which presents data sources in decreasing order of the number of complying resources they provide.

Result Snippets. For each relevant data source, we can provide example resources and their labels. This allows the user to get a first impression of the data he might encounter at a listed data source and, thereby, supports him in the decision whether or not the data is relevant to his information need.

Result Set Size Estimation. For a given SPARQL query, we can estimate the total amount of matching resources on the entire LOD cloud. This is achieved by aggregating the number of complying resources over all relevant data sources. In addition to supporting the end users in judging the amount of suitable data, the result set estimation plays an important role in the next two services.

Did You Mean? For queries that provide only few or even no results, the user might want to generalize his information need. LODatio supports this task by taking the original SPARQL query and iteratively removing or relaxing its constraints provided in the

The screenshot displays the LODatio web interface. At the top left is the 'lodatio' logo. The main area contains a SPARQL query box with the following text:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT ?x
WHERE {
  ?x rdf:type dbpedia:Writer .
  ?x <http://xmlns.com/foaf/0.1/depiction> ?unknown .
}
```

Below the query box is a 'Send Query' button. Underneath, there is a 'Did you mean?' section with two suggestions:

- Remove: ?x foaf:depiction ?unknown (Try this query)
- Remove: ?x rdf:type dbpedia:Writer (Try this query)

The next section is '500+ datasources with 502+ instances', listing various URIs and their instance counts:

- http://dbpedia.org/data/Dave_Mirra_BMX_Challenge.xml (2 instances)
- http://dbpedia.org/data/Friedrich_D77rennmatl.xml (2 instances)
 - Friedrich Dürrenmatt
- <http://dbpedia.org/data/??inasl.xml> (1 instances)
 - ?inasl
- <http://dbpedia.org/data/??jpest.xml> (1 instances)
- http://dbpedia.org/data/??lvoro_de_Campos.xml (1 instances)
 - Álvoro de Campos
- http://dbpedia.org/data/??mer_Seyfettin.xml (1 instances)
 - Ömer Seyfettin
- http://dbpedia.org/data/??ngel_Cappelletti.xml (1 instances)
 - Ángel Cappelletti
- http://dbpedia.org/data/??ric-Emmanuel_Schmitt.xml (1 instances)
 - Éric-Emmanuel Schmitt
- http://dbpedia.org/data/??tefan_Octavian_Iosif.xml (1 instances)
 - ?tefan Octavian Iosif
- http://dbpedia.org/data/??udo_Ondrejov.xml (1 instances)
 - ?udo Ondrejov

At the bottom of this list is 'Page 1 of 50' and a 'Next Page' button. Below that is a 'Related Queries' section with three suggestions:

- Add: ?x foaf:name ?unknown0 (Try this query)
- Add: ?x foaf:page ?unknown0 (Try this query)
- Add: ?x dcterms:subject ?unknown0 (Try this query)

On the right side, there is a 'WeST' logo and a 'Examples:' section with several query snippets and 'Try It!' buttons:

- qb:Observation (Try It!)
- qb:Observation qb:dataSet ?any (Try It!)
- dbpedia:Actor (Try It!)
- More examples

Fig. 1. Web frontend of the LODatio prototype demo

form of query patterns. The resulting new, generalized query candidates are analysed for the estimated size of their result set. Finally, the user receives suggestions from this set of generalized query candidates that extend the current result set in a modest way.

Related Queries. As complement to the *Did you mean?* function, we provide a service which recommends more specific queries. Here, we make use of the schema index and look up query patterns to extend the original information need of the users. This allows for a directed extension of the constraints in a way that does not lead to empty result sets. The quality of the extension is again measured by the size of the obtained new result set.

3 The LODatio Prototype Demo

In LODatio, the users state their information need as an initial SPARQL query. This query is entered in the box at the top of the prototype as depicted in Figure 1. As we do not assume that users in general are capable or willing to write SPARQL queries, we are well aware that a future extension of LODatio is to add a feature for automatically mapping keyword-based input to a SPARQL query, e.g. following the approach of [9]. For the sake of allowing users an easy access to the LODatio system, we currently provide illustrative example queries at the right side of the query box.

User Interaction. After specifying their information need as SPARQL query, the users submit this initial query to LODatio by pressing the “Send Query” button. Subsequently, the different features of LODatio as described in Section 2 are used to fill the

result screen as depicted in the screenshot shown in Figure 1. The first three features of *ranked retrieval*, *result snippets*, and *result set size estimation* are used to provide the content of the middle section of the user interface presenting the result list of Linked Data sources relevant to the query. The ranked result list is paginated, as the size of the set of relevant data sources can easily become very large. The users can navigate through the pages by clicking on the “Next Page” button as shown in Figure 1 (the “Previous Page” button is not shown as the screenshot depicts the first result page). For each entry in the result list, the users can click on the URI of the data sources to directly inspect the information that is behind a specific data source. Furthermore, up to three example resources are used as illustrative *result snippets* for each relevant data source. Above of the page-based result list is an overview of the overall number of relevant data sources found as well as an estimation of the number of instances in these data sources.

Besides the navigable page-based result list, the users can also modify the query expressing their information need in order to either broaden or narrow down the result set. These features are shown in terms of *Did you mean?* and *Related Queries* query suggestions at the top and the bottom of the result list, respectively. The user can select one of these query suggestion by pressing the “Try this query” button next to the depiction of the query modification. In Figure 1, the *Did you mean?* query suggestions are to remove the property `foaf:depiction` or the RDF type `dbpediaowl:Writer` from the current query. Regarding the *Related Queries*, LODatio has retrieved from the underlying schema information the query suggestions to add the property `foaf:name`, `foaf:page` (for the homepage of a person), or `dcterms:subject`.

Implementation. We have implemented all of the above services and integrated them into the live LODatio prototype. As data background, we use a schema index computed on the data set provided for the Billion Triple Challenge 2012¹. The data set represents a 17GB crawl of a part of the LOD cloud and contains nearly 1.5 billion RDF triple. A real-time computation of the LODatio services can be achieved on a single core machine. As data structure, we use in LODatio a schema-based index over linked data [6]. In this index, we encode schema related information of LOD in a look up data structure. The elements in this schema index correspond to RDF resources that satisfy certain types of SPARQL query patterns. The schema can precisely answer queries addressing (i) resources of a given type or set of types, (ii) resources having certain properties, or even (iii) resources of certain type that are linked to other resources of a given type via a specific set of properties. While more complex queries cannot be answered exactly they can be answered in an approximative way by combining the available patterns (i) to (iii). Looking up the information stored in the index allows for a fast retrieval of meta data of the RDF resources complying with the specified query patterns. This payload is composed of the URIs of relevant data sources, example resources and their labels as well as count information of how many matching resources can be found at this data source. The meta data in the payload of the schema-based index is at the core of the implementation of all the features discussed above. The most essential information is certainly the count information as it feeds also into the *Did you mean?* and *Related Queries* features.

¹ <http://challenge.semanticweb.org/2012>

Technically, the index is implemented in a hybrid architecture [3]. We use a triple store for the schema level of the index and store the payload, instead, in a relational database. This allows for a compact representation of the flexible schema information while leveraging RDBMS technology for looking up meta data information following a consistent schema format.

4 Summary

Our advanced retrieval system LODatio for identifying relevant data sources on the LOD cloud leverages a schema index to respond to a user's information need specified as a SPARQL query. The system provides core retrieval functionality such as ranked result lists, result snippets, and estimates of the total result set size. Furthermore, it supports two typical tactics of users when seeking information: generalizing and refining the information need. As future work we plan to evaluate the benefits on the user's side in extensive user studies.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257859, ROBUST.

References

1. Bates, M.J.: Where should the person stop and the information search interface start? *Information Processing and Management* 26(5), 575–591 (1990)
2. Ding, L., Finin, T.W., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: *CIKM*. ACM (2004)
3. Gottron, T., Scherp, A., Kraye, B., Peters, A.: LODatio: Using a Schema-Based Index to Support Users in Finding Relevant Sources of Linked Data. In: *K-CAP 2013: Proceedings of the Conference on Knowledge Capture* (2013)
4. Heath, T., Bizer, C.: *Linked Data: Evolving the Web Into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool (2011)
5. Kelly, D., Cushing, A., Dostert, M., Niu, X., Gyllstrom, K.: Effects of popularity and quality on the usage of query suggestions during information search. In: *Conference on Human Factors in Computing Systems* (2010)
6. Konrath, M., Gottron, T., Staab, S., Scherp, A.: Schemex—efficient construction of a data catalogue by stream-based indexing of linked data. *Web Semantics: Science, Services and Agents on the World Wide Web* 16, 52–58 (2012)
7. Lei, Y., Uren, V.S., Motta, E.: Semsearch: A search engine for the semantic web. In: Staab, S., Svátek, V. (eds.) *EKAW 2006*. LNCS (LNAI), vol. 4248, pp. 238–245. Springer, Heidelberg (2006)
8. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. *IJMSO* 3(1), 37–52 (2008)
9. Shekarpour, S., Auer, S., Ngomo, A.C.N., Gerber, D., Hellmann, S., Stadler, C.: Keyword-driven sparql query generation leveraging background knowledge. In: *Web Intelligence*, pp. 203–210 (2011)
10. Tombros, A., Sanderson, M.: Advantages of query biased summaries in information retrieval. In: *SIGIR* 1998, pp. 2–10 (1998)
11. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: Live views on the web of data. *J. Web Sem.* 8(4), 355–364 (2010)

Payola: Collaborative Linked Data Analysis and Visualization Framework*

Jakub Klímek^{1,2}, Jiří Helmich¹, and Martin Nečaský¹

¹ Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
{klimek, necasky}@ksi.mff.cuni.cz

² University of Economics, Prague
Nám. W. Churchilla 4, 130 67 Praha 3, Czech Republic

Abstract. Payola is a framework for Linked Data analysis and visualization. The goal of the project is to provide end users with a tool enabling them to analyze Linked Data in a user-friendly way and without knowledge of SPARQL query language. This goal can be achieved by populating the framework with variety of domain-specific analysis and visualization plugins. The plugins can be shared and reused among the users as well as the created analyses. The analyses can be executed using the tool and the results can be visualized using a variety of visualization plugins. The visualizations can be further customized according to ontologies used in the resulting data. The framework is highly extensible and uses modern technologies such as HTML5 and Scala. In this paper we show two use cases, one general and one from the domain of public procurement.

Keywords: Linked Data, RDF, visualization, analysis.

1 Introduction

As the Linked Data begins to gain interest from general public in addition to semantic web developers, there is a growing need for user friendly linked data tools that do not require deep knowledge of semantic web technologies. The example of such a user can be a journalist who is interested in the data published as Linked Data, but does not want to learn SPARQL and RDF specifics. The problem arising here is that for each data domain, the data needs to be visualized differently to be comprehensible to non-technical users familiar with that domain. Therefore, it is hard to come up with a general analysis and visualization tool that would be usable by experts who know RDF and SPARQL and at the same time could shield the end-user from the RDF triples in a useful manner. Our vision is to build a community around a framework that provides all that is necessary to access and visualize Linked Data. The community would consist of both expert developers and end-users. Developers would create, share and reuse both general and domain-specific plugins for analysis and visualization of Linked Data. End-users could then easily use and combine the plugins, give feedback and suggest improvements, all without extensive technical knowledge.

* The research is supported in part by the EU ICT FP7 under No.257943, LOD2 project and in part by the Technology Agency of the Czech Republic (TAČR) grant number TA02010182.

2 Framework and Plugins

In this section, we will briefly describe the framework and how the plugins fit together. Payola consists of a server part and a client part. The client part runs in an HTML5 enabled web browser using JavaScript and handles analyses editing and results visualization. The server part runs as a Scala program and handles users, datasource querying and data manipulation. Payola is a framework, so the concrete functionality and usability is the responsibility of plugins that populate the framework.

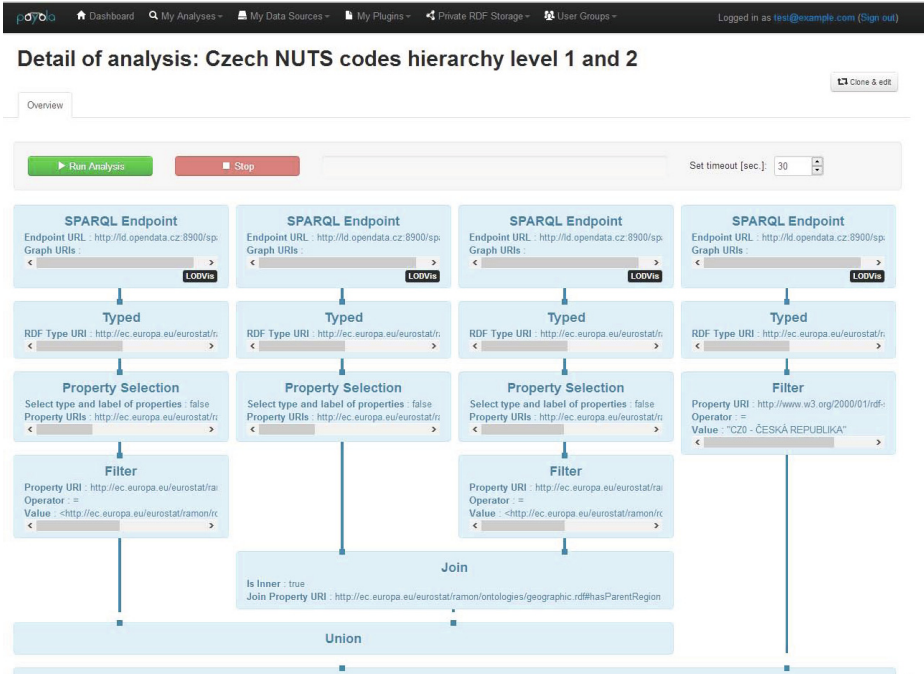


Fig. 1. Analysis editor - NUTS code hierarchy of the Czech Republic

There are two basic types of plugins. Firstly, the created analyses consists of interconnected analytical plugins - *analyzers*. When an analysis is executed and results are returned, they are visualized using visualization plugins - *visualizers*. As a proof-of-concept, we created a library of default analyzers that represent typical parts of SPARQL queries. These include selection of resources of a certain type, selection of properties that are interesting to the user and filters for resources which satisfy SPARQL filter conditions. However, more complex analyzers are possible. For example an ontological filter selects resources and properties which are instances of a given ontology. Currently under development is a set of domain-specific analyzers for the domain of public procurement. These analyzers are translated into SPARQL queries and executed on selected SPARQL endpoints. An issue to consider here is that when we connect analyzers arbitrarily, the resulting SPARQL queries can be inefficient. The framework now

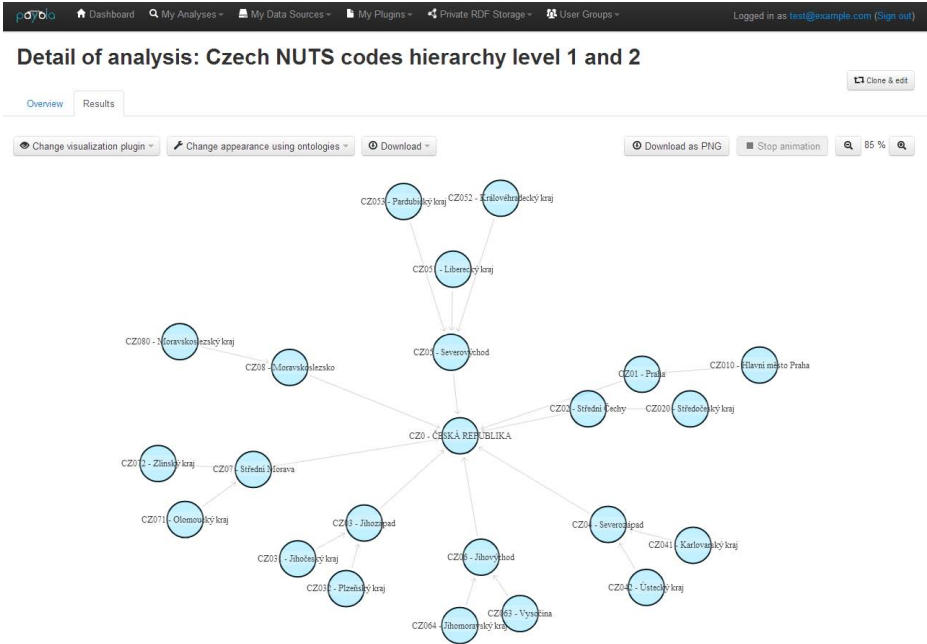


Fig. 2. Results browser - NUTS code hierarchy of the Czech Republic

includes a simple optimizer that combines many analyzers into one SPARQL query. However, it has its limits and it is a whole research area of query optimization which we need to focus on in order for the framework to be efficient even with more complicated analyses. An important feature for collaboration which is now under development is the ability to use a completed analysis as an analyzer (one box) in another analysis. So far a user can only clone an analysis and edit his copy. Once an analysis is correctly defined (see Figure 1 for an example of how an analysis can look like), it can be executed and the results visualized using different visualizers. We created a small proof-of-concept library of simple visualizers that include various graph and chart visualizations. An example can be seen in Figure 2 and Figure 3. While graph visualization is generically applicable for RDF data, for some data a chart visualization can be more suitable. The column chart visualizer has certain requirements on the input RDF graph. It has to contain entities of the same type, each equipped with a label and property with a value. When the analysis result graph data meets this criteria, it can be visualized as a column chart. Currently, the Payola framework is populated by proof-of-concept basic plugins which are not suited for the non-technical end-users yet. Currently under development is a feature for packaging an analysis an analyzer with user-friendly parameter selection. These packaged analyses are then directly usable by end-users.

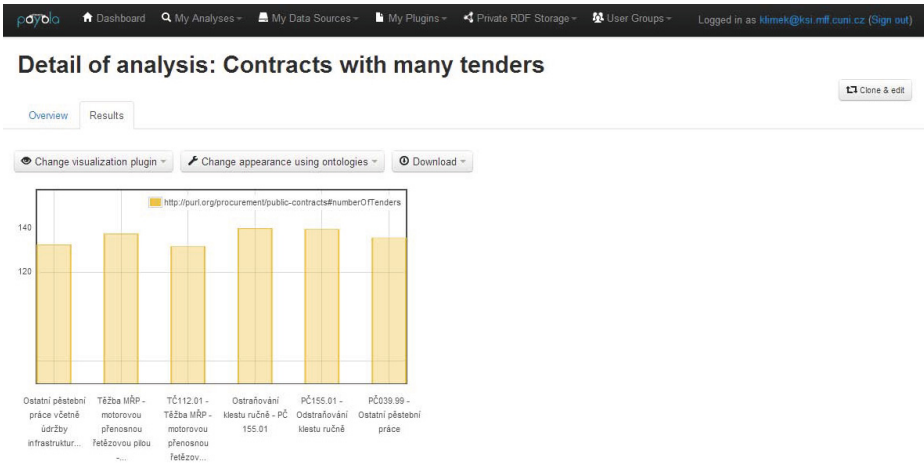


Fig. 3. Column chart visualization - numbers of tenders for each public contract

3 Use Cases

In this section, we will describe two use cases. The first goal is to reuse an analysis that shows a hierarchy of NUTS¹ codes for the Czech Republic, but for Germany. We will show how to adjust the existing analysis, which will show how easy it is to reuse analyses created by other users. The user finds the appropriate analysis in a list of available analyses. Then he creates his own copy (see Figure 1) and identifies the parameters that need to be changed. In future this step will be deprecated by creating analyzers out of existing analyses, which will be parametrized by selected values such as the country name. When the user changes the appropriate parameters, he executes the analysis and gets the results. Then the user can switch visualizers without the need to recompute the analysis.

In the second scenario, the user will see various visualizers applied to a result of a simple analysis from the domain of public procurement. It returns public contracts to which more than 120 tenders were submitted as a triple table. The analysis is labeled "Contracts with many tenders". The user can switch to, for example, the Gravity visualizer and see the results as a graph. Next, the general graph visualization can be customized for the Public Contracts Ontology² using the "Change appearance using ontologies". The user will see that the nodes representing public contracts are now green and look like a bobby pin. Because the result has a format required for the column chart visualizer (see Figure 3), the user can select it and see a column chart where the columns represent the number of tenders for each public contract in the result.

We asked our test users a couple of questions regarding usability and we got the responses we expected, which is that the general idea is great, but we need to work on the usability and easiness to use.

¹ Nomenclature of Territorial Units for Statistics - European region codes.

² <http://purl.org/procurement/public-contracts>

4 Related Work

There are analytical tools such as *Semantic Pipes* [3]. It is a tool for transformation of data on the web which also supports RDF. However, it is focused solely on the analysis part and it is not possible to create an analysis without semantic web expertise, nor does the tool support creation of plugins by experts that would bridge this gap. There is a number of Linked Data visualization tools available, such as *Tabulator* [2] or *Explorator* [1], which usually visualize Linked Data as a graph and also support table view of the RDF triples. *Sgvizler*³ allows web developers to render results of SPARQL queries as charts, maps, etc. These solely focus on the visualization part and offer a limited number of visualization techniques without support for extensibility. Also, there is *ViziQuer* [4] for exploring unknown SPARQL endpoints and getting an idea about what data is inside. None of the tools aim to support non expert users and none of them provide both analysis and visualization capabilities to a sufficient extent.

5 Conclusions

In this demo paper, we presented Payola, a framework for analysis and visualization of Linked Data which aims to combine the advantages found in some of the above mentioned tools to create an environment for analysis specification, execution and results visualization with a potential to be usable by non-expert users, for who the expert ones can create plugins, which are then easy to use. It supports analyses editing and collaboration via sharing and reuse. When an analysis is executed, results can be visualized using various plugins. The framework is currently populated by basic proof-of-concept plugins and more sophisticated plugins are currently under development. The tool is available at its project website: <http://live.payola.cz>.

References

1. Araujo, S., Shwabe, D., Barbosa, S.: Experimenting with Explorator: a Direct Manipulation Generic RDF Browser and Querying Tool. In: WS on Visual Interfaces to the Social and the Semantic Web, VISSW 2009 (2009)
2. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and analyzing linked data on the semantic web. In: 3rd Int. Semantic Web User Interaction WS (2006)
3. Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: Proceedings of the 18th international conference on World wide web, WWW 2009, pp. 581–590. ACM, New York (2009)
4. Zviedris, M., Barzdins, G.: ViziQuer: A Tool to Explore and Query SPARQL Endpoints. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 441–445. Springer, Heidelberg (2011)

³ <http://code.google.com/p/sgvizler/>

A System for Aligning Taxonomies and Debugging Taxonomies and Their Alignments

Valentina Ivanova and Patrick Lambrix

Department of Computer and Information Science and the Swedish e-Science Research Centre
Linköping University, 581 83 Linköping, Sweden

Abstract. With the increased use of ontologies in semantically-enabled applications, the issues of debugging and aligning ontologies have become increasingly important. The quality of the results of such applications is directly dependent on the quality of the ontologies and mappings between the ontologies they employ. A key step towards achieving high quality ontologies and mappings is discovering and resolving modeling defects, e.g., wrong or missing relations and mappings. In this demonstration paper we present a system for aligning taxonomies, the most used kind of ontologies, and debugging taxonomies and their alignments, where ontology alignment is treated as a special kind of debugging.

1 Motivation

To obtain high-quality results in semantically-enabled applications such as the ontology-based text mining and search applications, high-quality ontologies and alignments are both necessary. However, neither developing nor aligning ontologies are easy tasks, and as the ontologies grow in size, it is difficult to ensure the correctness and completeness of the structure of the ontologies. A key step towards high-quality ontologies and alignments is debugging the ontologies and alignments.

This demonstration paper is a companion paper to [1]. We discuss a system that supports a unified approach for debugging and alignment of taxonomies, where ontology alignment can be seen as a special kind of debugging. Although the system can be used as an ontology alignment system or an ontology debugging system, the integration of both leads to additional benefits for both and an overall improvement of the quality of the ontologies and the alignments [1]. The alignment algorithms generate/extend (available) alignments between the ontologies in use which are further employed during the debugging. The debugging algorithms debug and possibly extend/generate alignments. Thus the integration between both provides a unique opportunity to debug and align ontologies in a unified approach even when alignments are not available.

The system has been used in experiments with the ontologies and alignments from the Anatomy track of the Ontology Alignment Evaluation Initiative. In the main experiment in [1], an alignment was created with 1311 mappings. Further, 47 wrong is-a relations were removed from the ontologies and 341 missing is-a relations were added. The examples in this paper and the demonstration are from these experiments. To our knowledge there is no other system that integrates ontology debugging and ontology alignment in a uniform way and that allows for a strong interleaving of these tasks.

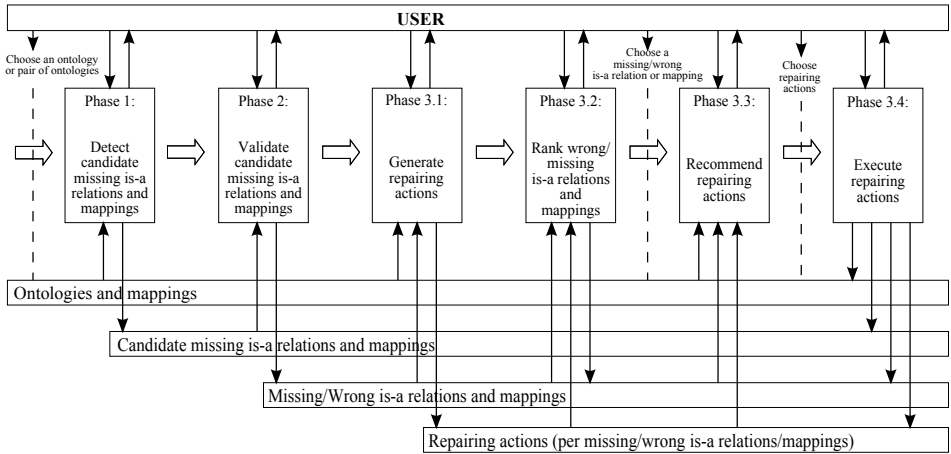


Fig. 1. Workflow [1]

In this paper we focus on showing the use of the system. For formal definitions, the underlying algorithms and experiment results, we refer to [1].

2 Implemented System

In our system RepOSE the user starts a debugging session by loading the ontologies and alignments (if available). The output is the set of repaired ontologies and alignments. In this section we describe the different phases in our debugging workflow (Figure 1) through the user interface of the system. We note that the phases can be interleaved.

Detecting and Validating Candidate Missing Is-a Relations and Mappings. In the detection phase (Phase 1) candidate missing is-a relations (CMIs) and candidate missing mappings (CMMs) are generated, which then need to be validated (Phase 2) by a user. This can be done in several ways. For the CMIs the user uses the tab 'Step1: Generate and Validate Candidate Missing is-a Relations' (Figure 2) and chooses an ontology for which the CMIs are computed. In this case the knowledge intrinsic in the network is used. An is-a relation is a CMI if it can be inferred via logical derivation from the network, but not from the ontology alone. The user can then validate all or some of the CMIs as well as switch to another ontology or another tab. The validation partitions the CMIs into *missing* and *wrong* is-a relations. CMIs are visualized in groups in order to show them in their context, while avoiding cluttering of the display. Initially, CMIs are shown using arrows labeled by '?' (as in Figure 2 for (*acetabulum*, *joint*)) which the user can toggle to 'W' for wrong relations and 'M' for missing relations. For each CMI the justifications (derivation paths) in the ontology network are shown as an extra aid for the user. For instance, in Figure 2 (*palatine bone*, *bone*) is selected and its justifications shown in the justifications panel. Concepts in different ontologies are presented with different background color. The domain expert can also ask for recommendations. When a user decides to finalize the validation of a group of CMIs, RepOSE checks for

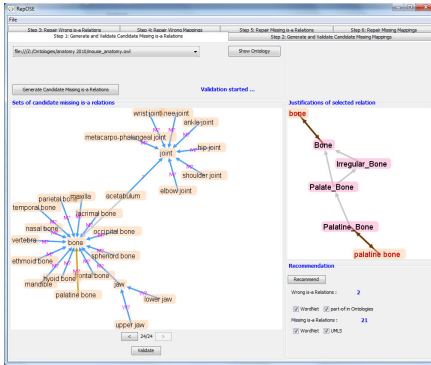


Fig. 2. Generating and validating CMI

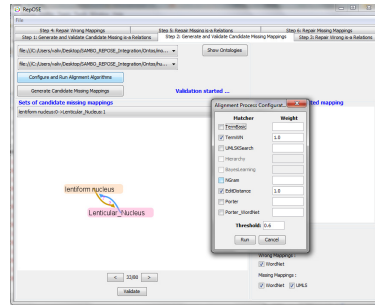


Fig. 3. Aligning

contradictions in the current validation as well as with previous decisions and if contradictions are found, the current validation will not be allowed and a message window is shown to the user.

For CMMs a similar tab 'Step 2: Generate and Validate Candidate Missing Mappings' can be used to choose a pair of ontologies and their alignment for which the CMMs are generated. There are two approaches to generate CMMs. The first approach uses logical derivation as for the CMIs. In this case a mapping is a CMM if it can be inferred via logical derivation from the network, but not from the two source ontologies and their alignment alone. The other approach uses ontology alignment algorithms. Clicking on the Configure and Run Alignment Algorithms button opens a configuration window (Figure 3) where the user can select the matchers (linguistic, WordNet-based and UMLS-based algorithms from the SAMBO system [3]), their weights and the threshold for the computation of the mapping suggestions. Clicking on the Run button starts the alignment process. The similarity values for all pairs of concepts belonging to the selected ontologies are computed, combined and filtered, and the resulting mapping suggestions are shown to the user for validation. The validation process continues in a similar way as for the CMIs. The validation partitions the CMMs into *missing* and *wrong* mappings. During the validation a label on the edge shows the origin of the CMMs - derived from the network, computed by the alignment process or both. The CMMs computed only by the alignment algorithms do not have justifications since they were not logically derived. The rest of the process is as described above.

Repairing Wrong Is-a Relations and Mappings. Figure 4 shows the RepOSE tab 'Step 3: Repair Wrong is-a Relations' for repairing wrong is-a relations (Phase 3). Clicking on the Generate Repairing Actions button, results in the computation of repairing actions for each wrong is-a relation of the ontology under repair. A wrong is-a relation can be repaired by removing at least one element in every justification. The wrong is-a relations are then ranked in ascending order according to the number of possible repairing actions and shown in a drop-down list. Then, the user can select a wrong is-a relation and repair it using an interactive display. The user can choose to repair all wrong is-a relations in groups or one by one. The display shows a directed graph representing the justifications. The nodes represent concepts. As

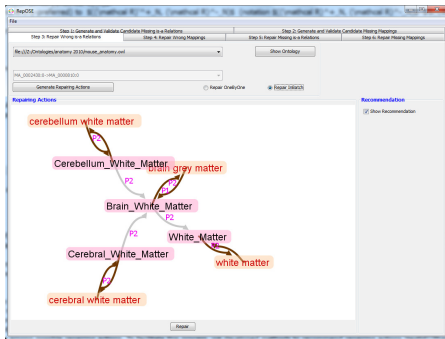


Fig. 4. Repairing wrong is-a relations

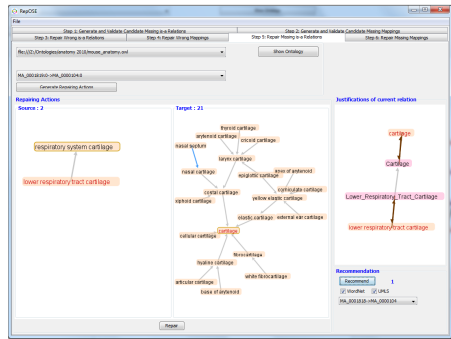


Fig. 5. Repairing missing is-a relations

mentioned before, concepts in different ontologies are presented with different background color. The concepts in the is-a relation under repair are shown in red. The edges represent is-a relations in the justifications. These is-a relations may be existing asserted is-a relations (shown in grey), mappings (in brown), unrepaired missing is-a relations (in blue) and the added repairing actions for the repaired missing is-a relations (in black).

In Figure 4 the user has chosen to repair several wrong is-a relations at the same time, i.e., (*brain grey matter*, *white matter*), (*cerebellum white matter*, *brain grey matter*), and (*cerebral white matter*, *brain grey matter*). In this example we can repair these wrong is-a relations by removing the mappings between *brain grey matter* and *Brain_White_Matter*. We note that, when removing these mappings, all these wrong is-relations will be repaired at the same time.

For the wrong is-a relations under repair, the user can choose, by clicking, multiple existing asserted is-a relations and mappings on the display as repairing actions and click the Repair button. RepOSE ensures that only existing asserted is-a relations and mappings are selectable, and when the user finalizes the repair decision, RepOSE ensures that the wrong is-a relations under repair and every selected is-a relation and mapping will not be derivable from the ontology network after the repairing. Further, all consequences of the repair are computed (such as changes in the repairing actions and changes in the lists of wrong and missing is-a relations and mappings).

A similar tab ('Step 4: Repair Wrong Mappings') is used for repairing wrong mappings. Only the wrong mappings derived by logical derivation need to be repaired. The wrong mappings that are computed by the alignment algorithms only are stored to avoid duplication of validation work, but do not lead to a debugging opportunity.

Repairing Missing Is-a Relations and Mappings. Figure 5 shows the RepOSE tab 'Step 5: Repair Missing is-a Relations' for repairing missing is-a relations (Phase 3). Clicking on the Generate Repairing Actions button, results in the computation of repairing actions for the missing is-a relations of the ontology under repair. A missing is-a relation can be repaired by adding is-a relations to the ontology. It was shown in [2] that repairing missing is-a relations is a generalized TBox abduction problem. For each missing is-a relation, the solutions computed by RepOSE are visualized using two sets - Source and Target. The missing is-a relation can then be repaired by adding an is-a relation between an element from Source and an element from Target.

Once the solutions are computed, the missing is-a relations are ranked with respect to the number of possible repairing actions. The first missing is-a relation in the list has the fewest possible repairing actions, and may therefore be a good starting point. When the user chooses a missing is-a relation, its Source and Target sets are displayed on the left and right, respectively, within the `Repairing Actions` panel (Figure 5). Both have zoom control and can be opened in a separate window. Similarly to the displays for wrong is-a relations, concepts in the missing is-a relations are highlighted in red, existing asserted is-a relations are shown in grey, unrepaired missing is-a relations in blue and added repairing actions for the missing is-a relations in black. For instance, Figure 5 shows the Source and Target sets for the missing is-a relation (*lower respiratory tract cartilage, cartilage*), which contain 2 and 21 concepts, respectively. The Target panel shows also the unrepaired missing is-a relation (*nasal septum, nasal cartilage*). The `Justifications of current relation` panel is a read-only panel that displays the justifications of the current missing is-a relation as an extra aid.

The user can repair the missing is-a relation by selecting a concept in the Source panel and a concept in the Target panel and clicking on the `Repair` button. RepOSE checks for possible conflicts and all consequences of a chosen repair are computed (such as changes in the repairing actions of other is-a relations and mappings and changes in the lists of wrong and missing is-a relations and mappings).

The tab 'Step 6: Repair Missing Mappings' is used for repairing missing mappings. The main difference with the tab for repairing missing is-a relations is that we deal with two ontologies and their alignment and that the repairing actions can be is-a relations within an ontology as well as mappings. Further, there are no justifications for missing mappings generated by the alignment process only.

3 Conclusion

In this paper we showed the use of RepOSE, a system that supports a unified approach for debugging and alignment of taxonomies. To our knowledge it is the first system in its kind. In the demonstration we show a debugging session for parts of the experiments in [1].

Acknowledgements. We thank the Swedish Research Council (Vetenskapsrådet) and the Swedish e-Science Research Centre (SeRC) for financial support.

References

1. Ivanova, V., Lambrix, P.: A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 1–15. Springer, Heidelberg (2013)
2. Lambrix, P., Dragisic, Z., Ivanova, V.: Get my pizza right: Repairing missing is-a relations in *ALC* ontologies. In: Takeda, H., Qu, Y., Mizoguchi, R., Kitamura, Y. (eds.) JIST 2012. LNCS, vol. 7774, pp. 17–32. Springer, Heidelberg (2013)
3. Lambrix, P., Tan, H.: SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics* 4(3), 196–206 (2006)

ALASKA for Ontology Based Data Access

Jean-François Baget, Madalina Croitoru, and Bruno Paiva Lima da Silva

LIRMM (University of Montpellier II & CNRS), INRIA Sophia-Antipolis, France

Abstract. Choosing the tools for the management of large and semi-structured knowledge bases has always been considered as a quite crafty task. This is due to the emergence of different solutions in a short period of time, and also to the lack of benchmarking available solutions. In this paper, we use ALASKA, a logical framework, that enables the comparison of different storage solutions at the same logical level. ALASKA translates different data representation languages such as relational databases, graph structures or RDF triples into logics. We use the platform to load semi-structured knowledge bases, store, and perform conjunctive queries over relational and non-relational storage systems.

1 Motivation and Impact

The ONTOLOGY-BASED DATA ACCESS (ODBA) problem [4] takes a set of facts, an ontology and a conjunctive query and aims to find if there is an answer / all the answers to the query in the facts (eventually enriched by the ontology). Several languages have been proposed in the literature where the language expressiveness / tractability trade-off is justified by the needs of given applications. In description logics, the need to answer conjunctive queries has led to the definition and study of less expressive languages, such as the \mathcal{EL} ([1]) and DL-Lite families [2]. Properties of these languages were used to define profiles of the Semantic Web OWL 2 language (www.w3.org/TR/owl-overview).

When the above languages are used by real world application, they are encoded in different data structures (e.g. relational databases, Triple Stores, graph structures). Justification for data structure choice include (1) storage speed (important for enriching the facts with the ontology) and (2) query efficiency. Therefore, deciding on what data structure is best for one's application is a tedious task. While storing RDF(S) has been investigated from a database inspired structure [3], other logical languages did not have the same privilege. Even RDF(S), often seen as a *graph*, has not been thoroughly investigated from a *ODBA perspective wrt graph structures* and emergence of *graph databases* in the NoSQL world.

This demo will allow to answer the following research question: “*How to design an unifying logic-based architecture for ontology-based data access?*”.

2 ALASKA

We thus demonstrate the ALASKA (acronym stands for **A**bstract and **L**ogic-based **A**rchitecture for **S**torage systems and **K**nowledge bases **A**nalysis) platform. ALASKA's goal is to enable and perform ODBA in a logical, generic manner, over existing, heterogeneous storage systems. The platform architecture is multi-layered.

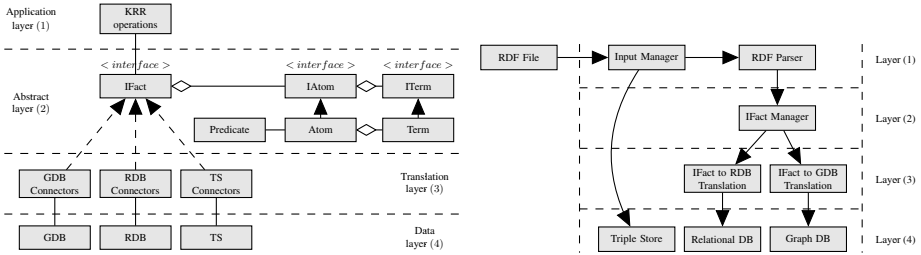


Fig. 1. ALASKA class diagram, and workflow of storage protocol

The first layer is (1) the **application** layer. Programs in this layer use data structures and call methods defined in the (2) **abstract** layer. Under the abstract layer, the (3) **translation** layer contains functions by which logical expressions are translated into the languages of several storage systems. Those systems, when connected to the rest of the architecture, compose the (4) **data** layer. Performing higher level reasoning operations within this architecture consists of writing programs and functions that use exclusively the formalism defined in the abstract layer. Once this is done, every program becomes compatible to any storage system connected to architecture.

To have a functional architecture, representative storage systems were selected. The systems already connected to ALASKA are listed below (please note that this list is not final and subject to constant updates):

- **Relational databases:** Sqlite 3.3.6¹, MySQL 5.0.77²
- **Graph databases:** Neo4J 1.8.1³, DEX 4.7⁴, OrientDB 1.0rc6⁵, HyperGraphDB 1.1⁶
- **Triples Stores:** Jena TDB 0.9.4⁷

Figure 1 displays, on the left-hand side, the class diagram of the architecture. On the right-hand side the workflow of knowledge base storing is illustrated. Let us analyse the workflow. We consider a RDF file as input. The RDF file is passed to the Input Manager (layer 1). According to the storage system needs the Input Manager directs it accordingly. If the RDF file will be stored in a Triple Store than the file is directly passed to the Triple Store of choice (layer 4). If the RDF file needs to be stored in a graph database the file is first transformed in an IFact object (layer 2). It is then translated (layer 3) to the language of the system of choice (graph database in this case) before being stored onto disk (layer 4).

Querying in ALASKA follows a similar workflow as the storage. In Figure 2, on the left hand side we show the storing workflow for storing a fact F in either a relational database or a graph database (for simplification reasons). On the right hand side of

¹ <http://www.sqlite.org/>
² <http://www.mysql.com/>
³ <http://www.neo4j.org/>
⁴ <http://www.sparsity-technologies.com/dex>
⁵ <http://www.orienttechnologies.com/orient-db.htm>
⁶ <http://www.hypergraphdb.org/>
⁷ <http://jena.sourceforge.net/>

the figure the querying workflow is depicted for graph and relational databases. Let us consider a fact F both stored in a relational database and in a graph database. Let us also consider a query Q . This query can either be expressed in SQL (or in a graph language of choice) and be sent directly to the respective storage system (e.g. the SQL Q query to the F in the relational database). Alternatively, the query can be translated in the abstract logic language and a generic backtrack algorithm used for answering Q in F . This generic backtrack algorithm will solely use the native language “elementary” operations for accessing data.

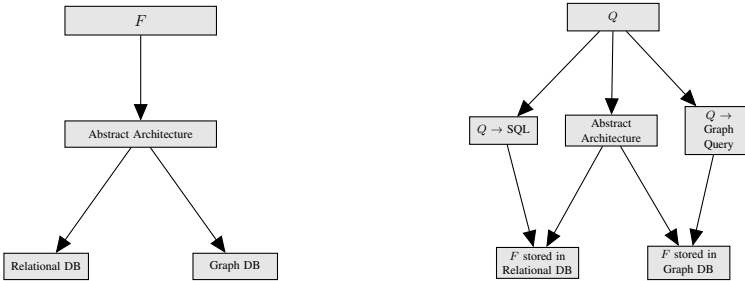


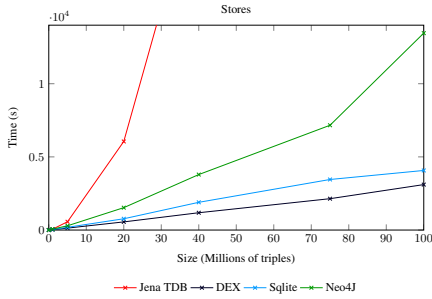
Fig. 2. ALASKA storage and querying workflow

3 ALASKA Demo Procedure

In a nutshell, the demo procedure of ALASKA goes as follows. Given a knowledge base (user provided or selected amongst benchmarks provided by ALASKA) a set of storage systems of interest are selected by the user. The knowledge base is then transformed in the Abstract Architecture and consequently stored in the selected systems. The storage time per system is then showed to the user (excluding the time needed for translation into Abstract Layer). Once the storage step is finished, users are able to perform conjunctive queries over the knowledge bases and, once again, compare the time of each system for query answering.

Let us consider an example. The knowledge base used here has been introduced by the SP2B project [5]. The SP2B project supplies a generator that creates knowledge bases with a certain parametrised quantity of triples maintaining a similar structure to the original DBLP knowledge base. The generator was used to create 5 knowledge bases of increasing sizes (5 million triples, 20, 40, 75 and respectively 100). Each of the knowledge bases has been stored in Jena, DEX, SQLite and Neo4J. In Figure 3 we show the time for storing the knowledge bases and their respective sizes on disk.

The user can see that the behavior of Jena is worse than the other storage systems. This is due to the Jena RDF parser uses central memory for buffering purposes when loading a file. For comparison, the other systems use the custom made RDF parser of ALASKA. Let us also note that DEX behaves much better than Neo4J and this is due to the fact that ACID transactions are not required for DEX (while being respected by Neo4J). Second, the size of storage is also available to the user. One can see, for instance, that the size of the knowledge base stored in DEX and Neo4J is well under



System	Size of the stored knowledge bases				
	5M	20M	40M	75M	100M
DEX	55 Mb	214.2 Mb	421.7 Mb	785.1 Mb	1.0 Gb
Neo4J	157.4 Mb	629.5 Mb	1.2 Gb	2.3 Gb	3.1 Gb
Sqlite	767.4 Mb	2.9 Gb	6.0 Gb	11.6 Gb	15.5 Gb
Jena TDB	1.1 Gb	3.9 Gb	7.5 Gb	-	-
RDF File	533.2 Mb	2.1 Gb	4.2 Gb	7.8 Gb	10.4 Gb

Fig. 3. Storage time and KB sizes in different systems

the size of initial RDF file. However, the size of the file stored in Jena is bigger than the one stored in SQLite and bigger than the initial size of the RDF file.

Once the storage step is finished, users are able to perform conjunctive queries. As already explained, querying the newly-stored knowledge base using the native interrogation engine (SQL for relational databases, SPARQL for 3Stores, etc.) is still possible with ALASKA. However, ALASKA also allows the possibility to perform conjunctive queries that access any storage system included in the platform using the same backtrack algorithm. The queries we have used here are:

1. `type(X, Article)`
Returns all the elements which are of type article.
2. `creator(X, PaulErdoes) . creator(X, Y)`
Returns the persons and the papers that were written with Paul Erdoes.
3. `type(X, Article) . journal(X, Journal1-1940) . creator(X, Y)`
Returns the creators of all the elements that are articles and were published in Journal 1 (1940).
4. `type(X, Article) . creator(X, PaulErdoes)`
Returns all the articles created by Paul Erdoes.

In the graphs in Figure 4 and 5 we show the combination storage and querying algorithm. For instance Jena(BT) stands for using Jena for elementary access operations and the generic backtrack for querying. SQLite(SQL) uses directly the SQL querying engine over the data stored in SQLite. In the graph corresponding to Q1 we also study the behavior of SQLite using the generic backtrack. For other queries we did not show it because the behavior is much worse than the other systems. We can also observe that for Q1, Q3 and Q4 queries SQLite and Jena behave faster than the graph bases. However, for Q2 this is no longer the case. In this case the fastest system for the generic backtrack is Jena followed by Neo4J and DEX, while SQLite explodes. The intuition behind this behavior is due to the phase transition phenomenon in relational databases but these aspects are out of the scope of this demonstration.

4 Discussion

An abstract platform (ALASKA) was created in order to perform storage operations independently of the data location. In order to enable the comparison between different

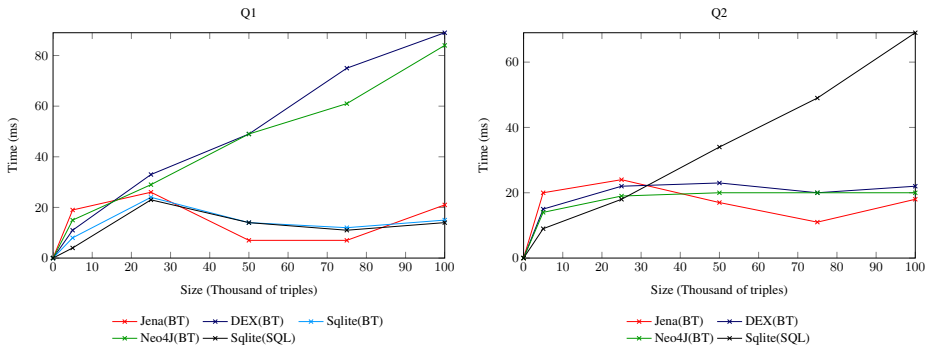


Fig. 4. Querying performance using ALASKA for large knowledge bases: Q1 and Q2

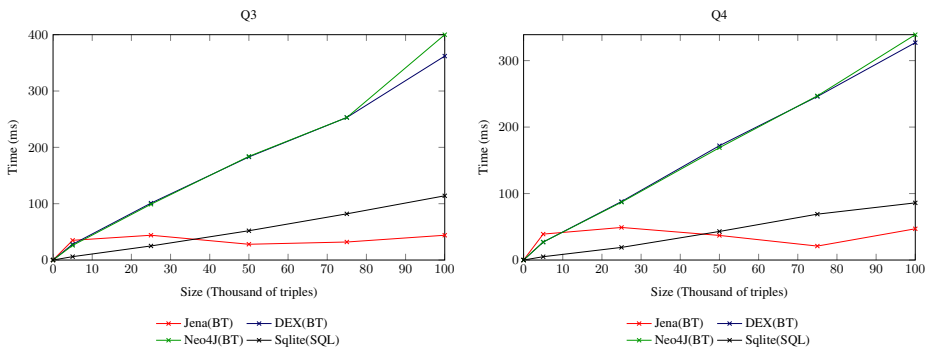


Fig. 5. Querying performance using ALASKA for large knowledge bases: Q3 and Q4

storage paradigms, ALASKA has to translate a knowledge base from a common language (i.e. First Order Logic) into different other representation language. Comparing different storage and querying paradigms becomes then possible. A knowledge base stored in a relational database can be also stored in a graph based database as well as a Triple Store and queried with an in built SPARQL engine etc.

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the el envelope. In: Proc. of IJCAI 2005 (2005)
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. Haslhofer, B., Rooki, E.M., Schandl, B., Zander, S.: Europeana RDF store report. Technical report, University of Vienna, Vienna (March 2011)
4. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of PODS 2002 (2002)
5. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: Sp2bench: A sparql performance benchmark. CoRR, abs/0806.4627 (2008)

Multilingual MoKi: How to Manage Multilingual Ontologies in a Wiki

Mauro Dragoni¹, Chiara Ghidini¹, and Alessio Bosca²

¹ FBK-irst, Via Sommarive 18 Povo, I-38123, Trento, Italy

² Celi s.r.l., Via S.Quintino 31, I-10131, Torino, Italy

Abstract. In this paper we describe an extension of the MoKi tool able to support the management of multilingual ontologies. The multilingual features of MoKi are based on an integration with Dictionary Based Translation and Machine Translation technologies. Also the collaborative features of MoKi are used to support the interaction between domain experts in order to discuss and agree on the translations of the terms to be used in each language.

1 Research Background

The construction of multilingual ontologies has become an important objective for organizations working in multilingual environments. As described in [3], obtaining multilingual ontologies is a complex activity which requires to tackle a number of problems spanning from the translation of the labels and description associated to a given ontology entity to the adaptation of the ontology to a concrete language and cultural community.

In this paper we describe an extension of the MoKi tool [4] able to support: (i) an automatic translation of labels and description associated to a given ontology entity, and (ii) the collaboration between domain experts in order to reach an agreement on the terms to be used in each language. This extension of MoKi is produced in the context of the Organic.Lingua EU project¹ where a multilingual version of an Organic Agriculture ontology is produced, starting from an English version, in order to support content classification and information retrieval in at least 16 different languages (Estonian, Slovenian, Romanian, Turkish, English, Spanish, Greek, German, Hungarian, Russian, Bulgarian, Hindi, French, Armenian, Dutch, Norwegian).

The first part of the extension is based on the integration of MoKi with a translation service able to provide translations for both individual labels and longer textual descriptions of a given entity. The second part of the extension takes advantage of the collaborative, wiki-style, characteristics of MoKi. In the following we better elaborate on these two aspects.

Language technologies and ontology translation. The translation of a domain specific ontology such as the Organic Agriculture ontology used in the Organic.Lingua project requires the usage of different translation techniques. In fact, the translation of such ontologies concerns both the translation of labels, usually composed by (single or multi)

¹ <http://www.organic-lingua.eu/>

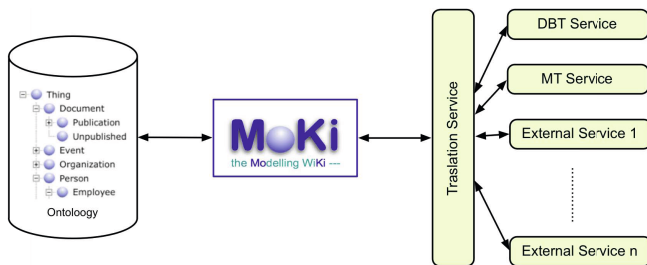


Fig. 1. MoKi and the translation services

words, and the translation of longer textual descriptions that can be associated to a given entity. To take into account the specificity of the domain (e.g., the agro-ecology domain), an adaptation of the translation models should also be adopted. For this reason MoKi is integrated with a translation service able to automatically re-direct a translation request to three different translation sub-services (see Figure 1): (i) a dictionary based translation (DBT) sub-service provided by Celi s.r.l. used to translate the ontology labels; (ii) a machine translation (MT) sub-service provided by Xerox for full-text translation and adapted to the organic agriculture domain for 8 language pairs; and (iii) a connection to external translation sub-services (such as Google translate) for the pairs of languages not supported in (ii).

Collaborative agreement on term translation. Given the complexity of translating domain specific ontologies, translations often need to be checked and agreed upon by a community of experts. This is especially true when ontologies are used to represent terminological standards which need to be carefully discussed and evaluated. To support this collaborative activity we foresee the usages of the wiki-style features of MoKi, expanded with the possibility of assigning specific translations of ontology entities to specific experts who need to monitor, check, and approve the suggested translations.

2 The Multilingual MoKi Tool

MoKi² is a collaborative MediaWiki-based [8] tool for modeling ontological and procedural knowledge. The main idea behind MoKi is to associate a wiki page, containing both unstructured and structured information, to each entity of the ontology and process model to support on-line collaboration between members of the modeling team, including collaboration between domain experts and knowledge engineers. An extensive description of the MoKi tool and of its main features can be found in [5,4].

In the context of the Organic.Lingua EU-Project, MoKi enables users (domain experts) to manage the editing of both the Organic.Lingua ontology and the Learning Object Metadata (LOM) ontology, as well as their translations. Here we briefly describe the MoKi main features, with a particular emphasis on the multilingual components and the technologies used to implement the DBT and MT translation services.

² See also <http://moki.fbk.eu>

2.1 The MoKi Features

The version of MoKi presented in this paper is equipped with five groups of functionalities, described below, that can be accessed via a wikis style menu bar.

Ontology Import and Export. This set of functionalities support the automatic import and export of knowledge in and from MoKi (see [4]). Here we have added a filtering criterion that enables the export of the ontology in a specific (set of) language(s).

Ontology Editor. This set of functionalities enables the management of the ontology (see [4]) and of its translation.

Interface Translator. This set of functionalities is used to add a new language to the MoKi interface and manage the translation of the labels that define the interface. It is based on the multi-lingual features of MediaWiki, extended with the translation features connected with MoKi.

Ontology Translator. This set of functionalities manage the translation operations required by MoKi. When a translation is requested (of an entity label or description), these functionalities contact the Translation service in order to obtain one. Also, when a new language is added to the ontology, the Ontology Translator retrieves, for each entity described in the ontology, the translations of their labels and descriptions. Then, new assertions, representing the relationships between the entities and their new translations, are added into the ontology.

Approval and Discussion. This set is composed of two parts: a first part that monitors the activities that are performed on the entity pages, and a second part that manages the discussions associated with each entity page. The module implemented in MoKi permits to manage the approval requests that are automatically generated when changes are carried out on entities. Indeed, when a user updates the translation of a label or a description, MoKi generates an approval request that is automatically received by the users in charge to review the translations of a particular language. Besides the possibility of approving the change, they are able to use the discussion facility of MoKi for discussing about the correctness of the new translation.

2.2 The Translation Features

The version of MoKi presented in this paper is connected, via the Translation Service, to two main types of translation features: a DBT service and a MT service.

The DBT service This consist of a language identifier, morphological analyzers and a dictionary translation lookup components; a short overview of these components is presented in the followings.

Language Identifier. A Language Identifier is a software module that is necessary whenever the language of a given textual resource (i.e. queries, metadata) is not explicitly known. The approach used here implements three different algorithms: a Character N-gram based strategy, a Word frequency based strategy, and a Function word based strategy [1].

Morphological Analyzer. A Morphological Analyzer is a software module capable of associating morphological features (grammatical category, tense, number, gender, etc.) to terms as well as performing tokenization, lemmatization, decompounding, multi-word detection and POS (part of speech) tagging. The service is based on proprietary and open source solutions (see [2,9]).

Translation Dictionaries. Translation Dictionaries are mappings between terms in different languages. The functionality provided by this software component consists in retrieving all the translations in a target language available for a given term. The translations are performed by using the lemmas of terms. If no suitable translations are found in the dictionary, the module searches for translations of the term's lemma via a predefined bridge language.

The MT Service. This service is tailored for full-text translation as opposed to the translation of keywords or terms. The underlying translation model is a Phrase-based Statistical Translation model [7]. The service is currently available for the following 8 language pairs: English from/into German, French, Italian and Spanish. The underlying decoder (translator) used in the context of the Organic.Lingua project is based on Moses [6]. There is a number of additional pre- and post- processing steps done with in-house (Xerox) tools which interact with the decoder such as de-compounding (for German), Named Entities Recognition, Brackets Management, Truecasing (recovering the case of the lowercased translations produced by the decoder), and so on. The translation models are adapted to the Organic.Lingua domain using a combination of in-domain (agriculture) and out-of-domain (Europarl, Wikipedia) parallel and monolingual texts. In-domain resources include: multilingual abstracts and titles extracted from bibliographical records on agricultural science and technology provided by FAO and INRA, sub-corpora of JRC-Aquis³ comprising documents annotated with domain-related EUROVOC categories (agriculture, forestry and fisheries, agri-foodstuffs), as well as terminological data from the agricultural multilingual thesaurus AGROVOC⁴.

3 System Demonstration

In the live demonstration we will present the MoKi features that are used to manage multilingual aspects of ontology in a collaborative way. We will first show how pages describing entities in a multilingual way look like, and we will describe the editing functionalities used to manage the translations of each entity label as well as their descriptions.

Then, we will demonstrate how the collaboration part may be used, by the domain experts, for discussing and approving the changes carried out on the entities. As regards the visualization functionalities, we will demonstrate how to get different overviews of the translated models, in particular we will show:

- how to display (and edit) in a tree-view the taxonomy or paratomy hierarchy of the element of the domain model in different languages;

³ <http://langtech.jrc.it/JRC-Acquis.html>

⁴ <http://aims.fao.org/website/AGROVOC-Thesaurus/sub>

- how to change the language used for visualizing the ontology in each view;
- how to compare different translations of the ontology and to perform quick edit operations.

We will show how it is possible to add a new language to the ontology and how to initialize the translations by invoking the machine translation service. Then, we will demonstrate how it is possible to add/update the languages used in the MoKi interface.

Finally, we will demonstrate how to export the ontology deployed in MoKi with the possibility to filter the exported axioms through language criteria.

4 Conclusions

In this demonstration we have presented a version of MoKi which supports the management of multilingual ontologies in a collaborative way. We are currently improving the tool in several directions, which span from including services for exposing the ontology, improving support for evolving the ontology, and implementing a module for the alignment between the content described in MoKi pages and external ontologies.

Acknowledgments. Organic.Lingua is funded under the ICT Support Program of the EU Commission (Grant Agreement Number 270999).

References

1. Bosca, A., Dini, L.: Language identification strategies for cross language information retrieval. Clef Working Notes (2010)
2. Bosca, A., Dini, L., Kouylekov, M., Trevisan, M.: Linguagrid: a network of linguistic and semantic services for the italian language. To be Published in LREC 2012 Conference (2012)
3. Espinoza, M., Gómez-Pérez, A., Mena, E.: Enriching an ontology with multilingual information. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 333–347. Springer, Heidelberg (2008)
4. Ghidini, C., Rospoche, M., Serafini, L.: Moki: a wiki-based conceptual modeling tool. In: ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China. CEUR Workshop Proceedings (CEUR-WS.org), vol. 658, pp. 77–80 (2010)
5. Ghidini, C., Rospoche, M., Serafini, L.: Conceptual modeling in wikis: a reference architecture and a tool. In: The Fourth International Conference on Information, Process, and Knowledge Management, eKNOW 2012 (2012)
6. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: open source toolkit for statistical machine translation. In: ACL 2007: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, pp. 177–180. Association for Computational Linguistics (2007)
7. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase based translation. In: Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics, HLT/NAACL (2003)
8. Wikimedia Foundation. Mediawiki, <http://www.mediawiki.org>
9. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of International Conference on New Methods in Language Processing, Manchester, UK (1994)

SAIM – One Step Closer to Zero-Configuration Link Discovery

Klaus Lyko, Konrad Höffner, René Speck,
Axel-Cyrille Ngonga Ngomo, and Jens Lehmann

Universität Leipzig, Postfach 100920, 04009 Leipzig, Germany
{lastname}@informatik.uni-leipzig.de
<http://aksw.org/projects/saim>

Abstract. Link discovery plays a central role in the implementation of the Linked Data vision. In this demo paper, we present SAIM, a tool that aims to support users during the creation of high-quality link specifications. The tool implements a simple but effective workflow to creating initial link specifications. In addition, SAIM implements a variety of state-of-the-art machine-learning algorithms for unsupervised, semi-supervised and supervised instance matching on structured data. We demonstrate SAIM by using benchmark data such as the OAEI datasets.

Keywords: Interlinking, Machine Learning, Data Integration.

1 Introduction

Links between instances are of central importance for a large number of tasks such as data integration, federated querying and knowledge retrieval as often pointed out in the literature [1]. Two main problems arise when trying to discover links between data sets or deduplicate data sets. First, naive solutions to Link Discovery (LD) need to compare all resources in the source dataset with all resources in the target dataset and, thus, have quadratic time complexity. Consequently, naive approaches are impractical when computing links across large datasets such as DBpedia [2]¹ or Yago². Time-efficient algorithms and frameworks such as LIMES [4] and SILK³ have been developed to reduce the number of comparisons which need to be made between resources. While these approaches achieve practicable runtimes even on large datasets, they do not guarantee the quality of the links that are returned by LD frameworks. Addressing this second problem of LD demands the development of techniques that can compute accurate *link specifications* for deciding whether two resources should be linked. Both supervised (e.g., [6,7]) and unsupervised machine-learning approaches (e.g., [8]) have been proposed to achieve this goal.

¹ <http://dbpedia.org>

² <http://www.mpi-inf.mpg.de/yago-naga/yago/>

³ <https://www.assembla.com/spaces/silk/>

SAIM⁴ encompasses solutions for both problems within a simple interface which implements a flexible workflow. The tool relies on algorithms implemented in LIMES⁵, which have been shown to outperform the state of the art in previous work w.r.t. time efficiency [3]. In addition to allowing expert users to create specifications manually, SAIM implements supervised and unsupervised learning algorithms including extensions of EAGLE [6] and the novel COALA [7] approach (presented at the same conference), which have been shown to lead to high-quality specifications. By these means, SAIM can support domain experts and lay users during the creation of link specifications. Moreover, it implements the time-efficient algorithms for class and property matching algorithms proposed in [5]. SAIM goes beyond existing interfaces for link discovery (e.g., SILK Workbench⁶) by supporting several self-configuration algorithms that allow the automatic creation of link specifications. In the following, we present the workflow underlying SAIM and then focus on the content of the SAIM demonstration.

2 SAIM

The workflow underlying SAIM consists of four main steps: (1) data selection, (2) schema matching, (3) creation of the specification and (4) execution of the specification. In the following, we explain how SAIM supports each of these steps.

2.1 Data Selection

SAIM allows users to specify SPARQL endpoints or local RDF files (for users with logins) as data sources. SPARQL endpoints are specified by stating the URL of the endpoint and (if necessary) the graph from which the data is to read. Moreover, each endpoint can be given a name. Our software signalsizes to its user whether the endpoint he selected is alive by issuing a simple SPARQL query to the specified endpoint. Moreover, it provides a list of commonly used endpoints such as DBpedia⁷. Local RDF files can be in any of the serialization formats supported by the Jena Framework⁸ on which SAIM relies. In addition, the tool supports using data stored as CSV files. In the latter case, the data is converted to RDF on the fly by using the strings contained in each column of the first row as property labels and the elements of the first column as URI for the resources.

2.2 Schema Matching

Our approach relies on simple yet the time-efficient schema matching algorithms presented for matching classes and properties (see Figure 1). The user can choose

⁴ SAIM stands for (Semi-)Automatic Instance Matcher and is pronounced like "same".

All information to the project including a demo and a screencast can be found at <http://aksw.org/projects/saim>.

⁵ See <http://limes.sf.net>.

⁶ https://www.assembla.com/spaces/silk/wiki/Silk_Workbench

⁷ <http://dbpedia.org/sparql>

⁸ <http://jena.apache.org/>

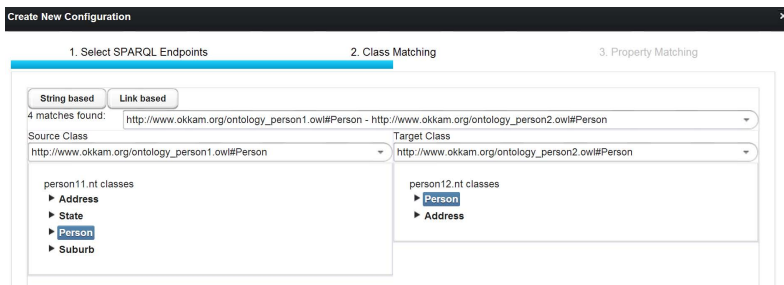


Fig. 1. Schema Matching step in SAIM

between intensional matching (**String based** button) and a matching approach based on stable marriage on links (**Link based** button, see [5] for more details). Per default, SAIM compute the string-based matching between the class labels by using the trigram similarity and return a sorted list of matching classes. We chose this approach because of its time-efficiency. The user can either choose the stable-marriage-based approach or navigate through the schemas of the dataset to perform the schema matching manually. Note that SAIM implements fallback solutions to be able to display the schemas of datasets with incomplete ontologies. For example, if no statement of the form $?x \text{ a } \text{rdf:Class}$ is found in the dataset, our approach falls back to retrieving all distinct $?x$ such that triples of the form $?y \text{ a } ?x$ is contained in the dataset.

2.3 Creation of Specifications

The creation of specifications is the most involved part of SAIM’s workflow. Once the schema matching has been carried out, the user is presented with SAIM’s main window. Initially, this window contains an *output node*. On the left, the expert user can choose between the different properties, several similarity and

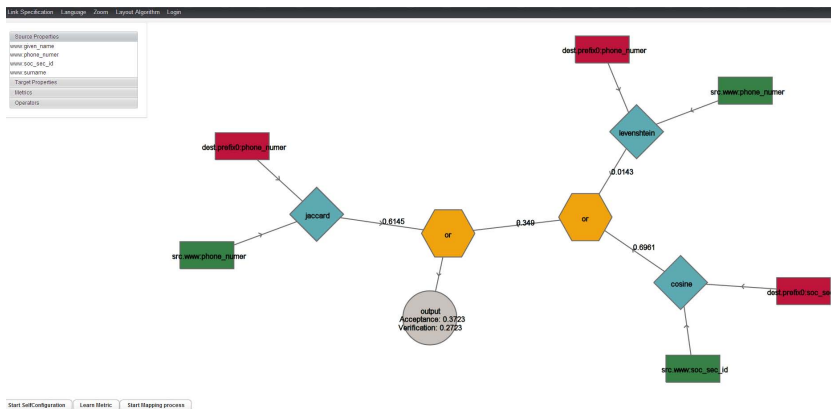


Fig. 2. SAIM specification window

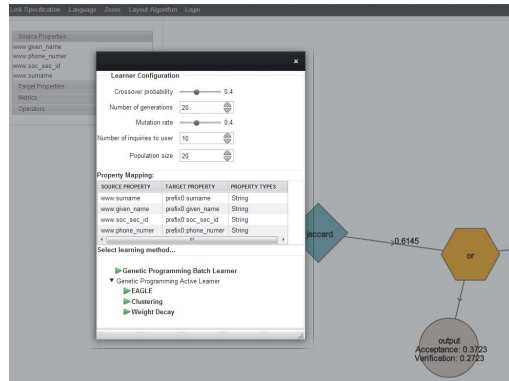


Fig. 3. Selection of algorithms

distance measures (incl. Levenshtein, Trigrams, Cosine) as well as operators (incl. AND, OR, MAX, MIN) to combine these metrics to a single specification manually (see Figure 2). The user can also choose the **Learn metric** button instead, which allows the user to select between several machine-learning algorithms for link specification learning including EAGLE [6] and its extensions in COALA [7] (Figure 3). After choosing these algorithms, the user is presented the most informative positive and negative examples and can choose whether they are matches or non-matches. SAIM supports the user in this process by allowing him to view the values of the properties of the resources that are part of the match or to dereference their URIs. SAIM also enables lay users to create link specification by offering a *self-configuration* mode. In the corresponding window (see Figure 4), SAIM allows the user to choose which algorithm to use and whether the specification should be tuned towards precision or recall (the default setting being that both are equally important). Once the user has selected a configuration, SAIM runs unsupervised machine-learning algorithms based on EAGLE or RAVEN and returns the specification that maximize a selected pseudo-F-measure. The specification shown in Figure 2 was learned fully automatically by the unsupervised version EAGLE.

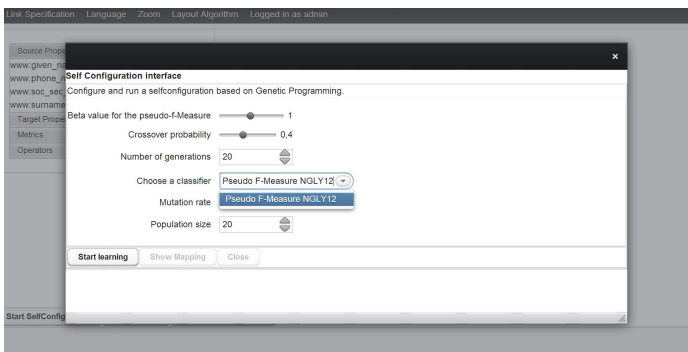


Fig. 4. Specification of self-configuration in SAIM

3 Demonstration

The goal of the demonstration will be to show the whole workflow described above from the datasets to the export of the resulting specification and links. We will begin by showing how SAIM deals with data in SPARQL endpoints and with local datasets. Thereafter, we will present and explain how SAIM suggests class and property matchings to the user. We will especially explain the hierarchy of fallback solutions that SAIM employs to generate both an overview of the class structure and the corresponding class matching as well as how it uses extensional and intensional schema matching approaches for both class and property matching. In a third step, we will then showcase the approaches implemented in SAIM. We will begin by showing how the expert user can use SAIM to create link specifications manually. Thereafter, we will present how domain experts can employ the active learning algorithms EAGLE and COALA to learn and refine link specifications iteratively. Then, we show how lay users can use unsupervised machine learning to have SAIM detect a high-quality link specification for them. Finally, we will demonstrate how the results of the specification process (i.e., the link specification and the resulting mappings) can be downloaded from SAIM for use in further applications. Throughout the demonstration, we will employ benchmark datasets such as those provided by the OAEI challenges ⁹.

4 Conclusions and Future Work

We present SAIM, an interface for the creation of high-quality link specifications. SAIM represents a further step towards the vision of zero-configuration link discovery as it allows users to create such specifications with minimal effort. In future work, we will extend SAIM with more algorithms for learning link specifications and aim to achieve our vision of easy and effective link discovery.

References

1. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to linked data and its lifecycle on the web. In: Polleres, A., d’Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 1–75. Springer, Heidelberg (2011)
2. Morsey, M., Lehmann, J., Auer, S., Stadler, C., Hellmann, S.: DBpedia and the live extraction of structured data from wikipedia. Program: Electronic Library and Information Systems 46, 27 (2012)
3. Ngonga Ngomo, A.-C.: On link discovery using a hybrid approach. Journal on Data Semantics 1, 203–217 (2012)
4. Ngonga Ngomo, A.-C., Auer, S.: LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data. In: Proceedings of IJCAI (2011)
5. Ngonga Ngomo, A.-C., Lehmann, J., Auer, S., Höffner, K.: RAVEN – Active Learning of Link Specifications. In: Proceedings of OM@ISWC, vol. 814 (2011)

⁹ <http://oaei.ontologymatching.org/>

6. Ngonga Ngomo, A.-C., Lyko, K.: EAGLE: Efficient active learning of link specifications using genetic programming. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 149–163. Springer, Heidelberg (2012)
7. Ngonga Ngomo, A.-C., Lyko, K., Christen, V.: COALA – correlation-aware active learning of link specifications. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 442–456. Springer, Heidelberg (2013)
8. Nikolov, A., d’Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012)

Linked Data Query Wizard: A Tabular Interface for the Semantic Web

Patrick Hoefler¹, Michael Granitzer²,
Vedran Sabol¹, and Stefanie Lindstaedt¹

¹ Know-Center, Graz, Austria

² University of Passau, Germany

{phoefler,vsabol,slind}@know-center.at,
michael.granitzer@uni-passau.de

Abstract. Linked Data has become an essential part of the Semantic Web. A lot of Linked Data is already available in the Linked Open Data cloud, which keeps growing due to an influx of new data from research and open government activities. However, it is still quite difficult to access this wealth of semantically enriched data directly without having in-depth knowledge about SPARQL and related semantic technologies. In this paper, we present the Linked Data Query Wizard, a prototype that provides a Linked Data interface for non-expert users, focusing on keyword search as an entry point and a tabular interface providing simple functionality for filtering and exploration.

Keywords: linked data, interfaces, semantic web, sparql, rdf.

1 Introduction

The Linked Open Data cloud provides an impressive wealth of semantically enriched, openly available Linked Data. However, this Linked Data is basically only accessible for experts in semantic technologies who know how to write SPARQL queries. And even for those who know how to use SPARQL, it can be quite laborious at times, especially while trying to explore an unknown SPARQL endpoint.

Therefore, the goal of the presented prototype – the Linked Data Query Wizard¹ – is to provide an easy-to-use interface for accessing Linked Data. It should be suitable for non-expert users without any prior knowledge of SPARQL or other semantic technologies.

The working hypothesis for the Linked Data Query Wizard is: There are not that many people who speak SPARQL and are familiar with graph structures. On the other hand, many people know spreadsheet applications like Microsoft Excel. Therefore, the idea is to develop a web-based tool that brings the graph structure of Linked Data into tabular form (see figure 1) and provides easy-to-use interaction possibilities for filtering and exploring Linked Data by using metaphors and techniques the users already know.

¹ <http://code.know-center.tugraz.at/search>

CODE Linked Data Query Wizard

Label [graz] x	Description	Type Funding	Partner 999977948	PartnerRole	Amount	Add column ...
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project MATURE		Funding	999977948	Partner	682950	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project CopPeR		Funding	999977948	Partner	525452	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project SECO		Funding	999977948	Partner	564378	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project ECRYPT II		Funding	999977948	Partner	132000	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project MOGENTES		Funding	999977948	Partner	445000	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project COCONUT		Funding	999977948	Partner	296716	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project IMPACT		Funding	999977948	Partner	681982	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project ICT-ENSURE		Funding	999977948	Coordinator	668886	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project PRE-DRIVE		Funding	999977948	Partner	62836	
Funding of partner TECHNISCHE UNIVERSITAET GRAZ in project CHOSEN		Funding	999977948	Partner	256620	

Displaying 10 of 41 results

[Load more results ...](#)

This is CODEResearch in progress.

Fig. 1. Current results page of the Linked Data Query Wizard

2 Research Context

The Linked Data Query Wizard is part of the CODE project [1], a research project funded by the European Union. As described in [2], the vision of CODE is to establish a sophisticated ecosystem for Linked Data. The extraction of knowledge encapsulated in scientific research papers along with its release as Linked Data serves as the major use case. A web-based visual analytics interface should empower end users to analyze, integrate, and organize data.

Our prototype is part of CODE's Visual Analytics work package [3]. The goal of the work package is to develop a web-based visual analytics platform that enables non-expert users to engage in a visually supported, collaborative analysis of Linked Data, and the Linked Data Query Wizard will play a crucial role in this undertaking.

3 Related Work

Although the Semantic Web has matured in recent years, and semantic technologies have become quite powerful, the Linked Open Data cloud is still only accessible for semantic technology experts and programmers. The problem of easy-to-use interfaces for accessing Linked Data is still largely unsolved. The majority of current tools are not aimed at non-expert users. As an example, the popular Semantic Web search engine Sindice [4] is practically unusable for people without a deep understanding of semantic technologies.

Currently, only very few web-based tools use tables for representing Linked Data. One such example would be Freebase Parallax [5]. Although its main feature is the ability to browse sets of related things, it also provides a table view for these result sets. Another online tool that shares similarities with our prototype is the Falcons Explorer [6]. Both tools feature a search box as the main entry point – an idea that is also central to our prototype. However, in

both tools, the table view is not the central focus, and usability is hindered by a multitude of different viewing options that are bound to confuse inexperienced users.

Another tool that shares a few similarities with our prototype is OpenRefine [7] (formerly known as Google Refine and Freebase Gridworks). It supports RDF, and there are also extensions such as LODRefine [8] that focus on Linked Data – however, OpenRefine’s main focus is cleaning up tabular data, and it’s also not available as a web service, even though its main interface is browser-based.

Our prototype also supports the current working draft of W3C’s RDF Data Cube Vocabulary [9] which provides a semantic framework for expressing statistical datasets as Linked Data. Datasets that comply with the RDF Data Cube standard can easily be displayed, filtered, and explored using the Linked Data Query Wizard. Because this standard is still very young, to the best of our knowledge, there are currently no other tools – with a feature set similar to our prototype – that support RDF Data Cubes.

4 Key Technologies

The Linked Data Query Wizard is a purely web-based system. It uses Python 2.7 on the back end and HTML5, CSS3, and JavaScript – compiled from CoffeeScript – on the front end. Its main SPARQL endpoint is provided by a bigdata [10] RDF database, which also provides an integrated full-text search [11].

Sadly, full-text search is sorely lacking from the current SPARQL specification [12], which is why certain SPARQL endpoints have come up with workaround solutions. Therefore, only Virtuoso and bigdata are currently supported as SPARQL endpoints by the Linked Data Query Wizard. However, since one of our prototype’s design philosophies is to use Semantic Web standards such as SPARQL wherever possible, support for other suitable endpoints could be added with minimal effort at a later point.

The Linked Data Query Wizard also makes use of certain SPARQL 1.1 features, especially the aggregation functions. The `COUNT()` function is critical and already in use by the current prototype for displaying the number of results for a given query. It is also planned to extend the prototype in order to let users calculate simple aggregate functions for a given dataset, using functions such as `SUM()` and `AVG()`. For advanced filtering mechanisms, functions such as `MIN()` and `MAX()` might be needed as well.

5 Initial Results

The Linked Data Query Wizard is currently available online as an early beta version. In its current form, it offers two entry points: Users can either initiate a keyword search, or they can select any available dataset, represented as an RDF Data Cube (see figure 2). In both cases, the users get presented with the results in tabular form, similar to what they are used from spreadsheet applications.

They can choose which columns (i.e. RDF predicates) they are interested in, and they can set filters to narrow down the displayed data.

Though the current functionality of the prototype is still rather limited, first usage experiments have shown that the tool can be helpful in exploring the data and respective data structures of unknown SPARQL endpoints. For example, with one keyword search and less than 20 clicks, it is possible to either create a dataset of all coordinators of finished EU FP7 research projects and their received funding, or a list of public companies located in the US and their number of employees.

Additionally, the Linked Data Query Wizard has been under permanent scrutiny of fellow researchers from the CODE project for several weeks now, providing us with valuable feedback on stability and usability issues as well as helpful feature requests.

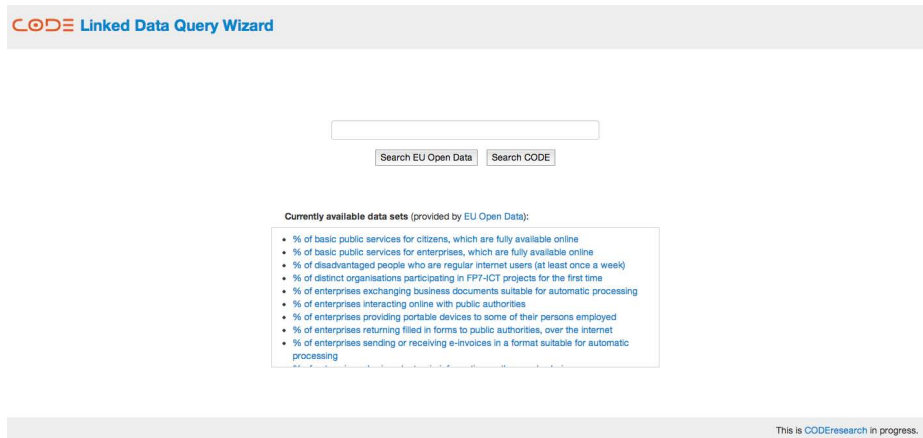


Fig. 2. Current entry page of the Linked Data Query Wizard

6 Conclusions and Future Work

In this paper, we presented the Linked Data Query Wizard, a tabular approach for filtering and exploring Linked Data.

The next steps regarding the prototype will be to expand its functionality, focusing on better filter mechanisms as well as more advanced exploration features that incorporate the underlying semantic structure. Additionally, the Linked Data Query Wizard will be integrated with a tool for visualizing Linked Data as well as a semantic enrichment service for turning generic RDF into RDF Data Cubes.

The development of the prototype will continue throughout the rest of the year, leading to a final evaluation at the beginning of 2014.

Acknowledgments. The Linked Data Query Wizard is being developed within the CODE project at the Know-Center, Graz, Austria. The CODE project is funded by the EU Seventh Framework Programme, grant agreement number 296150. The Know-Center is funded within the Austrian COMET Program – Competence Centers for Excellent Technologies – under the auspices of the Austrian Federal Ministry of Transport, Innovation and Technology, the Austrian Federal Ministry of Economy, Family and Youth, and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency (FFG).

References

1. CODE: Commercially Empowered Linked Open Data Ecosystems in Research, <http://code-research.eu/>
2. Stegmaier, F., Seifert, C., Kern, R., Hoefler, P., Bayerl, S., Granitzer, M., Kosch, H., et al.: Unleashing Semantics of Research Data. In: Proceedings of the Second Workshop on Big Data Benchmarking (WBDB 2012), Pune, India (2012)
3. CODE: Visual Analytics, <http://code-research.eu/visual-analytics>
4. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
5. Huynh, D.F., Karger, D.R.: Parallax and Companion: Set-based Browsing for the Data Web. In: WWW Conference. ACM (2009)
6. Cheng, G., Wu, H., Gong, S., Ge, W., Qu, Y.: Falcons Explorer: Tabular and Relational End-user Programming for the Web of Data. In: Semantic Web Challenge (2010)
7. OpenRefine, <https://github.com/OpenRefine>
8. LODRefine, <http://code.zemanta.com/sparkica/>
9. W3C: RDF Data Cube Vocabulary, <http://www.w3.org/TR/vocab-data-cube/>
10. bigdata, <http://www.systap.com/bigdata.htm>
11. bigdata Full Text Search, <http://sourceforge.net/apps/mediawiki/bigdata/?title=FullTextSearch>
12. W3C: SPARQL 1.1 Query Language, <http://www.w3.org/TR/sparql11-query/>

Facilitating Music Information Research with Shared Open Vocabularies

Alo Allik, György Fazekas, Simon Dixon, and Mark Sandler

Queen Mary University of London

{alo.allik,gyorgy.fazekas,simon.dixon,mark.sandler}@eecs.qmul.ac.uk

Abstract. There is currently no agreement on common shared representations of audio features in the field of music information retrieval. The Audio Feature Ontology has been developed as part of a harmonised library of modular ontologies to solve the problem of interoperability between music related data sources. We demonstrate a software framework which combines this ontology and related Semantic Web technologies with data extraction and analysis software, in order to enhance audio feature extraction workflows.

Keywords: semantic audio analysis, music information retrieval, linked open data, Semantic Web technologies.

1 Introduction

Researchers in audio and music information retrieval increasingly use a common set of feature extraction techniques to characterise audio material, and large data sets of features are released for commercial and scientific use. The development of data sets and research tools however is not governed by shared open vocabularies and common data structures. This raises several issues including the lack of interoperability between music analysis tools and the need for adapting program code and software environments for a variety of different formats. There is growing awareness in the music informatics community of the necessity of common representations using Semantic Web technologies and linked data [5].

We demonstrate a software framework which combines Semantic Web technologies with data extraction and analysis software in order to enhance audio feature extraction workflows and improve interoperability and sustainability. The main interface to the framework is the Sonic Annotator Web Application (SAWA), an online semantic audio analysis interface, which is accessible at <http://www.isophonics.net/sawa/>. The main contribution of this work is the development of the new Audio Feature Ontology. The previous version of the ontology [1] had relatively poor adoption in research communities, due to its limited domain coverage and incomplete tool support.

2 Components

Our framework utilises a number of software components which were originally developed during the OMRAS2 project[1]. The software components include a

set of extensible ontologies, a plugin interface for audio feature extraction and annotation, and open source research tools. The Audio Feature (AF) Ontology was created within the framework of the Music Ontology[2], the core of a harmonised library of music related ontologies. AF provides a model for describing acoustical and musicological data, and allows for publishing content-derived information about audio recordings. Its aim is to provide a framework for communication. This ontology is currently being updated in the course of an ongoing research project to provide a better coverage of features commonly used by researchers, to enable better harmonisation with existing software tools, and to support a wider set of data sets and use cases.

Vamp is a plugin system designed for audio feature extraction. It is a modular and extensible collection of shared libraries that are meant to be embedded in a host application. Vamp plugins accept audio data as input and produce structured data as output. The plugin system is supported by the Vamp Plugin Ontology within the context of Semantic Web data. It allows the association of richer metadata with each plugin - for example, information about plugin creators, licensing, or high level content description - and enables the description of plugin structure and configuration. The ontology also provides associations of plugin outputs with terms in the Audio Feature Ontology to express what the output describes.

Sonic Visualiser[3] is a desktop application for visualising audio and a wide range of automatic annotations, including anything from spectrograms to high-level information about musical structure. Sonic Visualiser can handle a wide range of audio formats, and supports the Vamp plugin interface. Sonic Annotator is an audio analysis application which applies Vamp feature extraction plugins to audio data in a batch. It is built using Sonic Visualiser libraries. The two applications therefore share capabilities such as broad support for different audio file formats, network retrieval of audio files, and the interpretation of feature extraction specifications in RDF using the Vamp Plugin Ontology.

Sonic Annotator Web Application (SAWA)[4] is a framework that ties the above described components into a coherent Web-application framework. It utilises Semantic Web vocabularies and ontologies, including the Audio Feature Ontology, which provides its internal model. The main objective of the SAWA framework is to facilitate the use of semantic audio analysis on the Web and support a wide range of different algorithms, while using a flexible, ontology-based common data model for representing information about audio analysis algorithms, configurations and results. SAWA also stores feature extraction results and avoids repeated computation of features given the same audio content and algorithm with the same parameter configuration. SAWA uses the Vamp plugin system to support a wide range of different audio feature extraction algorithms.

3 Web-Based Semantic Audio Analysis

This present work focusses on two novel aspects of interoperability within the described collection of components illustrating the practical applications of the

Audio Feature Ontology. First, the SAWA framework is utilised as a Web-based audio analysis system. The SAWA Feature Extractor is a comprehensive and user friendly feature extraction system, allowing the user to compile collections of audio files for batch analysis and to select desired feature plugins, eventually making the results available in multiple formats for download. The second part of the demonstration highlights how the SAWA SPARQL endpoint can be consumed from the Sonic Visualiser desktop application, and how SAWA can be utilised as an optimised data server in music information retrieval workflows.

SAWA is designed with the intention of providing an easy to use human interface as well as a SPARQL query interface. SAWA Feature Extractor works with a collection of audio files uploaded by the user. Optionally, each audio file is identified using a fingerprint, and basic bibliographic metadata given an associated identifier (PUID) is retrieved from the MusicBrainz service¹. The fingerprint serves as the identifier that is used for storing and later identifying provenance information and to facilitate RDF caching. SAWA Feature Extractor allows for

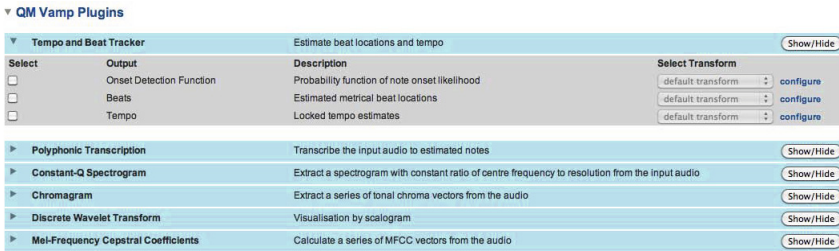


Fig. 1. SAWA interface for Vamp plugins

the selection and configuration of one or more Vamp outputs and the execution of transforms on previously uploaded files. The range of existing Vamp plugins include estimators for musically meaningful features (note onset detectors, beat and tempo estimators, structural segmentation, and key estimators), low-level audio feature extractors, metadata annotators and calculators for dense features that are often used in visualisations, such as chromagram and harmonic spectrum. Results, returned as RDF data, can be examined using an RDF browser, imported in Sonic Visualiser and viewed in context of the audio or published on the Semantic Web.

Figure 1 shows how a transform can be selected from a list of Vamp plugins and how for each plugin a specific set of parameters can be configured for execution. The user may click on the triangular symbols associated with Vamp plugins to expand the content such as the description of each output of a plugin. These transforms are saved temporarily and applied individually on each audio file after submitting a query. Once a set of feature extractors has been selected and configured, a query can be submitted to perform the analysis of all or a user defined

¹ <http://musicbrainz.org/>

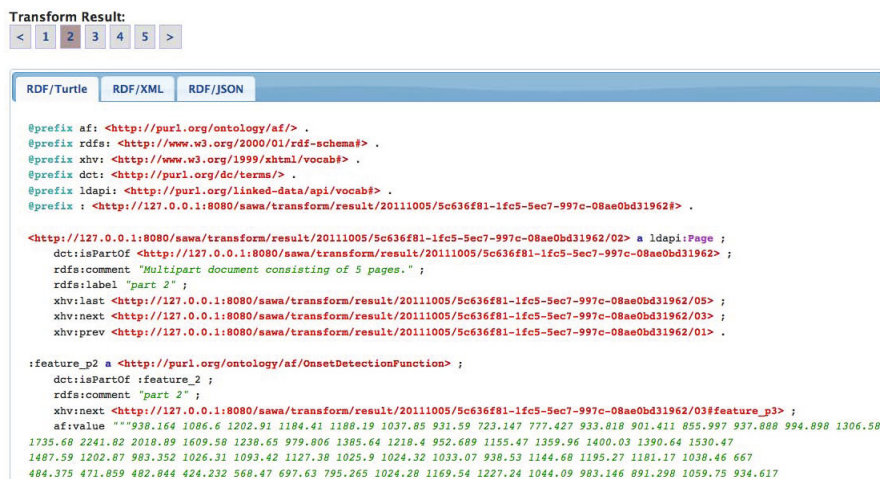


Fig. 2. Paginated feature extraction results in RDF/Turtle format

subset of the uploaded audio files. The progress of the analysis can be monitored using the status display. While the feature extraction process is running, the results are loaded by the Web application as they become available and a table of results is dynamically expanded. The feature extraction results are available in several different formats generated on demand from the default RDF/Turtle syntax. These results can be examined within the application, downloaded as uncompressed RDF or as a ZIP compressed file. Additionally, each result receives a permanent URI. This allows all transform results to be accessed later, even after the user session has expired. The URI is designed to include the date of feature extraction and a universally unique identifier (UUID) generated given the audio fingerprint and the transform configuration in order to avoid potential URI collisions. These permanent links are available in RDF/Turtle and RDF/XML formats. Results consisting of a large number of RDF triples and those that include large matrices are paginated.

The SPARQL-endpoint of SAWA Feature Extractor can be consumed from Sonic Visualiser. The SPARQL client accesses a user defined SPARQL-endpoint and converts the returned data into a format a Vamp host like Sonic Visualiser can understand. The simple query interface of this client is shown in Figure 3. This enables the user to incorporate the functionality of SAWA into the desktop feature extraction workflow and take advantage of the interoperability and efficiency the framework offers. The query results can be linked with other Web resources like MusicBrainz metadata, providing an enriched context for the audio content. For example, the files can be identified using the MusicDNS web service (<http://musicbrainz.org/doc/MusicDNS>) and thereby linked with all associated metadata available in the MusicBrainz database. The research workflow can be optimised through a SAWA facility that automatically stores feature extraction results so that these can be reused without having to perform the computation again.

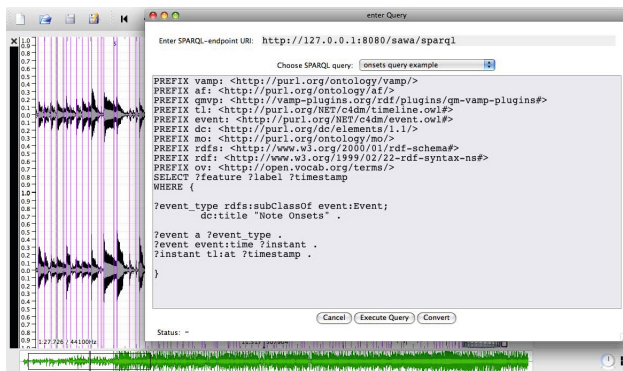


Fig. 3. SPARQL client in Sonic Visualiser

4 Conclusions

Our present work provides an overview of how the new Audio Feature Ontology in combination with related linked data vocabularies and open source research tools can be used to form a modular research framework for music information retrieval. The user is introduced to a simple Web-based interface as well as an open source desktop application, both of which enable extraction of musically meaningful information - such as pitch, loudness, timbre or note onsets - from audio files. The audio files can be linked up with online music metadata providing enriched content for music research workflows.

Acknowledgements. The development of the components presented in this demo was supported by the Engineering and Physical Sciences Research Council (EPSRC) through the OMRAS2 project EP/E017614/1, and by the JISC (<http://jisc.ac.uk>) funded Shared Open Vocabulary for Audio Research and Retrieval (SOVARR) project (<http://sovarr.c4dm.eecs.qmul.ac.uk/>).

References

1. Fazekas, G., Raimond, Y., Jakobson, K., Sandler, M.: An overview of semantic web activities in the OMRAS2 project. *Journal of New Music Research (JNMR)* 39(4) (2010)
2. Raimond, Y., Abdallah, S.A., Sandler, M.B., Giasson, F.: The Music Ontology. In: *Proc. 8th International Conference on Music Information Retrieval*, Vienna, Austria, September 23-27 (2007)
3. Cannam, C., Landone, C., Sandler, M.B.: Sonic visualiser: an open source application for viewing, analysing, and annotating music audio files. In: Bimbo, A.D., Chang, S.-F., Smeulders, A.W.M. (eds.) *ACM Multimedia*, pp. 1467–1468. ACM (2010)

4. Fazekas, G., Cannam, C., Sandler, M.B.: Reusable Metadata and Software Components for Automatic Audio Analysis. In: Proc. IEEE/ACM Joint Conference on Digital Libraries (JCDL 2009) Workshop on Integrating Digital Library Content with Computational Tools and Services, Austin, Texas, USA (2009)
5. Page, K.R., Fields, B., Nagel, B.J., O'Neill, G., Roure, D.D., Crawford, T.: Semantics for music researchers: How country is my country? In: Polleres, A., Chen, H. (eds.) ISWC Posters & Demos. CEUR Workshop Proceedings, vol. 658. CEUR-WS.org (2010)

Exploratory Search on the Top of DBpedia Chapters with the Discovery Hub Application

Nicolas Marie^{1,2}, Fabien Gandon¹, Damien Legrand¹, and Myriam Ribière²

¹ INRIA Sophia-Antipolis, Wimmics Team,
2004 Route des Lucioles, Sophia-Antipolis, 06410 BIOT
{nicolas.marie, fabien.gandon, damien.legrand}@inria.fr

² Alcatel-Lucent Bell Labs,
7 Route de Villejust, 91260 NOZAY
{nicolas.marie, myriam.ribiere}@alcatel-lucent.com

Abstract. Discovery Hub is an exploratory search engine that helps users explore topics of interests for learning and leisure purposes. It makes use of a semantic spreading activation algorithm coupled with a sampling technique so that it does not require a preprocessing step.

Keywords: Semantic web, linked data, DBpedia, spreading activation, semantic spreading activation, exploratory search system, discovery engine.

1 Linked Data Based Exploratory Search and Recommendation

Exploratory search [2] systems are designed to assist users during expensive cognitive consuming search tasks such as learning or topic investigation. They provide a high level of assistance during the navigation in the results space and advanced results explanations. In the past few years several works showed the interest of using linked data datasets, and especially DBpedia¹, for resources discovery in recommender and exploratory search systems. For instance Seevl² is a band recommender helping the discovery of musical content and artists on Youtube thanks to a recommendation algorithm and a faceted browsing functionality. MORE³ is a movie recommender in the form of a Facebook application that performs film recommendations thanks to DBpedia. Aemoo⁴ is an exploratory search system that offers a filtered view on the DBpedia graph and gives explanations on the relations between the resources shown to the user. Yovisto⁵ is a video platform offering an exploratory search feature that proposes a ranked list of related topic besides search results.

All these systems match the user query with DBpedia resource(s) and perform the selection, ranking and rendering of related/similar results. They use diverse methods

¹ <http://dbpedia.org>

² <http://seevl.net/>

³ <http://apps.facebook.com/new-more/>

⁴ <http://wit.istc.cnr.it/aemoo>

⁵ <http://www.yovisto.com/>

and depend on a partial or total preprocessing step. As it is a young research area there are many improvements possible:

- No work allows the expression of *composite* queries for exploratory search i.e. interests captured in the form of several resources (“*Claude Monet*” + “*Emile Zola*”). Using linked data to solve such queries gives the possibility to identify complex, indirect, non-trivial paths between the seed resources. It enables the suggestion of results that are at the cross-road of different topics of interest.
- No work deals with the data freshness issue. Indeed, linked data datasets are evolving over the time. The continuous update of the data and its impact on the preprocessing is not addressed in the state-of-the-art.
- No work proposes a lightweight method that is applicable on remote SPARQL endpoints. Indeed the pre-processing phases used in the state of the art are specific to the knowledge base addressed and often requires a local copy of the base to be performed. Consequently the existing systems are often limited to one defined knowledge base.

These limitations are mainly due to the preprocessing step that is strongly conditioning the type and the range of results that the applications are able to retrieve. We explore the potential of an on-the-fly linked data processing for exploratory search purpose. We use the term “*on-the-fly*” to stress that the method does not need any preprocessing to produce the results. It fetches data from the SPARQL endpoint and processes it at runtime.

2 On-the-Fly Semantic Spreading Activation

To reach our objective of an on-the-fly linked data processing we propose a method that is based on a semantic spreading activation coupled with a sampling phase (architecture shown on figure 1). Generally speaking, the spreading activation technique [1] consists in associating a numeric value to the node(s) representing the user’s interest(s) and then spreading this value to the neighborhood iteratively with heuristics depending on the application goal. Our approach is novel as the propagation controlling pattern is a class-based semantic weight that is function of the stimulated origin node. The origin node semantics plays a significant role in the distribution of activation even in “*distant*” parts of the graph:

- When a query is entered a local triple store instance is created. It imports the neighbourhood of the seed node(s) filtered with a class-based semantic pattern. This pattern aims to concentrate the activation on a consistent subset of nodes in order to increase the algorithm relevance. The most prevalent types of the seed’s neighbours are included in the pattern. For each neighbour only its deepest type(s) in the class hierarchy is/are taken in account.
- As the propagation spreads along the iterations the neighbourhoods of the most activated nodes are imported till a limit (maximum number of triples) is reached. The semantic pattern is re-used for the imports during all the process.
- The propagation stops when the maximum number of iterations is reached. The most activated nodes are suggested to the user in decreasing order of activation.

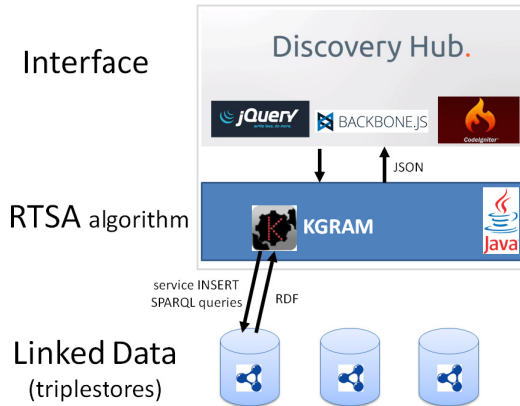


Fig. 1. Discovery Hub architecture

We performed extensive analysis on a large set of queries to understand the behaviour of the algorithm. It helped us to set its main parameters correctly in order to get a fast response time without degrading the results too much. We obtained an average response time of 2031ms with a standard deviation of 1952 ms on a set of 100.000 queries having one node as input i.e. ego-centric queries. During our experiment the sample had a size of 6000 triples (details in [3]). The 100.000 nodes were selected randomly thanks to a random walker. The class-based pattern and the sampling considerably lower the amount of triples needed to compute the results. Thus it is possible to run the algorithm on-the-fly, for instance on public SPARQL endpoints like the localized DBpedia chapters (e.g. Italian, Spanish).

3 The Discovery Hub Web Application

Discovery Hub⁶ is an exploratory search engine that helps its users to explore topics of interest through an interface optimized for exploration. Discovery Hub uses the DBpedia knowledge to render the algorithm results (e.g. label, description, pictures) and organize the results space (e.g. filters, facets). See figure 2.

As the understanding of the results is very important for exploratory search systems Discovery Hub also provides several results explanations functionalities that help the user to understand the relation between the query-resources(s) and the results: one showing the common properties they share, one highlighting their cross-references in Wikipedia, one showing direct and indirect connections in a graph-format (see figure 3). The application proposes also many redirections to tierce platforms to extend the search process. The third-party services are proposed according to the type of the considered result e.g. music service for a *Band* or a tourism platform for a *Museum*. Several demonstration videos are available online⁷.

⁶ <http://semreco.inria.fr>

⁷ <http://semreco.inria.fr/hub/videos/>

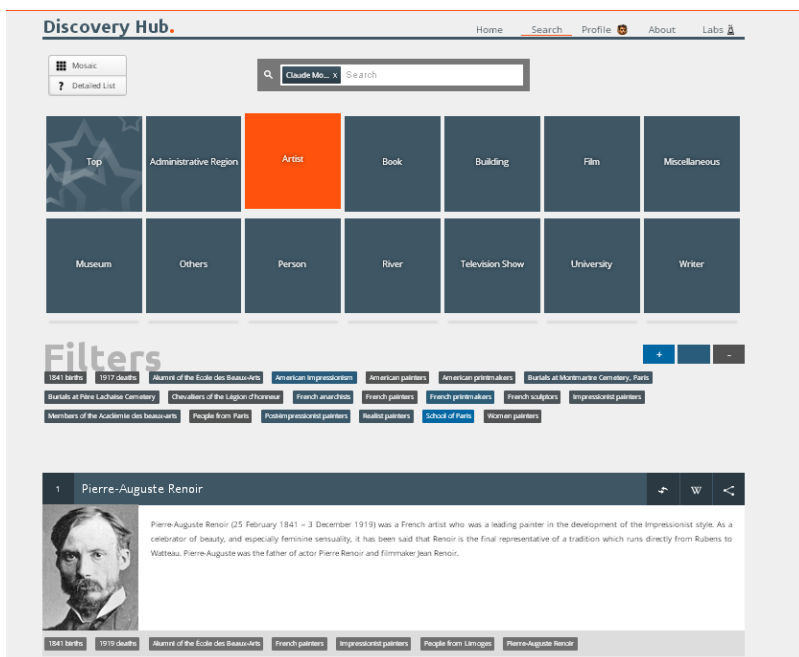


Fig. 2. Discovery Hub results space for the query *Claude Monet*

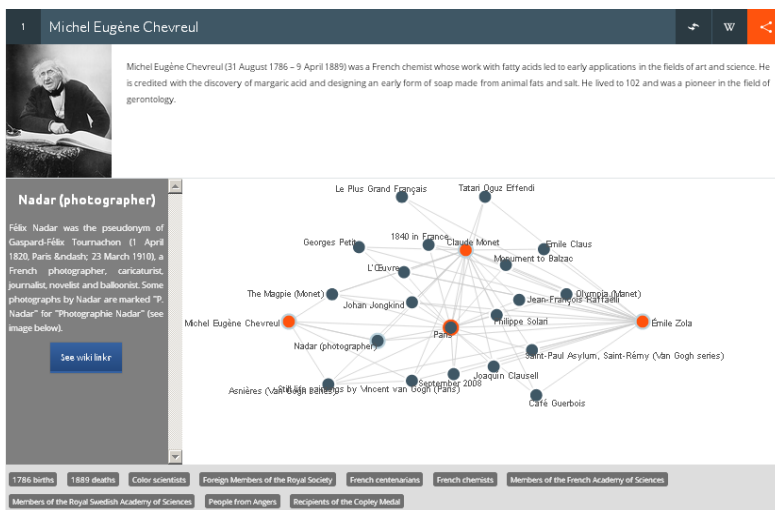


Fig. 3. Explanation unveiling the connections between *Michel Eugène Chevreuil* (scientist) with and two seeds *Claude Monet* (painter) + *Emile Zola* (writer)

The results retrieved by the application were successfully evaluated thanks to a user’s experimentation.

More and more local DBpedia chapters emerge⁸. Thanks to our approach it is very easy to switch from one SPARQL endpoint to another. Today Discovery Hub is able to process Czech, English, French, German, Italian, and Spanish DBpedia data. With the “*internationality*” mode the user can see the level of description of the resource in the various DBpedia chapters. This level of description corresponds to the node degree in the respective DBpedia chapters. This mode helps the user to select the richest data source to perform his query (e.g. use the Italian data for an Italian painter query). This flexibility in the choice of the SPARQL endpoint addressed is an advantage offered by the absence of preprocessing.

On demonstration we will develop an exploratory search scenario centered on the painter *Claude Monet*. We will use the faceted browsing features to explore the results space. As these interface elements convey lot of knowledge about the results, we will show that they can be a source of inspiration during the search process, drive the user in unexpected browsing paths or help him to identify his knowledge gaps. We will give examples of the type-based redirections: Google Art Project⁹ for an *Artist* result and TripAdvisor¹⁰ for a *Museum* one. We will show the explanations features for the result *Camille Pissaro* result and how they can be combined. Then we will showcase an example of polycentric query starting from *Claude Monet* and *Emile Zola* seeds. We want to demonstrate that Discovery Hub is able to identify and retrieve quickly a synthesized view on complex resources connections. The query will be composed with the “*search box*” in which the user can drag and drop resources of interest all along his navigation. Finally we will show the capacity of the application to address remote public SPARQL endpoints by executing the *Claude Monet* query on several localized DBpedia chapters. We will finish by observing the results differences depending on the knowledge base.

4 Conclusion

Discovery Hub prototype implements a semantic sensitive spreading activation algorithm coupled with a sampling technique. It allows processing the linked data on-the-fly i.e. without any pre-processing and offers flexibility in the choice of the SPARQL endpoint and explanation mechanisms.

References

- [1] Crestani, F.: Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review* 11(6), 453–482 (1997)
- [2] Marchionini, G.: Exploratory search: From finding to understanding. *Comm. of the ACM* 49(4) (2006)
- [3] Marie, N., Corby, O., Gandon, F., Ribière, M.: Composite interests’ exploration thanks to on-the-fly linked data spreading activation. In: *Hypertext 2013* (to appear, 2013)

⁸ <http://dbpedia.org/internationalization>

⁹ <http://www.googleartproject.com/>

¹⁰ <http://www.tripadvisor.com/>

Applying SPARQL-DQP for Federated SPARQL Querying over Google Fusion Tables

Freddy Priyatna¹, Carlos Buil Aranda², and Oscar Corcho¹

¹ Ontology Engineering Group, Facultad de Informática, UPM, Spain
`{fpriyatna, ocorcho}@fi.upm.es`

² Department of Computer Science, PUC Chile
`cbuil@ing.puc.cl`

Abstract. Google Fusion Tables (GFT) is a data management, integration and visualization service provided by Google. Users can upload their structured data, integrate it with other people's data, and visualize it on various tools provided, such as Google Maps, charts or graphs. Every GFT table constitutes a data silo that is not commonly linked to other data sources. A way to enable data to be linked and reused is by exposing it as (virtual) RDF dataset (for instance, using R2RML) and to query it using SPARQL. In this work, we present a system that exposes GFT tables as SPARQL endpoints, enabling federated SPARQL queries with data from other SPARQL endpoints.

Keywords: Google Fusion Tables, SPARQL, DQP, R2RML.

1 Introduction

Announced in 2009, Google Fusion Tables [5] is a data management service provided by Google to help users in working with their structured data. Users can upload their CSV data and host in the cloud on Google infrastructure such as Google Bigtable for scalability purposes. Hosting data in the cloud brings several benefits, such as enabling the data to be shared with other users, to be merged with other people's data, and even allowing other users to collaborate by giving read/write permission. Google also integrates various visualisation tools so that users can share their data easily in various forms, such as on maps (using Google Maps), charts, or graphs. While the number of GFT tables users or available tables is not publicly advertised, it is believed that there are 400 millions active GMail users¹, and each of them potentially can use this service and contribute their own tables.

As aforementioned, data hosted in GFT tables can be merged with other's people data. However, this can work only among GFT tables. Seen from outside the service, GFT tables consistute data silos that are not commonly linked to other data sources.

In this work, we present a system that exposes GFT tables as SPARQL endpoints by using R2RML [4], and we use SPARQL-DQP to demonstrate that

¹ <http://www.quora.com/Gmail/How-many-total-active-Gmail-users-are-there>

federated queries can be evaluated over them as if they were RDF datasets, thus taking benefit of the ease and scalability provided by the Google infrastructures while at the same time exploiting the power of querying over RDF enabled sources. The remainder of the paper is as follows: Section 2 presents the architecture of our system and in Section 3 we see how the system works by using some examples. Finally in Section 4 we close the paper by giving the conclusion and the future work.

2 Architecture

The general architecture of the system is shown in Figure 1.

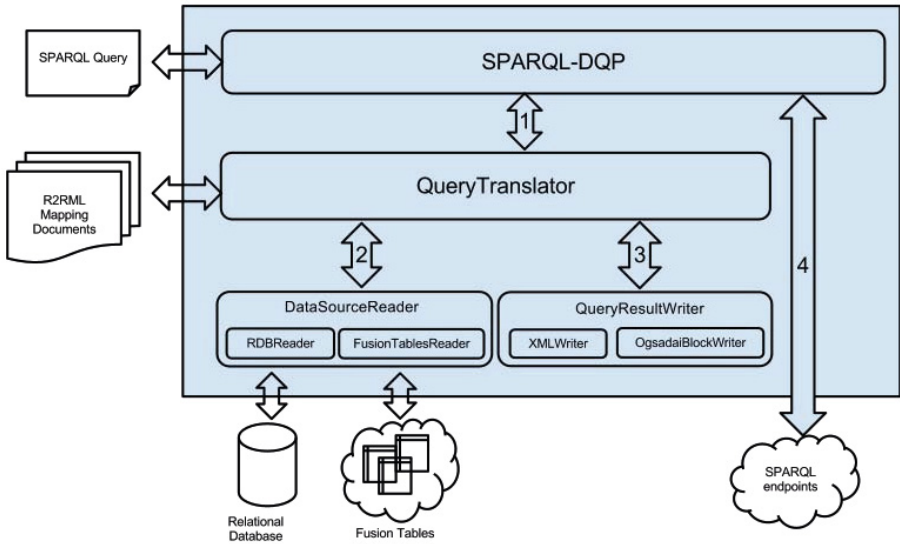


Fig. 1. Architecture of the system

A federated SPARQL query sent by the user is received by SPARQL-DQP [2]. SPARQL-DQP is a SPARQL 1.1 Federated Query system based on OGSA-DAI/DQP system [1]. The SPARQL 1.1 Federated Query specification [6] defines a `SERVICE` keyword as the location of the SPARQL endpoint that corresponds to the part of the query. By using SPARQL-DQP, we enable the SPARQL queries to query data not just from Fusion Tables, but also from available public SPARQL endpoints such as the ones in Linked Open Data (LOD) cloud. The user query is a normal SPARQL 1.1 query in which the `SERVICE` keyword may identify either a remote SPARQL endpoint or the R2RML mapping file that is used in the query translation process. The following processes occur when the system receives a SPARQL query:

- SPARQL-DQP divides the query into queries to be evaluated in each remote endpoints, as explained in [2], and sends each of them to the Query Translator (1) or directly to the corresponding SPARQL endpoint (4).
- The Query Translator translates (2) the SPARQL query from the user into an SQL query using the algorithm defined in [3]. That new SQL query is sent to the Fusion Table service by the Data Source Reader which will also retrieve the results.
- These results will be converted into the internal SPARQL-DQP representation and streamed into the main data workflow (3). This data workflow is managed by SPARQL-DQP [2].

3 Example

We now explain the process when the system receives the SPARQL query “*give me all the members of the Ontology Engineering Group coming from a country whose capital is Madrid*”, shown in Listing 1.1.

Listing 1.1. Example of Federated SPARQL Query

```

1 SELECT ?n ?c
2 WHERE {
3   SERVICE <http://mappingpedia.linkeddata.es/mappings/fusiontables/
4     1pQBGUqR_g-j1WQavu-Fi1wGS7jsdRxomGc0DxMI/oegmembers.ttl>
5     {
6       ?m rdf:type foaf:Person.
7       ?m foaf:name ?n.
8       ?m ex:hasCountry ?c.
9     }
10  SERVICE <http://DBpedia.org/sparql> {
11    ?c <http://dbpedia.org/property/capital> <http://dbpedia.org/resource/Madrid>.
12  }
13 }
```

That query asks for data about the members of the OEG research group (name and country) and also asks for data about Spain. For that, the query contains two **SERVICE** calls, one to the DBpedia SPARQL endpoint (asking about the country resource) and another one to a Google Fusion table containing the data about the members of this research group.

The part of the query that is addressed at the Fusion Table is sent to the Query Translator along with the mapping document specified in the **SERVICE** URI. The Query Translator component translates the SPARQL query into a SQL query using the mapping information specified in the mapping document. The SPARQL and resulting SQL query can be seen in Listing 1.2. That SQL query is then sent to Fusion Table Reader component, which uses Google Fusion Table API to execute the query and process the results which are streamed into the SPARQL-DQP data workflow.

Meanwhile the **SERVICE** call to DBpedia is executed in parallel and SPARQL-DQP will join the results with the ones obtained from the GFT table

query execution. All these remote query executions and data transfers form a single data workflow which is managed by SPARQL-DQP for finally send the results back to the user.

A video showing the example can be seen at this URL : <http://www.youtube.com/watch?v=qrTBJbnTHnc>.

Listing 1.2. Query that is addressed at the GFT table

```

1  -- SPARQL query sent to Query Translator
2  SELECT ?n ?c
3  WHERE {
4    ?m rdf:type foaf:Person.
5    ?m foaf:name ?name.
6    ?m ex:hasCountry ?c.
7  }
8
9  -- SQL query produced by Query Translator
10 SELECT name, country
11 FROM 1pQBGUqR_g-j1WQavu-Fi1wGS7jsdRxomGc0DxMI
12 WHERE name NOT EQUAL TO ""
13 AND country NOT EQUAL TO ""

```

4 Conclusion and Future Work

In this demo we will present a system that combines two features: first, exposing Google Fusion Tables as SPARQL endpoints using R2RML mapping documents in order to map the values from Fusion Tables as virtual RDF datasets; second, querying these virtual RDF datasets together with public SPARQL endpoints by using SPARQL-DQP.

In the current system, the user has to put the mapping document URL manually in the SPARQL query. In the future work, we will store those mapping documents in our triples stores and enable a SPARQL endpoint. In that way, instead of providing the URL of the mapping document, a user can just ask "get me all the mapping documents that map the concept foaf:person".

Acknowledgments. Freddy Priyatna and Oscar Corcho have been supported by the PlanetData (FP7-257641) and myBigData (TIN2010-17060) projects. Carlos Buil-Aranda has been supported by the CONICYT/FONDECYT project number 3130617.

References

1. Nedim Alpdemir, M., Mukherjee, A., Gounaris, A., Paton, N.W., Watson, P., Fernandes, A.A.A., Fitzgerald, D.J.: OGSA-DQP: A service for distributed querying on the grid. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 858–861. Springer, Heidelberg (2004)

2. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and optimization of the SPARQL 1.1 federation extension. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part II*. LNCS, vol. 6644, pp. 1–15. Springer, Heidelberg (2011)
3. Chebotko, A., Lu, S., Fotouhi, F.: Semantics preserving SPARQL-to-SQL translation. *Data & Knowledge Engineering* 68(10), 973–1000 (2009)
4. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language (2012)
5. Gonzalez, H., Halevy, A., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W.: Google fusion tables: data management, integration and collaboration in the cloud. In: *Proceedings of the 1st ACM Symposium on Cloud Computing*, pp. 175–180. ACM (2010)
6. Prud'hommeaux, E., Buil-Aranda, C.: SPARQL 1.1 federated query. World Wide Web Consortium, Proposed Recommendation (November 2012)

Querying Multilingual DBpedia with QAKiS

Elena Cabrio, Julien Cojan, Fabien Gandon, and Amine Hallili

INRIA Sophia Antipolis, France
{firstname.lastname}@inria.fr

Abstract. We present an extension of QAKiS, a system for open domain Question Answering over linked data, that allows to query DBpedia multilingual chapters. Such chapters can contain different information with respect to the English version, e.g. they provide more specificity on certain topics, or fill information gaps. QAKiS exploits the alignment between properties carried out by DBpedia contributors as a mapping from Wikipedia terms to a common ontology, to exploit information coming from DBpedia multilingual chapters, broadening therefore its coverage. For the demo, English, French and German DBpedia chapters are the RDF data sets to be queried using a natural language interface.

1 Question Answering over Linked Data

The Semantic Web community has recently started an effort of internationalization of the DBpedia project (Bizer *et al.* [1]), born to extract structured information from Wikipedia multilingual pages, and to make such structured information accessible on the Web. This demonstration presents an extension of QAKiS, a system for open domain Question Answering over linked data (Cabrio *et al.* [2]), that allows to query DBpedia multilingual chapters. Such chapters, well connected through Wikipedia instance interlinking, can contain different information with respect to the English version. In particular, they can provide *i*) more specificity on certain topics, *ii*) fill information gaps, or *iii*) conceptualize certain relations according to a specific cultural viewpoint. QAKiS exploits the alignment between properties carried out by DBpedia contributors as a mapping from Wikipedia terms to a common ontology, to exploit information coming from DBpedia multilingual chapters. Given the multilingual scenario, attributes are labeled in different natural languages: the common ontology enables to query the multiple DBpedia chapters with the same vocabulary on the mapped data.

The ability to exploit all the amount of multilingual information brings several advantages to QAKiS [3], both considering *i*) the intersection of such resources in different languages to detect contradictions or divergences, and *ii*) the union of such resources, to fill information gap (cross-fertilization among languages) (Rinser *et al.* [4]). Extending QAKiS to query multilingual data sets goes in the direction of enhancing users consumption of semantic data originally produced for a different culture and language, overcoming language barriers¹.

¹ Currently a hot topic, see the Multilingual Question Answering over Linked Data challenge (QALD-3) <http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=home>

2 Extending QAKiS to Query Multilingual DBpedia

QAKiS System Description. QAKiS (Question Answering wiKiFramework-based System) [2] addresses the task of QA over structured knowledge-bases (e.g. DBpedia), where the relevant information is expressed also in unstructured forms (e.g. Wikipedia pages). It implements a relation-based match for question interpretation, to convert the user question into a query language (e.g. SPARQL). More specifically, it makes use of relational patterns (automatically extracted from Wikipedia and collected in the WikiFramework repository [2]), that capture different ways to express a certain relation in a given language. QAKiS is composed of four main modules (Fig. 1): *i*) the **query generator** takes the user question as input, generates the typed questions, and then generates the SPARQL queries from the retrieved patterns; *ii*) the **pattern matcher** takes as input a typed question, and retrieves the patterns (among those in the repository) matching it with the highest similarity; *iii*) the **sparql package** handles the queries to DBpedia; and *iv*) a **Named Entity (NE) Recognizer**.

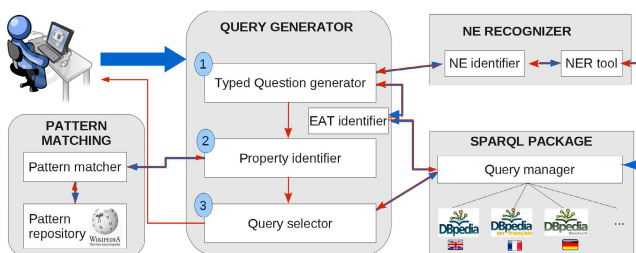


Fig. 1. QAKiS workflow [2]

The actual version of QAKiS targets questions containing a Named Entity related to the answer through one property of the ontology, as *Which river does the Brooklyn Bridge cross?*. Such questions match a single pattern (i.e. one relation).

Before running the *pattern matcher* component, the question target is identified combining the output of Stanford NE Recognizer, with a set of strategies that compare it with the instances labels in the DBpedia ontology. Then a *typed question* is generated by replacing the question keywords (e.g. who, where) and the NE by the types and supertypes. A Word Overlap algorithm is then applied to match such typed questions with the patterns for each relation. A similarity score is provided for each match: the highest represents the most likely relation. A set of patterns is retrieved by the pattern matcher component for each typed question, and sorted by decreasing matching score. For each of them, a set of SPARQL queries is generated and then sent to an endpoint for answer retrieval.

Multilingual DBpedia Alignment. Multilingual DBpedia chapters² have been created following Wikipedia structure: each chapter contains therefore data

² <http://wiki.dbpedia.org/Internationalization/Chapters>

extracted from Wikipedia in the corresponding language, and so reflects local specificity. Data from different DBpedia chapters are connected by several alignments: *i)* *instances* are aligned according to the inter-language links, that are created by Wikipedia editors to relate articles about the same topic in different languages; *ii)* *properties* that mostly come from template attributes, i.e. structured elements that can be included in Wikipedia pages so as to display structured information. These properties are aligned through mappings manually edited by the DBpedia community. Since in the demo we are focusing on English, French and German DBpedia chapters, Figure 2 shows the additional information coverage provided by the mentioned DBpedia chapters. Areas **FR only**, **DE only** and **DE+FR only** correspond to aligned data made available by French and German DBpedia chapters.

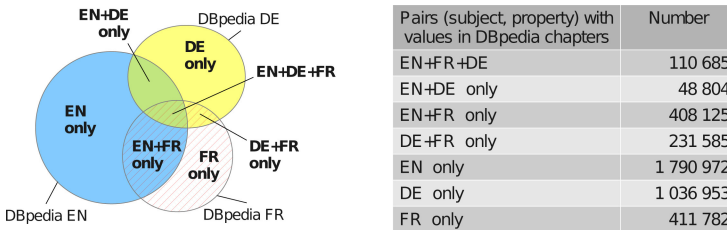


Fig. 2. Relative coverage of English, German and French DBpedia chapters

QAKiS Extension. QAKiS extension to query the ontology properties of multilingual DBpedia is integrated at the *SPARQL package* level. The typed questions generation and the pattern matching steps work as before, but now, instead of sending the query to English DBpedia only, the *Query manager* reformulates the queries and sends them to multiple DBpedia chapters. As only the English chapter contains labels in English, this change has no impact on the NE Recognition. The main difference is in the query selection step. As before, patterns are taken iteratively by decreasing matching score, the generated query is then evaluated and if no results are found the next pattern is considered, and so on. However, as queries are now evaluated on several DBpedia chapters, it is more likely to get results, terminating query selection with a higher matching score. Currently, the results of a SPARQL query are aggregated by the set union. Other strategies could be considered, as using a voting mechanism to select the most frequent answer, or enforcing a priority according to data provenance (e.g. English chapter could be considered as more reliable for questions related to English culture).

3 QAKiS Demonstrator

Figure 3 shows QAKiS demo interface.³ The user can select the DBpedia chapter he wants to query besides English (that must be selected as it is needed for

³ <http://dbpedia.inria.fr/qakis/>

NER), i.e. French or German DBpedia (box 1). Then the user can either write a question or select among a list of examples, and click on *Get Answers!* (box 2). As output, QAKiS provides: *i)* the user question (the recognized NE is linked to its DBpedia page), *ii)* the generated typed question, *iii)* the pattern matched, *iv)* the SPARQL query sent to the DBpedia SPARQL endpoint, and *v)* the answer (box 3). The demo we will present follows these stages for a variety of queries.

The screenshot shows the QAKiS interface. On the left is the logo 'QAKiS Question Answering with Knowledge-based System'. The main interface is divided into several sections:

- Input Section (Box 2):** A text input field containing the question "What is the nationality of Barack Obama?". Below it are "Get answers!" and "Clear" buttons.
- Queried DBpedias (Box 1):** A panel on the right showing flags for English (checked), French (checked), and German (unchecked).
- Pattern Matching Section:**

Your asked: **What is the nationality of Barack Obama ?**

Pattern matching:
typed question: What is the nationality of [Person] ?
had a best match with pattern: nationality [R:Thing] today [D:Person]
with score 7.5
- The query generated is:**

```

select distinct *
where {
  <http://dbpedia.org/resource/Barack_Obama> <http://dbpedia.org/ontology/nationality> ?v .
  OPTIONAL (?v <http://www.w3.org/2000/01/rdf-schema#label> ?l filter (lang(?l)="en"))
} \limit 20
                
```
- Result Section (Box 3):** A table with a green header row:

#	result
1	United States

Fig. 3. QAKiS demo interface

3.1 Queries and Datasets for Demonstration and Evaluation

QALD-2. Since QAKiS currently targets only questions containing a NE related to the answer through one property of the ontology (e.g. *In which military conflicts did Lawrence of Arabia participate?*), we extracted from QALD-2 test set¹ the set of questions corresponding to such criterion (i.e. 32 questions). The discarded questions require either some forms of reasoning (e.g. counting or ordering) on data, aggregation (from datasets different from DBpedia), involve n-relations, or they are boolean questions. We run both QAKiS_{EN} (i.e. the system taking part into the challenge) and QAKiS_{EN+FR} and QAKiS_{EN+DE} (the versions enriched with the French and German DBpedia, respectively) on the reduced set of questions. Since the answer to QALD-2 questions can be retrieved in English DBpedia, we do not expect multilingual QAKiS to improve its performances. On the contrary, we want to verify that QAKiS performances do not decrease (due to the choice of the wrong relation triggered by a different pattern that finds an answer in multilingual DBpedia). Even if often extended QAKiS select different patterns with respect to the original system, the selected relation is the same (except than in one case), meaning that generally performances are not worsen by the addition of multilingual DBpedia chapters.

Multilingual DBpedia. Since we are not aware of any reference list of questions whose answers can be found in French or German DBpedia only, we create

our list to evaluate the improvement in QAKiS’s coverage as follows: *i*) we take the sample of 32 QALD-2 questions; *ii*) we extract the list of triples present in French and German DBpedia only; in each question we substitute the NE with another entity for which the asked relation can be found respectively in the French or German chapters only. For instance, for the QALD-2 question *How tall is Michael Jordan?*, we substitute the NE *Michael Jordan* with the entity *Margaret Simpson*, for which we know that the relation **height** is missing in English DBpedia, but is present in the French chapter. As a result, we obtain the question *How tall is Margaret Simpson?*, that we submit to QAKiS_{EN+FR}. Following the same procedure for German, in *Who developed Skype?* we substituted the NE *Skype* with the entity *IronPython*, obtaining the question *Who developed IronPython?* ⁴ For some properties (e.g. **Governor**, **Battle**), no additional links are provided by the multilingual chapters, so we discarded the questions asking for those relations. QAKiS precision on the new set of questions over French and German DBpedia is in line with QAKiS_{EN} on English DBpedia (~ 50%). This evaluation did not have the goal to show improved performances of the extended version of QAKiS with respect to its precision, but to show that the integration of multilingual DBpedia chapters in the system is easily achievable, and that the expected improvements on its coverage are really promising and worth exploring (see Figure 2). To double-check, we run the same set of questions on QAKiS_{EN}, and in no cases it was able to detect the correct answer, as expected.

4 Future Perspectives

Extensions are planned in several directions, to improve: *i*) the WikiFramework pattern extraction algorithm; *ii*) the question-pattern matching algorithm; *iii*) the system coverage, addressing boolean and n-relation questions. With respect to multilingualism, we plan to *i*) extend QAKiS to query multilingual DBpedia chapters other than the presented ones; *ii*) port QAKiS to multiple languages, i.e. allowing input questions in languages different from English. Moreover, we plan to experiment approaches to automatically extend DBpedia existing alignments.

References

1. Bizer, C., et al.: DBpedia - A crystallization point for the web of data. *Web Semant.* 7(3), 154–165 (2009)
2. Cabrio, E., Cojan, J., Aprosio, A.P., Magnini, B., Lavelli, A., Gandon, F.: QAKiS: An open domain qa system based on relational patterns. In: *Proceedings of the ISWC 2012 Posters and Demonstrations Track*, Boston, US (November 2012)
3. Cojan, J., Cabrio, E., Gandon, F.: Filling the gaps among DBpedia multilingual chapters for question answering. In: *Proceedings of ACM Web Science 2013*, Paris, France (May 2013)
4. Rinser, D., Lange, D., Naumann, F.: Cross-lingual entity matching and infobox alignment in wikipedia. *Information Systems* (2012)

⁴ The obtained set of transformed questions is available online at <http://dbpedia.inria.fr/qakis/>

LDtogo: A Data Querying and Mapping Framework for Linked Data Applications

Niels Ockeloen, Victor de Boer, and Lora Aroyo

Web and Media Group, Department of Computer Science, The Network Institute,
VU University Amsterdam, Amsterdam, The Netherlands
{niels.ockeloen,v.de.boer,lora.aroyo}@vu.nl

Abstract. Despite its rising popularity, using and managing Linked Data remains a challenge for developers of mainstream web and mobile applications. In this demo we present "LDtogo", a framework that makes it easy for application administrators to integrate and maintain Linked Data when building new applications or when re-using existing ones. LDtogo does this by 1) supporting data processing by means of plug-ins, 2) providing an easy-to-use interface to create a customized API wrapper for applications and 3) using only technologies available on common web hosting platforms (e.g. LAMP hosting environments). Its modular structure and support for standards are important properties.

Keywords: RDF, Linked Data, Framework, Data Mapping, WSDL, Web Services, LAMP (Linux, Apache, MySQL, PHP), Shared Hosting.

1 Introduction

Despite the growing possibilities for the creation of Linked Data applications, the integration of Linked Data into applications has not yet become mainstream. For Linked Data applications to become mainstream, we need solutions that make it easier to develop, deploy and maintain Linked Data based applications.

In this demo, we present a framework that facilitates Linked Data based applications for mainstream developers.

2 Related Linked Data Solutions

There are several solutions that aim to increase or improve the usage of Linked Data in applications, such as The Callimachus project [1], the Large Knowledge Collider (LarKC) [2], ARC2 [3], the RAP Toolkit [4], the Linked Data API [5], ActiveRDF [6], Drupal [7] and others. However none of these provide a solutions that a) targets main stream web application developers involved in small or medium size projects; b) is usable and installable alongside the targeted application on a standard LAMP hosting environment; c) is easy to deploy; d) acts as a middle layer between Linked Data sources and the targeted application; e) provides a graphical user interface (GUI) that allows for easy setup and the division of roles; e) provides a configurable API

interface towards the application; f) provides a generic plug-in system. General ‘data mash-up’ solutions such as Yahoo Pipes¹ and especially DERI Pipes² also provide support for Linked Data, however, they are not deployable on standard LAMP hosting environments as a standalone system. Yahoo pipes is offered as a web service while DERI pipes can be installed but requires Tomcat.

3 Description of the LDtogo Framework

We have developed ‘LDtogo’: A framework that facilitates easy data management for Linked Data applications. LDtogo is a (web) application in its own right, which manifests itself as a ‘control panel’ - type application that can be installed on any PHP/MySQL hosting environment, in many cases being the hosting environment of the ‘target application’. By ‘target application’ we mean the application that will be using LDtogo to gather data. The target application can be of several types, e.g. a web application, a mobile app, or a native application that relies on HTTP requests to gather data. An example instance of LDtogo can be found at <http://www.amstertour.com/LDtogo/> with the target application AmsterTour³ at <http://www.amstertour.com>.

Extracting the Data Selection Process. Using LDtogo, the data selection process no longer takes place within the target application, but in the framework. LDtogo has a modular setup, to allow for easy editing of the data selection process. The framework uses SPARQL to select data from remote sources, and has a plug-in system to allow for manipulation of the selected data. The devised queries can be tested immediately from within the framework, and the sequence of queries and plug-ins can easily be edited using a visual ‘plug-in chain’. The target application interacts with the framework using an API interface. The target application can do a ‘data request’, i.e. a request to the framework to provide it with data, optionally providing input parameters. LDtogo will handle the request by selecting data using the chain of queries and plug-ins as setup for the specific request.

Flexibility in Application Maintenance and Reuse. With the data selection process handled by LDtogo, the way in which data is selected can be edited without the need to make changes to the application. This can include simple modifications e.g. to select additional data, as well as more drastic changes; By altering settings in the framework such as the used selection query and/or the used plug-ins, the main application (or modules thereof) can be reused with data from other Linked Data sources. These ‘application modules’ can be shared among users of the framework, while the same holds for plug-ins. To make the distinction between these two terms more clear, an application module is a piece of code that is part of the target application and does something with the data received from the framework (after the complete request has

¹ <http://pipes.yahoo.com/pipes/>

² <http://pipes.deri.org/>

³ A mobile city tour guide application. See [8] for details.

been handled), while a plug-in does something with the data as part of the request handling within the framework. .

Components within LDtogo. LDtogo can be used to specify a collection of settings that together determine how data is selected from the Linked Data Cloud to fulfill the target applications’ data needs. In order to be flexible and allow for re-use, we have chosen a modular setup for LDtogo. The framework consists of ‘Linked Data Sources’ (i.e., sources that provide an accessible endpoint), ‘Queries’ to select data from specified Linked Data Sources, ‘Plug-ins’ to alter selected data, ‘Data Requests’ that contain the settings to define how a specific request should be handled, and ‘Application Modules’ that group together multiple types of requests that can be made by the same target application. In short, these components in LDtogo together define how to transform the request input of data requests done by the target application into the desired request output.

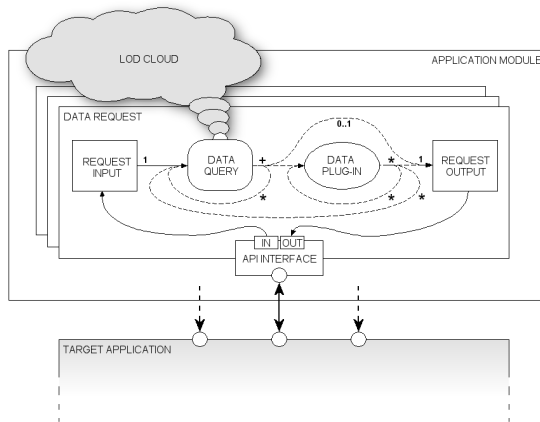


Fig. 1. Conceptual model of LDtogo. An Application Module groups together multiple Data Requests. The ‘Data Chain’ of a Data Request can consist of multiple queries and plug-ins.

Interplay of Components. These components together form the configurable set of tools that allow an application administrator to define and adjust the data selection strategy for an application without the need to modify the target application. The target application uses an API interface provided by the framework to request data needed for its operation. Figure 1 provides an overview of the various components and how they work together to fulfill a data request made by the target application.

The target application can perform a data request by making an API call to the framework. This can either be a HTTP request or a direct function call in PHP. When LDtogo receives a data request, it loads the settings for the particular request and executes the Data Queries and Plug-ins that together form the data chain for that

request. The queries and plug-ins are executed in the order that was specified using the GUI (figure 2) of LDtogo. The end result of the data chain is used as response to the original data request made by the target application.

4 Usage Scenario

An advantage of using the LDtogo GUI is that duties can be split over multiple parties/persons. A developer can focus on developing the core functionality of the application, while a data administrator (e.g. the data owner or someone familiar with SPARQL) can focus on the data selection process. The application developer can assume an API interface as suits him/her best, and together parties can use the GUI interface to map selected data onto the assumed API interface. Many organizations, for example in the cultural heritage sector, now have mature interesting data, but the number of applications is still not overwhelming. Using LDtogo, these organizations can cooperate with any ‘random’ developer to create interesting applications, where the organization fulfills the role of data manager within LDtogo. The development party can specify the desired API interface needed to implement the functionality requested, and together they can create the needed mappings and plug-in chain. Vice versa, a web application developer not familiar with Linked Data can contact a third party with knowledge in this area (possibly the publisher of the data) to assist with the data selection process.

5 Demonstration

During the demonstration we will show the different components within the LDtogo framework, explain their purpose and how they interact with each other. We will demonstrate how to define Linked Data sources, Data Requests and Plug-ins within LDtogo and how to combine them to fulfill a data request made by a target application. To this end, we will use the data mapping solution in the GUI of LDtogo.

We will demonstrate LDtogo in combination with two applications; the aforementioned AmsterTour application and the Agora Touch Demonstrator of the Agora project⁴. Using the AmsterTour application, we will explain the steps involved in fulfilling a data request. We will show how to map the data selected for the application to the defined API interface using the visual data chain. Thereafter we will demonstrate how to alter the data using plug-ins. Along with that we will explain how the plug-in system works.

Next, we will demonstrate how existing applications can be re-used with LDtogo by looking at the Agora Touch demonstrator. We will take a brief look at the API interface of this application, developed before LDtogo was conceived, and use LDtogo to mimic the expected API interface and re-use the Agora Touch demonstrator with Linked Data from other sources.

⁴ <http://agora.cs.vu.nl/>

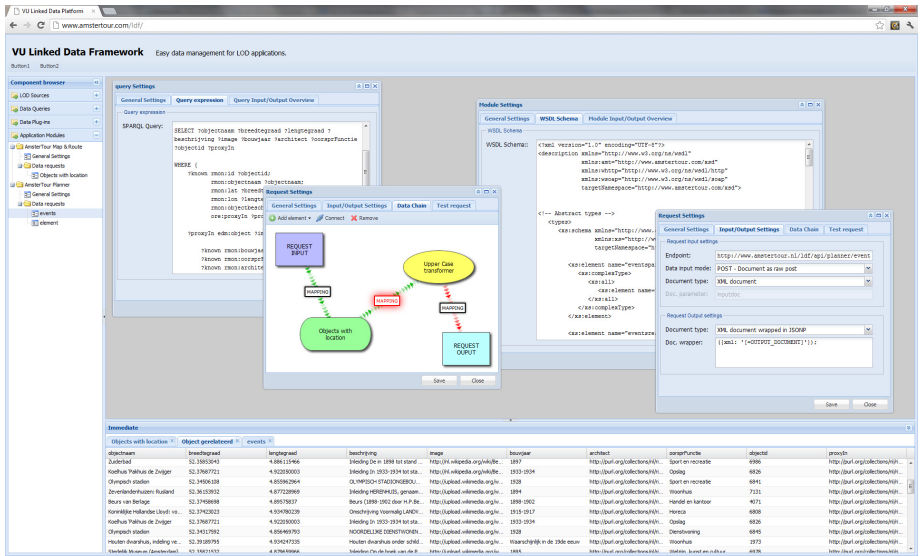


Fig. 2. An overview screenshot of the LDtogo GUI, showing the component browser, the immediate testing panel, and several setting windows for a data request such as the data chain

References

1. The Callimachus Project, <http://www.callimachusproject.org> (accessed on January 15)
2. Fensel, D., Van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Della Valle, E., Fischer, F., et al.: Towards LarKC: A Platform for Web-Scale Reasoning. IEEE International Conference on Semantic Computing 12(12), 524–529 (2008)
3. Nowack, B.: ARC: appmosphere RDF Classes for PHP Developers (2008)
4. Oldakowski, R., Bizer, C., Westphal, D.: RAP: RDF API for PHP. In: Proceedings of the Workshop Scripting for the Semantic Web (2005)
5. The Linked Data API, <http://code.google.com/p/linked-data-api/wiki/Specification> (accessed on January 15)
6. Oren, E., Delbru, R., Gerke, S., Haller, A., Decker, S.: Object-oriented semantic web programming. In: WWW ACM, pp. 817–824 (2007)
7. Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and Consume Linked Data with Drupal! In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 763–778. Springer, Heidelberg (2009)
8. Ockeloen, C.J.: A Data Querying and Mapping Framework for Linked Data Applications. Master Thesis, VU University Amsterdam (2012), http://www.amstertour.com/docs/Master_Thesis.pdf (accessed on January 15)

Exploring the Linked University Data with Visualization Tools

Miika Alonen, Tomi Kauppinen, Osma Suominen, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University, Department of Media Technology
`firstname.lastname@aalto.fi`
`http://www.seco.tkk.fi/`

Abstract. University data is typically stored in separate data silos even though the data is implicitly richly related together. Such data has a large and diverse user base, including faculty members, students, industrial partners, alumnis, collaborating universities, and media. In this paper, we demonstrate two tools for understanding and using the contents of linked university data. The first tool, Visualization Playground (VISU), supports querying and visualizing the data for example for illustrating emerging trends in universities (e.g., about publications) and for comparing differences. The second tool, Vocabulary Visualizer (V^2), demonstrates the usage of shared vocabularies in the Linked University Data Cloud. It reveals what kinds of data different universities have published, and what terms are used to describe the contents. Such analysis is a basis for facilitating design of Linked Data applications across university data boundaries.

1 Towards Linked University Data

Data production and knowledge publication in universities are traditionally based on separate data silos for different data types and domains. Such silos include data such as publication information, course and event descriptions, educational materials, web pages and news feeds. University information systems have traditionally been implemented without considering opening the data stored in there and how it could be done. Another big challenge with separated data silos is the wide diversity of data models and practices in use.

Linked Open Data (LOD) principles and technologies enable universities to publish their legacy data with shared open standards, and offer a variety of approaches for integrating university contents with the existing Web of Data[1]. The promise is that the use of LOD technologies supports academic organizations to be more transparent, comparable, and even more open for new ideas. Linked Universities¹ is a collaboration alliance and application scenario where open datasets from universities are published and linked together using the 5-star methodology². Several universities³ have already published SPARQL endpoints

¹ <http://linkeduniversities.org>

² <http://5stardata.info/>

³ <http://linkeduniversities.org/lu/index.php/datasets-and-endpoints/>

for accessing their contents as Linked Data. These efforts enable applications to use enriched data in novel ways[2,4].

Our Linked Open Aalto (LOA) project⁴ develops a Linked Data infrastructure for the Aalto University. One of the results so far is the Linked Open Aalto Data Service portal⁵, an ongoing collaborative effort with different schools and service providers of Aalto. The bigger goal is that the portal links data from the Aalto University to the other Linked Universities. This would thus include data for example from The Open University⁶, the University of Southampton⁷, the University of Münster⁸ and the University of Bristol⁹. The data in the Linked University Cloud consists of publications, profiles of people, course and event descriptions, educational materials, project information, and service descriptions.

Use cases and advantages of using the Linked University Data have been shown in projects like Lucero [6] and Linked Open Data University of Münster (LODUM) [4] by demonstrating the use of Linked University Data in novel ways. The main focus of the projects so far has been to show the usefulness of linked data within one university. However, the full potential of Linked Open Universities can be achieved by interlinking resources across the universities. Use cases for the web of university data include at least the following:

1. **Linking Scientific Assets.** Connecting publications, patents, projects, people and other scientific assets created by universities and other research communities could foster the reuse of limited resources, bring out new ideas, and facilitate the creation of new projects.
2. **Linking Educational Materials.** Linking related courses, study materials, and other educational assets across universities help students and teachers to find relevant educational material [3].
3. **Finding Funding and Career Offers.** Linking researchers with related opportunities could help in increasing mobility and for building social networks among researchers.
4. **Analyzing the Web of University Data.** Linked Data can be analyzed statistically in order to get an insight of activities, trends, and other phenomena in universities. For these purposes, methods such as combining the R statistical computing environment with SPARQL are available¹⁰.

2 Tools for Exploring the Linked University Data

Shared open vocabularies facilitate the semantic interoperability between datasets. However, this happens only if either the same terms are used in the

⁴ <http://www.seco.tkk.fi/projects/loa/>

⁵ <http://data.aalto.fi>

⁶ <http://data.open.ac.uk>

⁷ <http://data.southampton.ac.uk/>

⁸ <http://data.uni-muenster.de>

⁹ <http://resrev.ilrt.bris.ac.uk/research-revealed-hub/>

¹⁰ See our accompanying contribution to analyze university contents with R <http://linkedscience.org/tutorials/analyzing-and-visualizing-linked-data-from-the-aalto-university-with-r>

datasets or if mappings relate the terms. Well-defined vocabularies and their mappings make it possible to aggregate data from different universities. In practice this is still very challenging without the knowledge of the vocabularies used in the datasets. Our contribution for this problem are two visualization tools: VISUalization playground and Vocabulary Visualizer (V^2). They explicate 1) what datasets and vocabularies are used in separate endpoints 2) and to show similarities and differences between the use of classes and properties between given endpoints.

The tools were implemented using a set of JavaScript libraries¹¹ for visualizing the SPARQL-query results with standard Scalable Vector Graphics (SVG). This enables the processing of the queries on the client side without server-side help other than a proxy. The proxy is in use to overcome cross-domain issues with some of the endpoints.

2.1 Visualization Playground VISU

The visualization playground VISU¹² is a novel SPARQL interface for creating data visualizations. The goal of our work was to create a flexible and easy-to-use tool for exploring and visualizing data from Linked Universities. VISU can aggregate query results from multiple endpoints, manipulate the results, create a visualization and export the data in several formats.

Visualizations are created by using a query that is sent to the selected SPARQL endpoints. The SPARQL-query responses are tabular variable bindings, which are processed in a way similar to Sgvizler[5]. Here VISU utilizes the Google Chart editor that can be used create the visualizations on the fly. Query-reponses from multiple endpoints are processed and the resources using the same identifiers are aggregated in the resulting visualization. For example, the comparison of vocabulary usage (see Figure 1) in separate endpoints can be easily implemented with the playground. Using VISU is further demonstrated in our accompanying tutorial for data exploration and information visualization¹³.

2.2 Vocabulary Visualizer

Designing queries to visualize and explore relationships between multiple endpoints can be very time-consuming. Vocabulary Visualizer¹⁴ (V^2) supports exploring and comparing the vocabulary usage in multiple SPARQL endpoints. V^2 extracts metadata from selected SPARQL endpoints using concurrent SPARQL queries, and then visualizes the joint use of the vocabularies.

V^2 consists of two views. 1) *Class usage view* provides a high level abstraction of used classes in the endpoints. Class definitions that are used by the given

¹¹ D3, see <http://d3js.org/> and Google Charts, see <https://developers.google.com/chart/>

¹² <http://data.aalto.fi/visu>

¹³ <http://linkedscience.org/tutorials/sparql-tutorial-for-data-exploration-and-information-visualization/>

¹⁴ <http://data.aalto.fi/V2>

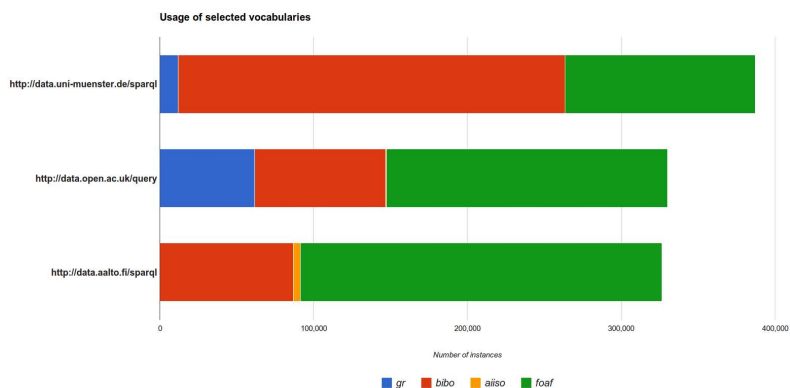


Fig. 1. Vocabulary usage in multiple endpoints

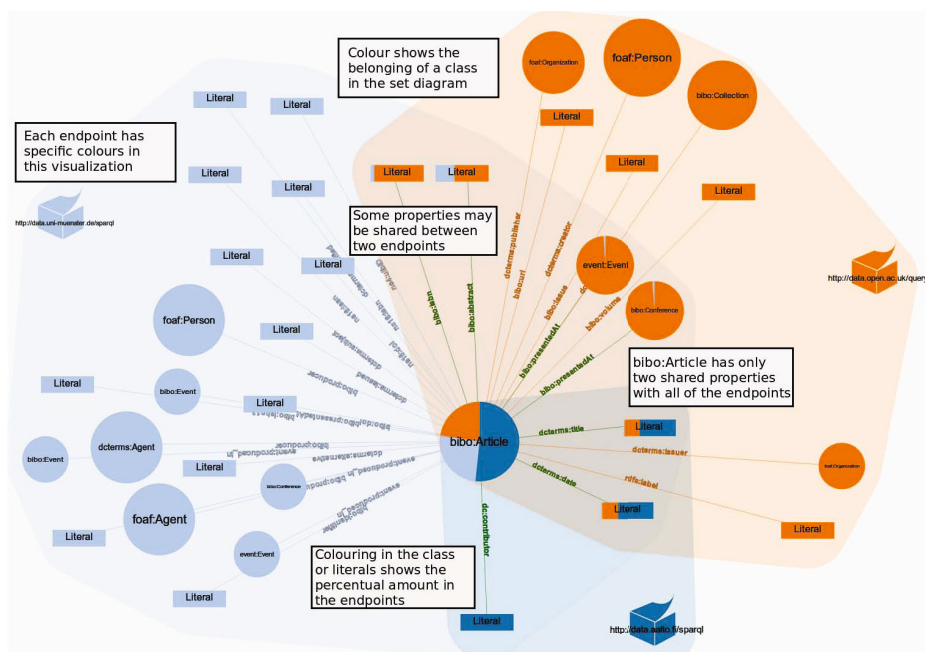


Fig. 2. Usage of properties in separate endpoints

endpoints are visualized using set diagrams and pie charts. In other words, they represent the joint use of classes and the number of class instances. 2) *Property usage view* (see Figure 2) visualizes the usage of properties in the instances of a certain class. The idea is that this analysis supports for designing the use of terms for existing and new datasets and mappings between the existing terms.

3 Conclusions

We argued that the usefulness of the Linked University Data is based on solid standards and reusability of the data by different user groups and universities. A key requirement for this is to use shared vocabularies. The number of vocabularies used in the linked datasets is likely to increase as new data from different domains are being opened. This leads to interoperability problems in linking data across datasets.

The two tools introduced in the paper address these fundamental problems by proving an end-user new means for analyzing vocabulary usage, and for comparing and contrasting Linked Data across different endpoints. 1) VISUalization Playground is an interactive tool for specifying and creating visualizations for exploring and comparing the Linked Data from multiple endpoints. 2) Vocabulary Visualizer (V^2) enables the comparison of Linked Data by revealing the metadata structures, number of instances and the actual usage of vocabularies across the endpoints in novel ways. It supports endpoint developers to take account other relevant datasets and to foster reuse of shared vocabularies to enable better interoperability between the endpoints.

V^2 supports simultaneous comparison with all of the Linked University Endpoints, but with more than three datasets the visualization becomes harder to interpret. Our plan for the future version is to improve V^2 by utilizing ontology alignment for better data aggregation in the visualization and to point out probable term mappings. Another future task is to add functionality to find issues like range and domain violations, as well as OWL-contradictions. The tools presented were developed in the context of Linked University Data. However, the tools and ideas behind them are not domain specific and can thus be adapted to virtually any SPARQL endpoints in the Linked Open Data Cloud.

References

1. Dietze, S., Yu, H.Q., Giordano, D., Kaldoudi, E., Dovrolis, N., Taibi, D.: Linked Education: Interlinking Educational Resources and the Web of Data. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 366–371. ACM (2012)
2. d’Aquin, M.: Putting Linked Data To Use In a Large Higher-Education Organisation. In: Interacting with Linked Data (ILD 2012), p. 9 (2012)
3. Fernandez, M., d’Aquin, M., Motta, E.: Linking Data Across Universities: An Integrated Video Lectures Dataset. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 49–64. Springer, Heidelberg (2011)
4. Kessler, C., Kauppinen, T.: Linked Open Data University of Muenster—Infrastructure and Applications. In: Demos the Extended Semantic Web Conference 2012 (ESWC 2012), Heraklion, Crete, Greece (May 2012)
5. Skjæveland, M.: Sgvizler: A JavaScript Wrapper for Easy Visualization of SPARQL Result Sets. In: Extended Semantic Web Conference 2012 (ESWC 2012), Heraklion, Crete, Greece (May 2012)
6. Zablith, F., Fernandez, M., Rowe, M.: The OU Linked Open Data: Production and Consumption. In: Garcia-Castro, R., Fensel, D., Antoniou, G. (eds.) ESWC 2011. LNCS, vol. 7117, pp. 35–49. Springer, Heidelberg (2012)

Sextant: Browsing and Mapping the Ocean of Linked Geospatial Data^{*}

Charalampos Nikolaou, Kallirroï Dogani,
Kostis Kyzirakos, and Manolis Koubarakis

National and Kapodistrian University of Athens, Greece
{charnik,kallirroï,kkyzir,koubarak}@di.uoa.gr

Abstract. Linked geospatial data has recently received attention as researchers and practitioners have started tapping the wealth of geospatial information available on the Web. With the rapid population of the Web of data with geospatial information, applications to manage it have also started to emerge. What the semantic geospatial web lacks, though, compared to the technological arsenal of the traditional GIS area are the tools that aid researchers and practitioners in making use of this ocean of geospatial data. In this demo paper, we present Sextant, a web-based tool that enables exploration of linked geospatial data as well as creation, sharing, and collaborative editing of thematic maps by combining linked geospatial data and other geospatial information available in standard OGC file formats.

1 Introduction and Motivation

Linked geospatial data has recently received attention as researchers and practitioners have started tapping the wealth of geospatial information available on the Web [4]. As a result, in the last few years, the Web of data is being rapidly populated with geospatial information. Ordnance Survey is the first national mapping agency that has made various kinds of geospatial data from Great Britain available as open linked data¹. Another representative example of such efforts is LinkedGeoData (<http://linkedgeodata.org/>) where OpenStreetMap (OSM) data are made available as RDF and queried using the declarative query language SPARQL [1]. A similar effort is GeoLinked Data (<http://geo.linkeddata.es/>) where geospatial data from Spain is made public using RDF [6]. Finally, the Greek Linked Open Data portal (<http://linkedopendata.gr/>) that is being developed by our group has recently published a number of open datasets for the area of Greece as linked data.

Applications for exploiting this abundance of geospatial information have also started to emerge. In project TELEIOS² we have built a service for real-time fire monitoring using semantic web and linked data technologies [3]. The core

^{*} This work has been funded by the FP7 project TELEIOS (257662).

¹ <http://www.ordnancesurvey.co.uk/oswebsite/products/os-opendata.html>

² <http://earthobservatory.eu/>

component of this service is the geospatial RDF store Strabon [5] which stores hotspot products extracted from satellite images and linked geospatial data offered by the Greek Linked Open Data portal mentioned previously and queries it to deliver the service. Strabon supports two query languages: stSPARQL, an extension of SPARQL 1.1 for querying linked geospatial data that changes over time [5] and GeoSPARQL [7], a recent OGC standard for static geospatial data.

What the semantic geospatial web lacks, though, compared to the technological arsenal of the traditional GIS area are the tools that aid researchers and practitioners in making use of this ocean of geospatial data. Although tools for exploring the content of linked geospatial data have made their appearance recently, these tools focus on browsing a single dataset or the content of a single SPARQL endpoint. The LinkedGeoData browser³ for example, being the most promising among these tools, offers browsing functionality for OSM data and also editing capabilities in the style of the original OSM portal. Map4RDF⁴ is another browser for RDF data with geospatial information which also supports the visualization of statistical data modeled according to the SCOVO vocabulary. Although such tools are very useful for exploring the data offered by a single SPARQL endpoint, they present severe limitations when they are faced with the task of exploring the linked geospatial data cloud. This problem has been tried to be eliminated in its general form (not only in the geospatial domain) by the recent tool LODVisualization⁵ which is based on the Linked Data Visualization Model [2] for visualizing RDF data. Although LODVisualization seems a very promising tool, it has very limited support regarding visualization of geospatial data and construction of meaningful thematic maps.

With the aim of filling this gap and going beyond data exploration to map creation and sharing, we designed and developed Sextant⁶. Similar to well-known GIS tools (e.g., ArcGIS, QGIS), Sextant can be used to produce thematic maps by layering geospatial information which exists in a number of data sources ranging from standard SPARQL endpoints, to SPARQL endpoints following OGC standards for the modeling and querying of geospatial information (i.e., GeoSPARQL), or even other standards or well-adopted geospatial file formats, such as KML and GeoJSON. The feature that distinguishes Sextant from other semantic web or GIS tools is that map creation and sharing, as well as exploration of data can be done in a declarative way using the query languages stSPARQL or GeoSPARQL. In this sense, Sextant is able to create useful thematic maps by layering information coming from the evaluation of SPARQL queries.

2 Sextant Overview

Figure 1a gives a high-level overview of Sextant. The two main functionalities of Sextant can be summarized as follows:

³ <http://linkedgeoata.org/LGD%20Browser>

⁴ <http://oeg-dev.dia.fi.upm.es/map4rdf/>

⁵ <http://lodvisualization.appspot.com/>

⁶ <http://wikipedia.org/wiki/Sextant>

- Exploration of linked geospatial data spanning multiple SPARQL endpoints.
- Creation, sharing, and collaborative editing of thematic maps by combining linked geospatial data and other geospatial information available in vector or raster formats, such as KML, GeoJSON, and GeoTIFF.

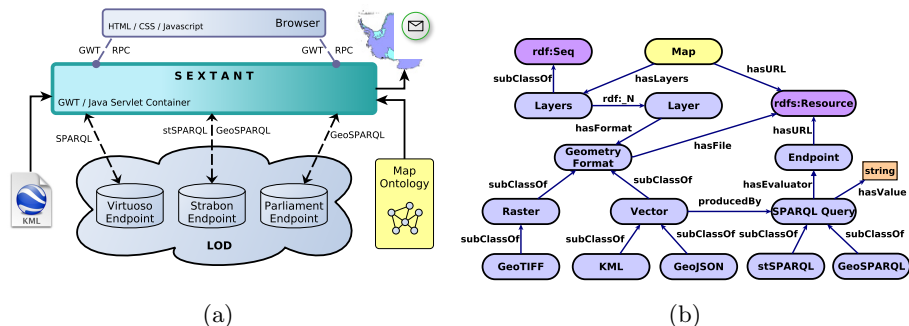


Fig. 1. (a) Sextant overview and (b) Map ontology

Sextant delivers the above functionality by modeling the content of a map according to the Map ontology of Figure 1b. In terms of exploration, Sextant harvests the content of a SPARQL endpoint to build a class hierarchy and discover the spatial extent of available information (assuming that the endpoint supports either stSPARQL or GeoSPARQL). According to the Map ontology, a map comprises an ordered set of layers the content of which may derive either from the evaluation of an stSPARQL/GeoSPARQL query on a SPARQL endpoint or directly from a standard file format for representing geospatial information, such as KML. A layer may also derive from the evaluation of standard SPARQL queries using other vocabularies such as the W3C GeoXG⁷⁸ and Neo-Geo (<http://geovocab.org/>). In such cases, however, one has to provide an adapter for transforming the representation of a geometry to a standard OGC representation format, such as Well-Known Text or GML, the two OGC geospatial data formats supported by stSPARQL and GeoSPARQL. Population of the Map ontology results in the production of a map as a web resource which can be shared with others for collaborative editing and viewing in Sextant. Collaboration can be achieved also using well-known desktop or web-based tools (QGIS, ArcGIS, Google Maps, OpenLayers) by leveraging the export facility of Sextant.

The main design goals in the development of Sextant have been flexibility, portability, and interoperability. Sextant achieves these goals based on the following technologies. It has been developed in the Google Web Toolkit framework⁹ which is a mature framework that provides rapid cross-platform and

⁷ <http://www.w3.org/2005/Incubator/geo/XGR-geo/>

⁸ The W3C GeoXG incubator group has also developed a geospatial ontology available at <http://www.w3.org/2005/Incubator/geo/XGR-geo-ont/>

⁹ <https://developers.google.com/web-toolkit/>

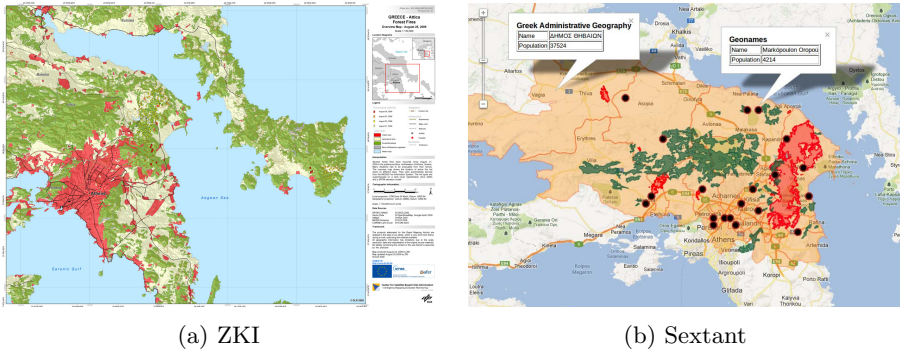


Fig. 2. Fire products of 2009 in the prefecture Attica, Athens, Greece

cross-browser implementations of web applications. It is also based on OGC standards for the representation and querying of geospatial information (GeoSPARQL, KML, WKT), thus making it easily interoperable with well-established and mature GIS applications.

3 Demonstration Overview

To demonstrate Sextant, we will present how one could explore and then create a map using linked geospatial data spanning multiple SPARQL endpoints and KML files. The demonstration will be based on a real user scenario in which an expert of an emergency response agency like the Center of Satellite Based Crisis Information¹⁰ (ZKI) or the National Observatory of Athens¹¹ (NOA) would like to quickly compile a map in response to an emergency event, for example a flood, a tsunami, an earthquake, a forest fire. A typical map product that ZKI produced on August 25, 2009 for assessing the area burnt as a result of the fires of the period from August 21 to August 24, 2009 in Attica, Greece is presented in Figure 2a and may be found at <http://www.zki.dlr.de/map/1034/>. Such a map, apart from depicting burnt areas, it is annotated with other kinds of information to aid experts in assessing the severity of the fire event. Typical information includes the road network, land use/cover, toponyms, population, and other more detailed information concerning map production and the data sources that were used.

We will use Sextant to produce the map presented in Figure 2b that is similar to the one produced by ZKI above. The major difference between the two maps is that the one produced by Sextant is based solely on Linked Open Geospatial Data that are publicly available on the web. This will allow an emergency response manager to quickly compile a map based on “open source intelligence” until more precise, detailed data becomes available, i.e., after contacting local

¹⁰ <http://www.zki.dlr.de/>

¹¹ <http://www.noa.gr/indexen.html>

authorities. Sextant will communicate with three endpoints on top of well-known spatially-enabled RDF stores (Strabon, Parliament, and Virtuoso) and also KML files that are publicly available on the web to get most of the information mentioned above. To demonstrate the collaborative editing capabilities of Sextant, we will share the URI corresponding to the produced map with another user who will be able to edit the map. The changes will be reflected back to the creator of the map.

4 Conclusions and Future Work

In this work, we presented Sextant, a web-based tool that offers functionalities which are of fundamental importance to leveraging linked geospatial data: a) exploration of linked geospatial data that span across multiple SPARQL endpoints, b) creation, sharing, and collaborative editing of thematic maps produced by querying the linked geospatial data cloud and other geospatial data sources available in OGC standard file formats, c) export of maps in OGC standard formats for viewing and editing using other GIS applications.

In the future, we plan to extend Sextant in two directions. First, we will enable the creation of a single layer of a map by combining geospatial information present in different SPARQL endpoints. Such a feature would turn Sextant into a very powerful and desirable tool for leveraging the linked geospatial data cloud. In principle, such functionality could have been easily realized by the existence of both a *spatially-enabled* and *federated* RDF store. Second, we will extend map rendering, which currently uses Google Maps, to use the more versatile OpenLayers project.

References

1. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: Adding a Spatial Dimension to the Web of Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009)
2. Fernéandez, J.M.B., Auer, S., Garcia, R.: The linked data visualization model. In: International Semantic Web Conference (Posters & Demos) (2012)
3. Koubarakis, M., et al.: Real-time wildfire monitoring using scientific database and linked data technologies. In: The 16th International Conference on Extending Database Technology (EDBT 2013), Genoa, Italy, March 18–22 (2013)
4. Koubarakis, M., Karpathiotakis, M., Kyzirakos, K., Nikolaou, C., Sioutis, M.: Data Models and Query Languages for Linked Geospatial Data. Invited papers from 8th Reasoning Web Summer School (2012)
5. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: A Semantic Geospatial DBMS. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 295–311. Springer, Heidelberg (2012)
6. de León, A., Saquicela, V., Vilches, L.M., Villazón-Terrazas, B., Priyatna, F., Corcho, O.: Geographical Linked Data: a Spanish Use Case. In: I-SEMANTICS. ACM (2010)
7. Open Geospatial Consortium: OGC GeoSPARQL - A geographic query language for RDF data. OGC[®] Implementation Standard (2012)

A Distributional Semantic Search Infrastructure for Linked Dataspaces

André Freitas, Seán O’Riain, and Edward Curry

Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway

Abstract. This paper describes and demonstrates a distributional semantic search service infrastructure for Linked Dataspaces. The center of the approach relies on the use of a distributional semantics infrastructure to provide semantic search and query services over data for users and applications, improving data accessibility over the Dataspace. By accessing the services through a REST API, users can semantically index and search over data using the distributional semantic knowledge embedded in the reference corpus. The use of distributional semantic models, which rely on the automatic extraction from large corpora, supports a comprehensive and approximative semantic matching mechanism with a low associated adaptation cost for the inclusion of new data sources.

Keywords: Distributional Semantics, Semantic Matching, Semantic Search, Explicit Semantic Analysis, Dataspaces, Linked Data.

1 Motivation

Within the realm of the Web and of Big Data, dataspace where data is more complex, sparse and heterogeneous are becoming more common. Consuming this data demands applications and search/query mechanisms with the semantic flexibility necessary to cope with the semantic/vocabulary gap between users, different applications and data sources within the dataspace. Traditionally, consuming structured data demands that users, applications and databases share the same vocabulary before data consumption, where the semantic matching process is done manually. As dataspace grow in complexity, the ability to semantically search over data using one’s own vocabulary becomes a fundamental functionality for dataspace.

Distributional semantics [1, 2] applied to semantic search can become a fundamental element in enabling semantic search and semantic matching in dataspace, by providing a comprehensive and low-cost semantic model based on unstructured data.

This paper demonstrates a distributional semantics service infrastructure for Linked Dataspace. The demonstration is exemplified over a question answering and a vocabulary search scenarios using DBpedia as the data source. In addition to these scenarios, distributional semantic search over dataspace can be used for tasks such as ontology alignment, service matching, content recommendation, entity disambiguation, among others.

2 Distributional Semantic Infrastructure

Distributional semantics is defined upon the assumption that the context surrounding a given word in a text provides important information about its meaning [2]. It focuses on the construction of a simplified semantic model for a word based on the statistical distribution of co-occurring words in large text collections. These semantic models are naturally represented by Vector Space Models (VSMs) [2], where the meaning of a word can be defined by a weighted vector over a term or concept space, which represents the association pattern of co-occurring words in a reference corpora. The existence of large amounts of unstructured text on the Web brings the potential to create comprehensive distributional semantic models (DSMs). DSMs can be automatically built from large corpora, not requiring manual intervention on the creation of the semantic model. These characteristics facilitates the creation of DSMs for multiple languages and can provide a more scalable solution to the problem of capturing large-scale commonsense semantic information, necessary for providing a flexible semantic model that could bridge the semantic gap between users, applications and data.

The suitability of the application of distributional semantics to semantic search is empirically supported [1, 2]. Distributional indexes and vector space abstractions (T-Space) specific for Entity-Attribute-Value (EAV) and graph databases were also formulated [2]. These recent developments bring the potential for distributional semantic models and distributional data indexes to become recurrent infrastructure elements for dataspace. This paper demonstrates different services provided by a distributional semantic infrastructure inside the *Treo* framework, where a set of semantic operations over dataspace are used to provide a natural language/semantic search interface over DBpedia.

3 High-Level Architecture Components

Figure 1 shows the high-level architecture components of the *Treo* distributional semantic infrastructure. The functionalities of the components are described below.

- **Distributional-Compositional Semantic Index:** Inverted index which implements the T-Space structured vector space model (VSM) [2]. The T-Space VSM allows a principled distributional-compositional semantic representation of labelled data graphs, supporting search operations over the data graphs with semantic approximation.
- **Explicit Semantic Analysis (ESA):** Provides the distributional concept vectors based on the TF/IDF indexing of the reference corpora (in this example English Wikipedia). Each word present in the graph is represented in the index by a weighted concept vector based on the ESA semantic interpretation approach.
- **Distributional Search:** Distributional search operations allow a semantic approximation/matching between the query terms and the graph triple labels

(*subject*, *predicate* and *object*). Different search patterns can be employed over the data graph. In the natural language query mechanism the query is translated into a set of graph search operations on the distributional index.

- **Instance Search:** Keyword-based instance search which ranks instances according to instance cardinality, string similarity and uses disambiguation/sameAs links when possible (non-distributional search).
- **Distributional Indexer:** Indexes a graph with a labelled graph/EAV representation over the distributional-compositional index (T-Space).
- **Semantic Relatedness:** Computes the ESA semantic relatedness measure between two words. Distributional semantic relatedness measures can be used in tasks such as entity disambiguation, ontology/schema alignment, among others.

The core service components can be accessed by third-party applications through a RESTful web API.

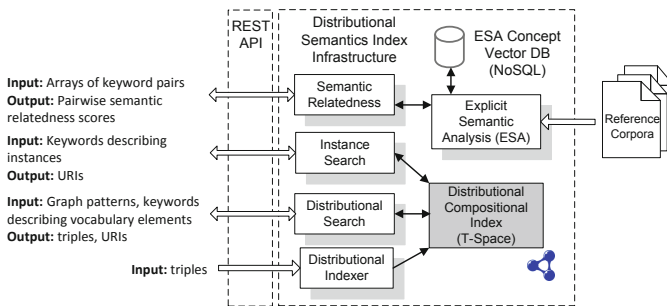


Fig. 1. High-level components of the *Treo* distributional semantics infrastructure

4 Demonstration

In this demonstration the core services available on the distributional semantic infrastructure are used by the components of the natural language interface (NLI) of the *Treo* query engine. Two query types are supported: (i) free natural language queries over graph data (e.g. ‘*How tall is Claudia Schiffer?*’) or (ii) vocabulary queries over graph elements (e.g. *give me all predicates related to the concept ‘geology’*). The second query type is fundamental for tasks such as exploratory dataset search, ontology alignment and matching as well as in the search and reuse existing schema elements. Both queries answer sets are displayed in Figure 2.

The *distributional indexer* service was used to index DBpedia 3.7. Taking the natural language query ‘*How tall is Claudia Schiffer?*’, the *Treo* NLI system parses the natural language query, transforming it into a triple-like representation. To determine the core entity of the query, the entity resolution module uses

the *instance search* and the *distributional search* services to search for instances and classes that match in the dataset. The named entity ‘*Claudia Schiffer*’ is defined as the pivot query entity and mapped to the URI *dbpedia:Claudia_Schiffer*. The query triple representation *dbpedia:Claudia_Schiffer - tall* is then determined and this query pattern is sent to the *distributional search* service for the predicate semantic matching. Using the distributional representation of the indexed DBpedia graph, the query term ‘*tall*’ is matched against *dbpedia-owl:height* predicate for Claudia Schiffer and the triple pattern *dbpedia:Claudia_Schiffer - dbpedia-owl:height* is sent as a graph pattern to the *distributional search* service, which returns the result in Figure 2. In the second example, the user wants the predicates which are related to the *geology* domain inside DBpedia. The keyword ‘*geology*’ is sent to the *distributional search* service which returns a list of semantically related predicates URIs (Figure 2).

The examples show how the distributional infrastructure uses the semantic knowledge extracted from the reference corpora to semantically search and rank datasets’ elements. No additional ontological information is used for the semantic matching. Distributional search queries can be translated as the computation and ranking of semantic relatedness scores between query terms and dataset elements. The reader can find additional query examples in the website¹ and additional information on distributional semantic search, including the evaluation under the Question Answering over Linked Data test collection² in [1, 2].

5 Related Applications

PowerMap [3] is a hybrid matching algorithm which includes terminological and structural schema matching techniques with the assistance of large scale ontological and lexical resources. PowerMap uses WordNet based similarity approaches as a semantic approximation strategy. PowerAqua, a question answering system for Linked Data, uses PowerMap to match query terms to vocabulary terms. The Semantic Web Search Engine (SWSE) [5] is a search and query service that implements an architecture with components for crawling, integrating, indexing, querying, and navigating over multiple data sources. The systems main components include query processing, ranking, an index manager, and an internal data store. Sindice [4] is a search and query service for the Linked Data Web that ranks entities according to the incidence of keywords associated with them. It uses a node-labelled tree model to represent the relationship between datasets, entities, attributes, and values. Similarly to SWSE, Sindice provides a comprehensive entity-centric search and query infrastructure. Compared to PowerMap, the distributional infrastructure for Treo concentrates its semantic matching strategy on DSMs, implementing the DSM as a distributional-compositional (T-Space) semantic index. Compared to SWSE and Sindice, the *Treo* semantic index focuses on the problem of the semantic matching strategies, while SWSE and Sindice does not focus on the vocabulary problem.

¹ <http://treo.deri.ie/eswcdemo>

² <http://www.sc.cit-ec.uni-bielefeld.de/qald-1>

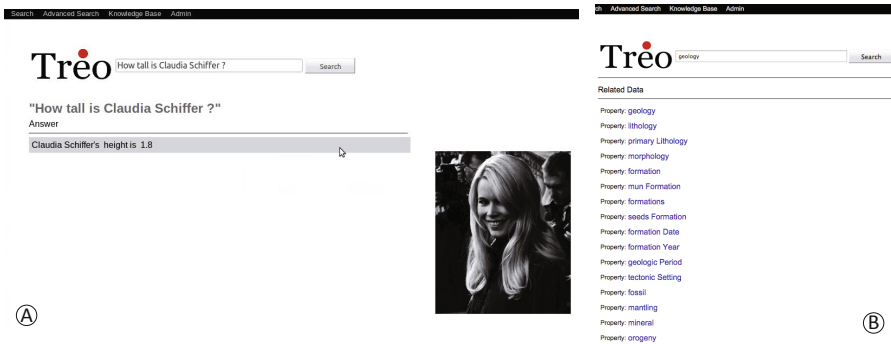


Fig. 2. Answer sets for the query examples: (A) Natural language query over the DBpedia data graph and (B) Keyword-based semantic search over the dataset terminological level

6 Conclusions and Future Work

This work demonstrates the Treo distributional semantic infrastructure for Linked Dataspaces using natural language queries and terminological search operations over DBpedia. Future work will concentrate on improvements over the distributional semantic model.

References

1. Freitas, A., Curry, E., O’Riain, S.: A Distributional Approach for Terminological Semantic Search on the Linked Data Web. In: Proc. of the 27th ACM Symposium on Applied Computing (SAC), Semantic Web and Applications, SWA (2012)
2. Freitas, A., Curry, E., Oliveira, J.G., O’Riain, S.: A Distributional Structured Semantic Space for Querying RDF Graph Data. *International Journal of Semantic Computing, IJSC* (2012)
3. Lopez, V., Sabou, M., Motta, E.: PowerMap: Mapping the Real Semantic Web on the Fly. 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. 414–427. Springer, Heidelberg (2006)
4. Delbru, R., Campinas, S., Tummarello, G.: Searching Web Data: An Entity Retrieval and High-Performance Indexing Model. *J. Web Semantics* (2011)
5. Hogan, A., et al.: Searching and Browsing Linked Data with SWSE: The Semantic Web Search Engine. *J. Web Semantics* (2011)

XSPARQL-Viz: A Mashup-Based Visual Query Editor for XSPARQL^{*}

Syed Zeeshan Haider Gillani, Muhammad Intizar Ali, and Alessandra Mileo

DERI, National University of Ireland, Galway, Ireland
{syed.gillani, ali.intizar, alessandra.mileo}@deri.org

Abstract. XSPARQL is a query language which facilitates query, integration and transformation between XML and RDF data formats. Although XSPARQL supports semantic data integration by providing uniform access over XML and RDF, but it requires users to be familiar with both of its underlying query languages (e.g XQuery and SPARQL). In this system demo, we show how mashup-based techniques can be used for auto generation and execution of XSPARQL queries. XSPARQL-Viz provides an easy to use *drag and drop* visual query editor, which supports novice users in designing complex mappings between XML and RDF and based on these mappings auto generates and executes XSPARQL queries. Results can also be visualised as a graph, table or list.

1 Introduction

XSPARQL is a query language that combines XQuery and SPARQL for mapping, querying and exchanging XML and RDF on the Web [1]. Despite both XQuery and SPARQL are widely adopted for XML and RDF data respectively, many modern data integration applications require interoperability and simultaneous access over both data formats. XSPARQL bridges this gap by providing uniform access over XML and RDF data, enabling query and transformation from one data format to the other using a single query. Figure 1 depicts a simplified version of the XSPARQL architecture.

Different approaches for the integration of XML and RDF data have been proposed in the literature [6,3]. However all proposed solutions, including XSPARQL, require the user to be familiar with both XQuery and SPARQL. Complex query language syntax is the main hindrance in the wide adoption of XSPARQL, because majority of users are domain experts of their respective field but lack expert knowledge of both query languages. Query-by-example, query-by-form, graphical user interface and assisted query editors have been proposed and used for auto query generation of SQL, XQuery and SPARQL. Similarly, mashups are web based ad hoc applications that consume the available data from third parties and combine them to build a new application. However mashups

^{*} This work has been funded by Science Foundation Ireland, Grant No. SFI/08/CE/11380 (Lion2).

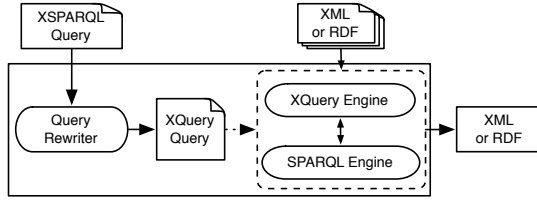


Fig. 1. XSPARQL Architecture[1]

mostly consume only web feeds or API's , hence, lacking the query language capabilities. Concept of data mashups has been utilised for auto query generation of SPARQL and for integration of XML and RDF [7,4]. Following the same principles, we design XSPARQL-Viz (a mashup-based visual query editor) for auto generation of XSPARQL queries. We believe that potential impact, adoption and usability of XSPARQL can be highly increased by providing such support for novice users to write XSPARQL queries, and our proposed Mashup-based XSPARQL visual query editor represents a step forward in this direction.

In this paper, we demonstrate a system demo which provides (i) *auto generation of the XSD schema for XML data sets and RDFS schema for RDF data sets*, (ii) *a visual query editor for facilitating mapping between XML and RDF data sets* (iii) *mashup-based approach for auto generation of XSPARQL queries*, and (iv) *transformation of query results into any desired output format or as input for another query*.

2 System Overview

Our system is written in Java using the Spring Framework. A request based Model-View-Controller (MVC) Framework is built over Spring Inversion of Control (IoC) Framework, which provides layered architecture for a strict separation between model, view and controller. The underlying layered services can be categorised into three classes (i) *XML Schema Generation Services*, (ii) *RDF Schema Generation Services*, (iii) *Query Generation Services*, and (iv) *Result Generation Services*.

The XSPARQL engine is used as a core engine for query and transformation between XML and RDF data, while XSPARQL query engine utilises Saxon¹ query engine for XQuery and ARQ² query engine for SPARQL queries execution. The visual query editor is a web application accessible using any web browser, which is designed using a combination of Jsplumb and JQuery.

¹ <http://www.saxonica.com>

² <http://jena.apache.org/documentation/query/index.html>

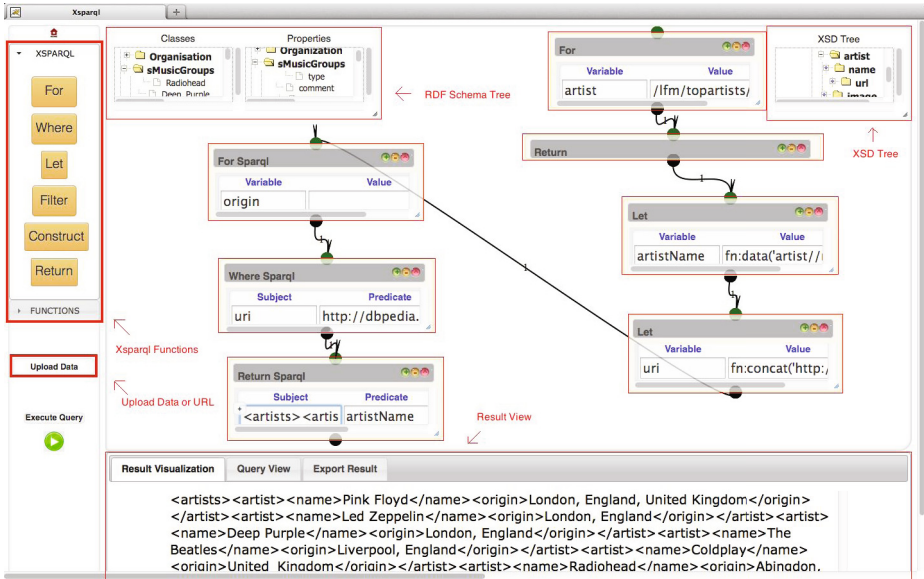


Fig. 2. XSPARQL Graphical User Interface

3 Demonstration Scenario

Inspired by [9], we consider the usecase of integrating Last.fm³ music data with DBpedia⁴. Consider the scenario of a user, who requires to query about the hometown of her top artists. Last.fm contains music information about a particular user e.g type of music or list of top artists listened by the user, while DBpedia extracts structured data from Wikipedia and stores them in RDF format. Last.fm provides access to its data using public api's or web services and returns the results in XML format, while DBpedia is accessible through SPARQL endpoint. Hence, in order to facilitate this query, we need to deal with data heterogeneity and here is where XSPARQL comes handy, and assist us to retrieve, integrate and transform XML and RDF data within a single query.

Listing 1 provides an XSPARQL query to get the desired answers. However, in order to write such XSPARQL query, the user requires to know both of its underlying query languages. We will employ this use case scenario to depict several components of XSPARQL-Viz and generate mashups for the execution of the XSPARQL query. Figure 2 demonstrates the mashups for the generation and execution of XSPARQL query shown in Listing 1. The live version of the demo can be found at <http://xsparql.deri.org/XSPARQL-Viz>

Uploading Data Sets: XSPARQL-Viz requires access to the data where the answer to our query lies. Lower left corner of the XSPARQL-Viz Editor as shown

³ <http://last.fm>

⁴ <http://dbpedia.org>

```

prefix dbprop: <http://dbpedia.org/property/>
let $doc := "http://ws.audioscrobbler.com/2.0/?method=user.gettopartists"
for $artist in doc($doc)//artist
return
let $artistName := fn:data($artist//name)
let $suri := fn:concat("http://dbpedia.org/resource/", $artistName)
for $origin from $suri
where { [] dbprop:origin $origin }
return
<artists>
  <artist>
    <name>{$artistname}</name>
    <origin>{$origin}</origin>
  </artist>
</artists>

```

Listing 1. A Sample XSPARQL Query

in Figure 2, allows user to register the data sources by either uploading an XML or RDF data set from local directory, or by providing a URI of a remotely available data set. Users can upload multiple data sets in one session which are temporarily stored for each user's session.

Schema Generations: Once data sets are uploaded or their URI's are provided, Schema Builder will execute multiple queries to automatically generate XML Schema for XML data sets and RDF Schema for RDF data sets. The generated schema will be displayed in tree format at the top center of the XSPARQL-Viz Editor, as shown in Figure 2. The generated tree structure component provides also an option to change the view into different visual structures, including lists, grids and graphs.

Query Editor: XSPARQL-Viz Editor provides a richer graphical front-end on top of the XSPARQL engine. This front-end enables users not only to execute custom queries but also to have drag-and-drop support from schema tree for efficiently designing the complex mapping between XML and RDF data sets. The left panel of the XSPARQL-Viz Editor in Figure 2 allows users to choose any of the XSPARQL clauses/statements to be applied over the data sets. In order to set variable values in the query, users can either drag and drop any data value from a generated schema tree or provide direct input. The flow of the query is maintained by combining various components of XSPARQL clauses. XSPARQL-Viz also provides a separate tab for built-in functions available in XSPARQL.

Output Data Format: XSPARQL serves as a bridge for transformation of XML and RDF data in both directions, which allows users to pick any of the RETURN or CONSTRUCT clause for the output data format. This feature enables to perform transformations either by lowering (from RDF to XML) or by lifting (from XML to RDF). Depending on users' choice, the output of an XSPARQL query will be in XML if the RETURN clause is used, or in RDF if the CONSTRUCT clause is used.

Visualisation of Results: XSPARQL-Viz Editor provides visualisation capabilities for the results of the mashups to be displayed as graphs, tables or lists. User can also export the results as an external file. A very important additional benefit of using mashup-based approach is that it allows to save the mashups as a component within XSPARQL-Viz Editor, and to use it later for similar query execution over different data set or use it as input for another query.

4 Potential Impact and Future Directions

In this demo, we showcase the XSPARQL visual query editor for auto-generation of XSPARQL queries. XSPARQL-Viz assists for mapping and transformation between XML and RDF data without having extensive knowledge of XQuery and SPARQL. XSPARQL-Viz is built upon and compliant with the strong theoretical foundations of XSPARQL defined in [1,5]. A preliminary evaluation was performed by inviting 20 semantic web technology experts to evaluate the functionality of XSPARQL-Viz and provide their feedback using an online survey. The initial feedback from the users, provides quite promising results and we believe that XSPARQL-Viz can play a pivotal role for the wide adoption and usability of XSPARQL in various domains and real-world applications. However, we plan to further improve XSPARQL-Viz based on valuable feedback from its users which can be submitted by filling an online survey accessible at <http://xsparql.deri.org/XSPARQL-Viz/survey>

As a future extension we plan to perform extensive usability evaluation of XSPARQL-Viz, and extend it to incorporate two recent extensions of XSPARQL, namely (i) XSPARQL over relational data [8], and (ii) XSPARQL Update Facility [2]. We are also considering to use XSPARQL-Viz as a platform for adaptive generation of data set statistics, which later can be used for query analysis, planning and optimisation.

References

1. Akhtar, W., Kopecký, J., Krennwallner, T., Polleres, A.: XSPARQL: Traveling between the XML and RDF Worlds – and Avoiding the XSLT Pilgrimage. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 432–447. Springer, Heidelberg (2008)
2. Ali, M.I., Lopes, N., Friel, O., Mileo, A.: Update Semantics for Interoperability among XML, RDF and RDB. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 43–50. Springer, Heidelberg (2013)
3. Ali, M.I., Pichler, R., Truong, H.-L., Dustdar, S.: DeXIN: An Extensible Framework for Distributed XQuery over Heterogeneous Data Sources. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2009. LNBIP, vol. 24, pp. 172–183. Springer, Heidelberg (2009)
4. Ali, M.I., Pichler, R., Truong, H.-L., Dustdar, S.: On integrating data services using data mashups. In: Fernandes, A.A.A., Gray, A.J.G., Belhajjame, K. (eds.) BNCOD 2011. LNCS, vol. 7051, pp. 132–135. Springer, Heidelberg (2011)
5. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. *Journal on Data Semantics* 1, 147–185 (2012)

6. Groppe, S., Groppe, J., Linnemann, V., Kukulenz, D., Hoeller, N., Reinke, C.: Embedding SPARQL into XQuery/XSLT. In: Proc. of SAC (2008)
7. Jarrar, M., Dikaiakos, M.D.: A data mashup language for the data web. In: Proc. of LDOW. CEUR Workshop Proceedings, vol. 538. CEUR-WS.org (2009)
8. Lopes, N., Bischof, S., Decker, S., Polleres, A.: On the Semantics of Heterogeneous Querying of Relational, XML and RDF Data with XSPARQL. In: Proc. of EPIA (2011)
9. Polleres, A., Sakr, S.: Querying and Exchanging XML and RDF on the Web. In: WWW 2012 Tutorial (2012),
<http://www2012.wwwconference.org/program/tutorials/>

LinDA: A Service Infrastructure for Linked Data Analysis and Provision of Data Statistics

Nicolas Beck, Stefan Scheglmann, and Thomas Gottron

WeST – Institute for Web Science and Technologies
University of Koblenz-Landau, 56070 Koblenz, Germany
{nico1510, schegi, gottron}@uni-koblenz.de

Abstract. We present LinDA: an extensible, data driven platform where analytical tools can be integrated to process, analyse and describe data sets of Linked Open Data. To date, we have integrated four different tools: the computation of a VoID description, the computation of schema-level index (SchemEX), an information theoretic analysis of the data and the construction of a formal concept lattice. A demo prototype of LinDA is publicly available and allows for uploading data sets for analysis. A web frontend allows users to interact with the system, access the results of previously analysed data sets or to upload new data sets for the analysis tools to operate on.

1 Introduction

The Linked Open Data (LOD) Cloud is growing fast. More and more data sets are published in an open, accessible and machine readable fashion. To make use of this data, end users, developers, data integrators and data engineers need good descriptions and statistics about the available data. Several languages (e.g. VoID), methods (e.g. SchemEX) and tools (e.g. RDFstats) have been proposed to analyse, describe or summarize LOD data on different levels. The uptake of these approaches, however, is far behind its potential. Reasons might be the work overhead in obtaining the description of the data, the installation or usage of research prototypes or simply the lack of awareness about the approaches. Therefore, for many linked data sets there is no good and descriptive meta data available [3].

To alleviate this situation we propose the LinDA platform. LinDA (an acronym for LINKed Data Analysis) provides a central platform where we provide analytical tools as a service to the LOD community. Users can upload snapshots of their data sets to this platform. Subsequently, the platform's data driven design enables the user to process the genuine data set with the tools available on the platform, to use output generated by these tools for further analysis, or both. Dependencies between the analytical tools can be modelled to allow for running the tools incrementally, e.g. the computation of a schema-level index precedes an information theoretic analysis using the schema index as input. Furthermore, all the results can be downloaded for offline use, e.g. by the data owner who wants to provide the descriptions together with the original data.

In this paper we present the general idea and setup of LinDA together with a first prototype implementation. As initial services the prototype includes four tools to analyse and describe Linked Data:

1. The generation of a VoID description [1,4] for the data set.
2. The computation of a formal concept lattice based on a formal concept analysis [8].
3. The computation of a SchemEX index [6].
4. An information theoretic analysis on the interdependencies between explicit and implicit schema information [5].

2 Architecture

The aim of LinDA is to enable users to create and share derived analytical and descriptive meta data about LOD data sets in a simple and effective way. To this end LinDA provides services that combine the computation and storage of meta data about data sets. The services are intended to wrap existing or novel analytical tools operating on LOD. The output and results of this analysis are provided for download as bulk files as soon as computation is completed. This approach has been chosen for three reasons: (a) storage of the data enables re-use of the results by other users, (b) in some cases the output of one analysis serves as basis for the execution of another analytical tools and (c) the analysis and its computation might require a long time which demands for asynchronous response to the user. LinDA is an extensible framework. New analytical tools, which build on or integrate existing results, can be developed and deployed.

2.1 Implemented Service

The current prototype version of LinDA includes four services. The general architecture—as mentioned—is extensible to include further services and also allows for the combination of services in workflow sequences. A light-weight XML-based configuration language allows to model dependencies among services. These dependencies specify which computations can be run in parallel and which ones need to be run in a sequential order. This configurations allows for a flexible description of the services, including parameter settings, execution information as well as input and output data format. Therefore, tools are not bound to be implemented in a specific programming language but we can integrate tools of different breed very easily.

Our services operate on RDF data sets. These data sets can be provided in different serializations: RDF/XML, N3, Turtle, N-Triples and N-Quads. As such serializations of RDF can be of quite large size quickly, we allow for the data to be provided in a compressed format: to date we support tgz, gz and zip compression. Along with the data sets we store some basic meta data about the data set. We use basic Dublin core meta data such as a title or name for the data set (`dcterms:title`) and its publisher (`dcterms:publisher`). Furthermore, we also ask for meta data used in the VoID vocabulary. This covers information like a textual description of the data set (`void:DatasetDescription`) or hand picked representative examples resources of the data set (`void:exampleResource`). This meta data can be provided when a data set is uploaded to LinDA and can help to identify and describe the data set at a very high level and for human users.

LinDA currently provides four services for analysing RDF data sets:

VoID. We compute a VoID description for a given RDF data set. To this end we incorporate a Shell script¹ provided in the context of [4]. The script operates on RDF data sets and generates a VoID description in Turtle notation.

Concept Lattice Construction. We have developed a novel analysis service employing a formal concept analysis [8] to extract a concept lattice from a given data set of RDF. The resulting lattice is encoded in RDF and describes the type and property composition of the analysed LOD resources. Additionally to providing the computed concept lattice as bulk download, the service also allows for browsing the descriptions and publishes them as LOD themselves. To this end we implemented a REST API to dereference the URIs of concepts.

SchemEX. The definition and construction of a schema based index for LOD is described in [6]. The index uses equivalence classes over schema-level information to segment the analysed data in distinct groups. The groups form an index which allows for the lookup of meta data about the resources being in the group, e.g. number or resources, relevant data sources on the LOD cloud or examples resources. We allow for the computation of such an index using the scalable stream based approach. The index itself is encoded in RDF as well.

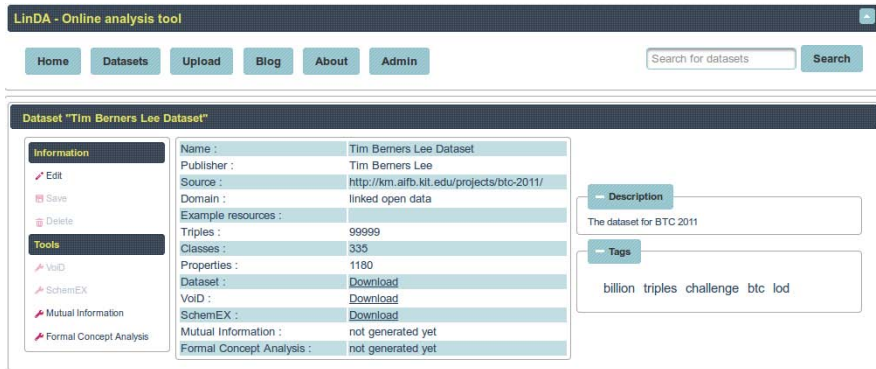
Information Theoretic Analysis. Using a SchemEX index as computed in the previous service allows for an information theoretic analysis of the data [5]. In this way it is possible to judge the homogeneity or diversity of the explicit and implicit schema information extracted from the data. The results come in the form of a few aggregated metrics, such as Conditional Entropy or Mutual Information.

2.2 Data Management and Processing

The processes in LinDA follow a data driven approach. All data sets and results are cached in the system. In this way the results of an analysis do not need to be computed from scratch whenever they are requested, but can be retrieved immediately. This is of importance not only for end users accessing the results but also for services which operate on the outcome of other services. These workflow can then be executed temporally separated and do not need to be executed immediately one after the other.

All services are triggered by events. These events can be issued explicitly by users interacting with the system or by other services which depend on the prior computation of another service. The events triggering the analysis jobs are deserialized into messages and sent to a JMS queue. Message consumers can subscribe to this queue, complete the jobs and store the results. In the current prototype system we have a setup with a single message consumer but the design itself allows for job execution on multiple machines. Making use of a JMS queue also provides a fail safe feature for incompleted jobs. For instance, on huge data sets a small scale consumer machine might run out of resources and fail to complete the requested analysis. These requests are not discarded but stored in the message queue until a consumer is eventually able to process them.

¹ The script is available online at http://code.google.com/p/rdffederator/source/browse/trunk/scripts/generate_void_description.sh



The screenshot shows the LinDA web interface. At the top, there is a navigation bar with links for Home, Datasets, Upload, Blog, About, and Admin. A search bar is located on the right side of the navigation bar. Below the navigation bar, the main content area displays the details for a dataset titled "Dataset 'Tim Berners Lee Dataset'". On the left side, there is a sidebar with two sections: "Information" and "Tools". The "Information" section contains links for Edit, Save, and Delete. The "Tools" section contains links for VoID, SchemEX, Mutual Information, and Formal Concept Analysis. The main content area is divided into two columns. The left column contains a table of metadata for the dataset, and the right column contains a description and tags.

Dataset "Tim Berners Lee Dataset"	
Name :	Tim Berners Lee Dataset
Publisher :	Tim Berners Lee
Source :	http://km.aifb.kit.edu/projects/btc-2011/
Domain :	linked open data
Example resources :	
Triples :	99999
Classes :	335
Properties :	1180
Dataset :	Download
VoID :	Download
SchemEX :	Download
Mutual Information :	not generated yet
Formal Concept Analysis :	not generated yet

Description
The dataset for BTC 2011

Tags
billion triples challenge btc lod

Fig. 1. Information about a data set as presented in the web frontend in LinDA

A dedicated services takes care of user notification. This service provides feedback to the end user who has triggered a service as soon as its computation is completed. This is useful in particular for larger data sets, processes with a longer runtime or during peak times when the system has many requests in the queue. In these cases, the users can issue their request and receive an e-mail once their job has been completed.

3 LinDA Prototype

For demo purposes we have implemented a prototype web interface for LinDA (cf. Figure 1). The system is available at <http://linda.west.uni-koblenz.de/>. The web interface lists all data sets which have been uploaded so far along with the descriptive high level meta data provided during upload. In a detailed view, the users can see for each data set which analytical services have already been computed on the data, which services are currently being computed and which results are available. By this means the users do also get an overview of the available services.

When uploading a new data set, users are asked to provide meta data on the title, publisher and a description of the data set. When issuing the computation of an analytical service on the data, users can provide their e-mail address to receive the notification about the completion of the job. Viewers interested in the result of a currently running computation can subscribe to jobs and receive the completion notification as well.

4 Related Work

Analysis of Linked Open Data or RDF data sets in general is addressed by many works. The services we integrated in our framework cover some of these analysis. SchemEX [6], for instance, extracts schema information from RDF to construct a schema-based index structure. This index allows for answering schema-oriented information needs. In a different setting the same index structure is used as basis for an information theoretic analysis of RDF data [5]. VoID is a vocabulary specifically designed for providing meta data about RDF data sets [1]. RDFStats [7] combines several

analysis and provides statistical information about RDF data sets as well as SPARQL endpoints. LODstats [2] is an extensible framework which focusses on statistical analysis of LOD data sets. It integrates RDFStats and makes use of VOID descriptions for computing further statistical metrics. With LinDA we aim for a more general approach which allows for computation of index structures and derived or descriptive meta data as well.

5 Summary

We presented LinDA, a web based service for computing various analytics on Linked Open Data. Data sets can be uploaded and analysed by different services. These services can also be combined into workflows to perform analysis on top of the results of another service. To date, the platform provides four types of analysis. However, the platform is extensible to include additional analysis in the future. In a next step we plan to extend the infrastructure to an actually distributed setup of several machines consuming requests for computing analytical services. Furthermore, we want to extend the dereferencing functionality implemented for the formal concept lattice to the results of the schema-level index in SchemEX. Then, it is possible to directly access the schema itself in the form of LOD. Finally, we plan to integrate visualizations for some of the results where such visualizations are applicable. This will include at least plots of histograms or distributions obtained from the statistical analysis. The visualizations will then be displayed in the web frontend together with the basic meta data available already.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257859, ROBUST.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets with the void vocabulary, <http://www.w3.org/TR/void/> (accessed: April 11, 2013)
2. Auer, S., Demter, J., Martin, M., Lehmann, J.: LODStats – an extensible framework for high-performance dataset analytics. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 353–362. Springer, Heidelberg (2012)
3. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the lod cloud, <http://www4.wiwiwiss.fu-berlin.de/lodcloud/state/> (accessed: April 11, 2013)
4. Görlitz, O., Staab, S.: SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In: Proceedings of the 2nd International CoLD Workshop, Bonn, Germany (2011)
5. Gottron, T., Knauf, M., Scheglmann, S., Scherp, A.: A Systematic Investigation of Explicit and Implicit Schema Information on the Linked Open Data Cloud. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 228–242. Springer, Heidelberg (2013)
6. Konrath, M., Gottron, T., Staab, S., Scherp, A.: Schemex—efficient construction of a data catalogue by stream-based indexing of linked data. Web Semantics: Science, Services and Agents on the World Wide Web 16(5), 52–58 (2012), The Semantic Web Challenge 2011

7. Langegger, A., Woss, W.: Rdfstats - an extensible rdf statistics generator and library. In: Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application, DEXA 2009, pp. 79–83. IEEE Computer Society, Washington, DC (2009)
8. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In: Ferré, S., Rudolph, S. (eds.) ICFA 2009. LNCS, vol. 5548, pp. 314–339. Springer, Heidelberg (2009)

Identifying Functions of Citations with CiTalO

Angelo Di Iorio¹, Andrea Giovanni Nuzzolese^{1,2}, and Silvio Peroni^{1,2}

¹ Department of Computer Science and Engineering, University of Bologna, Italy

² STLab-ISTC Consiglio Nazionale delle Ricerche, Italy

{diiorio,nuzzoles,essepuntato}@cs.unibo.it

Abstract. Bibliographic citation is one of the most important activities of an author in the production of any scientific work. The reasons that an author cites other publications are varied: to gain assistance of some sort, to review, critique or refute previous works, etc. In this paper we propose a tool, called *CiTalO*, to infer automatically the nature of citations by means of Semantic Web technologies and NLP techniques. Such a characterisation makes citations more effective for linking, disseminating, exploring and evaluating research.

1 Introduction

Bibliographic citations are the most used tools of academic communities for *linking* research, for instance by connecting scientific papers to related works or sources of experimental data. Citations are also tools for *disseminating*, as largely discussed in [9], and *exploring* research, for instance providing new interfaces for browsing data. Finally, citations are useful for *evaluating* research, e.g. through bibliometric measures such as *h-index* and *impact factor*.

All these activities can be radically improved by exploiting the actual “nature” of citations, i.e. the “author’s reason for citing a given paper” [11]. The mere existence of a citation, in fact, does not provide any information about the reasons the author had in mind when creating that citation to some particular document rather than to another. It is the characterization of a citation that really capture its meaning and effect.

The goal of this paper is to present *CiTalO*, a tool that automatically annotates citations with properties defined in *CiTO* (*Citation Typing Ontology*)¹ [7]. These properties describe the nature of citations in scholarly works.

CiTalO is implemented in Java and can be used as either stand-alone component or web service. A demo version is also available at <http://wit.istc.cnr.it:8080/tools/citalo>: users can use a simple HTML form to submit an English sentence containing a citation to CiTalO and to receive the list of *CiTO* properties that characterize the nature of that citation. Multiple configurations can also be tested by using the same prototype. CiTalO exploits Semantic Web technologies and NLP techniques to produce the output. The tool is designed as a *chain of analysers* that (i) produce ontological statements from texts, (ii) search

¹ CiTO: <http://purl.org/spar/cito>

patterns in those statements, (iii) maps those patterns into linguistic resources and (iv) use these resources to produce the final characterization conform to CiTO. The chain also includes a sentiment-analysis module to refine results.

The paper is structured as follows. In Section 2 we introduce previous works on classification of citations. In Section 3 we describe CiTalO introducing its structure. In Section 4, we conclude the paper sketching out some future works.

2 Related Works

In [3] Copestake *et al.* introduce the SciBorg framework, which includes a module for discourse and citation analysis that follows the *Argumentative Zoning* scheme proposed by Teufel *et al.* [10] and produces quite good results.

Teufel *et al.* present a study about *function* of citations [11]. They provide a categorisation of possible citation functions organised in twelve classes, in turn clustered in *Negative*, *Neutral* and *Positive* rhetorical functions. They also performed some tests on hundreds of articles in computational linguistics, evaluating the output of several human annotators and a novel machine learning approach, and showed that the agreement between humans is actually higher than the agreement between humans and automatic analysis. Along the lines of the latter work, also Jorg analysed several documents within the ACL Anthology Networks² with the intent of identifying verbs usually used to carry important information about the nature of citations [6].

Closely related to the annotation of citation functions, in [2] Athar *et al.* propose and evaluate (with good result) a sentiment-analysis approach to citations, so as to identify whether a particular act of citing was done with positive (e.g. praising a previous work on a certain topic) or negative intentions (e.g. criticising the results obtained through a particular method).

3 CiTalO

CiTalO tries to guess the function of citations by combining techniques of ontology learning from natural language, sentiment-analysis, word-sense disambiguation, and ontology mapping. These techniques are thought to be applied in a pipeline whose input is the sentence of an article containing the citation – e.g. “It extends the research outlined in earlier work X”, where *X* is a reference to a particular bibliographic entity – and the output is one or more properties of the CiTO ontology [7] – *cito:extends* for the previous example. The overall architecture is shown in Fig. 1, while an extensive explanation of features and drawbacks of CiTalO can be found in [4].

Sentiment-Analysis for Gathering the Polarity of Citation Functions. The aim of this step is to capture the sentiment polarity emerging from the text in which the citation is included. This is connected to the classification of CiTO properties provided in [7], where the semantics of rhetorical citations

² ACL Anthology Network: <http://clair.eecs.umich.edu/aan/index.php>

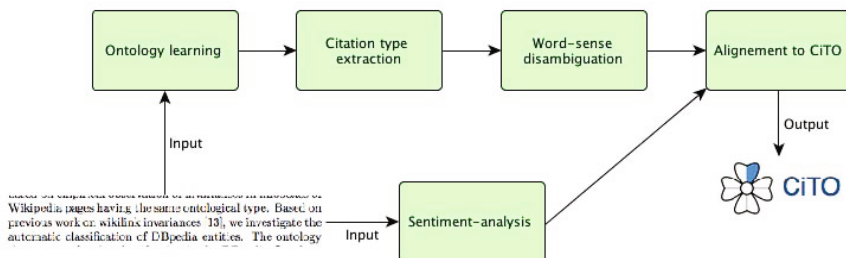


Fig. 1. The pipeline used by CiTaLO. The input is the textual context in which the citation appears and the output is a set of properties of CiTO.

is expressed according to three different polarities, i.e. positive, neuter and negative. Being able to recognize the polarity behind the citation, in fact, would restrict the set of possible target properties from the CiTO ontology to match. Notice also that such an analysis goes in parallel with the others in CiTaLO, being it a refinement filter of the results. The current sentiment-analysis component is based on *AlchemyAPI*³ but it can be easily replaced with other similar tools.

Ontology Extraction from the Textual Context of the Citation. The first mandatory step of CiTaLO consists of deriving a logical representation of the sentence containing the citation. This ontology extraction is performed by using FRED [8], a tool for ontology learning based on discourse representation theory, frames and ontology design patterns. The transformation of the sentence into a logical form allows us to recognize graph-patterns in order to detect possible types of rhetorical denotation of the citation. Consider, for instance, the sentence “it extends the research outlined in earlier work X”, where “X” is the cited work. The graphical representation of the output in FRED, that is also available as RDF statements, is presented in Fig. 2.

Citation Type Extraction through Pattern Matching. The second step consists of extracting candidate types for the citation, by looking for patterns in the FRED result. We designed several graph-pattern-based heuristics by following similar criteria as lexico-syntactic patterns [1], extended with the exploitation of RDF graph topology and OWL semantics. These heuristics are implemented as SPARQL queries and some example are shown below:

```

SELECT ?type WHERE {?subj ?prop fred:X . ?subj a ?type}
SELECT ?type WHERE {?subj ?prop fred:X . ?subj a ?typeTmp .
  ?typeTmp rdfs:subClassOf+ ?type}
SELECT ?type WHERE {?subj a dul:Event . ?subj a ?type .
  FILTER(?type != dul:Event)}
SELECT ?type WHERE {?subj a dul:Event . ?subj a ?typeTmp .
  ?typeTmp rdfs:subClassOf+ ?type.FILTER(?type != dul:Event)}
SELECT ?type WHERE {?subj a dul:Event .
  ?subj boxer:patient ?patient . ?patient a ?type}
  
```

³ *AlchemyAPI*: <http://www.alchemyapi.com>

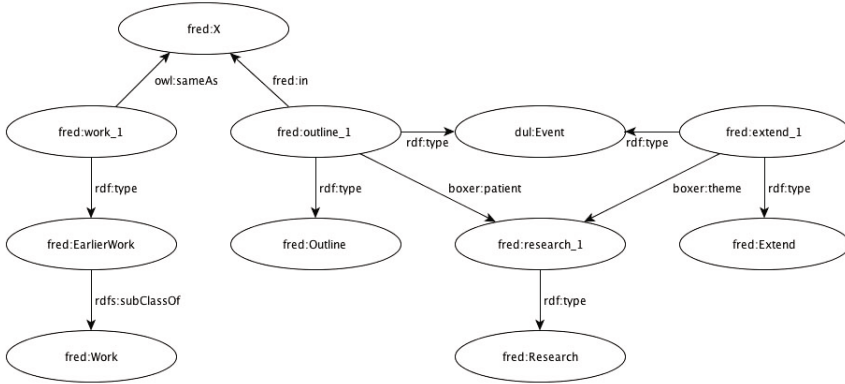


Fig. 2. FRED result for “It extends the research outlined in earlier work X”

The intended semantics of the above patterns is to select from the RDF graph all the types and their eventual taxonomies related to (i) the cited document, (ii) the events recognized into the citation, and the entities affected by those events (i.e. the entities playing the VerbNet role of being patient).

Applying these patterns to graph shown in Fig. 2, the following candidate types are found: *Outline*, *Extend*, *EarlierWork*, *Work*, and *Research*. The current set of heuristics is quite simple and incomplete, but we are continuously updating the catalogue by both investigating new heuristics.

Word-Sense Disambiguation. The next step consists of disambiguating the sense of each candidate type. This can be done through word-sense disambiguation services and APIs – in CiTalO we use IMS [12]. The disambiguation is performed with respect to OntoWordNet [5] and produces a list of synsets for the candidate types. Going back to the example, this phase would produce the following list⁴: (i) *Extend* is disambiguated as `own:synset-prolong-verb-1`, (ii) *Outline* as `own:synset-delineate-verb-3`, (iii) *Research* as `own:synset-research-noun-1`, (iv) *EarlierWork* and *Work* as `own:synset-work-noun-1`.

Alignment to CiTO. The last step consists of associating each synset to a CiTO property and refining results by using citation polarities and factual characterisation. We use two ontologies for this purpose: *CiTO2Wordnet* and *CiTOFunctions*. *CiTO2Wordnet*⁵ maps all the CiTO properties defining citations with the appropriate Wordnet synsets [5]. *CiTOFunctions*⁶ classifies each CiTO properties according to their factual and rhetorical functions [7]. The final alignment to CiTO is performed by means of a SPARQL CONSTRUCT query that uses the enhanced RDF graph obtained during the pipeline, the RDF graph of the polarity, OntoWordNet and the two ontologies just described.

⁴ The prefix *own* stands for <http://www.w3.org/2006/03/wn/wn30/instances/>

⁵ CiTO2Wordnet ontology: <http://www.essepuntato.it/2013/03/cito2wordnet>

⁶ CiTOFunctions: <http://www.essepuntato.it/2013/03/cito-functions>

4 Conclusions

CiTalO integrates Semantic Web technologies and NLP techniques to extract information about the nature, the motivations and the goals of each citation. The CiTalO architecture is composed of a pipeline of modules that map documents into ontological data, ontological data into linguistic resources and, finally, linguistic resources into CiTO properties.

The implementation is still at an early stage. On the other hand, the overall approach is very open to incremental refinements. We are currently working to improve *patterns' matching* phases in CiTalO and to include a mechanism for the automatic identification of *textual context* of citations given an input article. We also plan to perform exhaustive tests with a large set of documents and users.

References

1. Aguado de Cea, G., Gómez-Pérez, A., Montiel-Ponsoda, E., Suárez-Figueroa, M.C.: Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 32–47. Springer, Heidelberg (2008)
2. Athar, A., Teufel, S.: Context-Enhanced Citation Sentiment Detection. In: Proceedings of HLT-NAACL 2012, pp. 597–601 (2012)
3. Copestake, A., Corbett, P., Murray-Rust, P., Rupp, C.J., Siddharthan, A., Teufel, S., Waldron, B.: An architecture for language processing for scientific text. In: Proceedings of the UK e-Science All Hands Meeting (2006)
4. Di Iorio, A., Nuzzolese, A.G., Peroni, S.: Towards the automatic identification of the nature of citations. To appear in Proceedings of SePublica 2013 (2013)
5. Gangemi, A., Navigli, R., Velardi, P.: The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet. In: Meersman, R., Schmidt, D.C. (eds.) CoopIS/DOA/ODBASE 2003. LNCS, vol. 2888, pp. 820–838. Springer, Heidelberg (2003)
6. Jorg, B.: Towards the Nature of Citations. In: Poster Proceedings of FOIS 2008 (2008)
7. Peroni, S., Shotton, D.: FaBiO and CiTO: ontologies for describing bibliographic resources and citations. Journal of Web Semantics 17, 33–43 (2012), doi:10.1016/j.websem.2012.08.001
8. Presutti, V., Draicchio, F., Gangemi, A.: Knowledge extraction based on discourse representation theory and linguistic frames. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 114–129. Springer, Heidelberg (2012)
9. Shotton, D.: Semantic publishing: the coming revolution in scientific journal publishing. Learned Publishing 22(2), 85–94 (2009)
10. Teufel, S., Carletta, J., Moens, M.: An annotation scheme for discourse-level argumentation in research articles. In: Proceedings of the 9th Conference of the EACL 1999, pp. 110–117 (1999)
11. Teufel, S., Siddharthan, A., Tidhar, D.: Automatic classification of citation function. In: Proceedings of EMNLP 2006, pp. 103–110 (2006)
12. Zhong, Z., Ng, H.T.: It Makes Sense: A wide-coverage word sense disambiguation system for free text. In: Proceedings of ACL 2010, System Demonstrations, pp. 78–83 (2010)

Graphia: Extracting Contextual Relation Graphs from Text

Danilo S. Carvalho¹, André Freitas³, and João C.P. da Silva²

¹ PESC/COPPE

² Computer Science Department

Universidade Federal do Rio de Janeiro (UFRJ), Brazil

³ Digital Enterprise Research Institute (DERI)

National University of Ireland, Galway, Ireland

Abstract. This demo presents *Graphia*, an information extraction pipeline targeting an RDF representation of unstructured data in the form of structured discourse graphs (SDGs). It combines natural language processing and information extraction techniques with the use of linked open data resources and semantic web technologies to enable discourse representation as a set of contextualized relationships between entities.

Keywords: Open Information Extraction, Linked Open Data, Natural Language Processing.

1 Introduction

The Linked Data Web brings the vision of a Web-scale semantic data graph layer which can improve the ability of users and systems to access and semantically interpret information. Most of the information available on the Web today is in an unstructured text format. The integration of this information into the Linked Data Web is a fundamental step towards enabling the Semantic Web vision. The semantics of unstructured text, however, does not easily fit into structured datasets. The representation of information extracted from texts needs to take into account large terminological variation, complex context patterns, fuzzy and conflicting semantics and intrinsically ambiguous sentences.

Typical information extraction (IE) approaches for extracting relations from unstructured text have either focused on the extraction of simple relations (triples) or on specific patterns which are going to feed a well structured ontology (e.g. events), scenarios where accuracy, consistency and a high level of lexical and structural normalization are primary concerns. The purpose of Graphia is to show that these IE approaches can be complemented by alternative information extraction scenarios where accuracy, consistency and regularity are traded by domain-independency, context capture, wider extraction scope and maximization of the text's semantic dependencies representation, where data semantics and data quality can be improved over time.

The demo focuses on the contextual capture gain that can be attained by the combination of linguistic information, linked open data and a flexible and extensible graph relation representation model. The differences between Graphia and

other relation extraction systems are examined in the light of these motivations, and the extraction workflow is described. We also look into the entity-centric integration of the extracted graphs into the existing Linked Data Web.

2 Overview

Graphia is an application for graph relation extraction from natural language text, meaning that it takes factual text as input and produces entity-centric data graphs as output. Each node of the graph represents an entity (named or non-named) and the edges indicate the relations (e.g. actions, locations, pertinence) between the nodes.

The main difference in the Graphia relation extraction, when compared to other relation extraction tools, is its ability of capturing relations that are not in the scope of simple triples, either as a triple context or entity context. To accommodate the contextual representation, a principled entity-centric structured data graph (SDG) representation and interpretation model is introduced [1]. Figure 1, Figure 2 and Figure 3 depict examples of Graphia's SDG output.

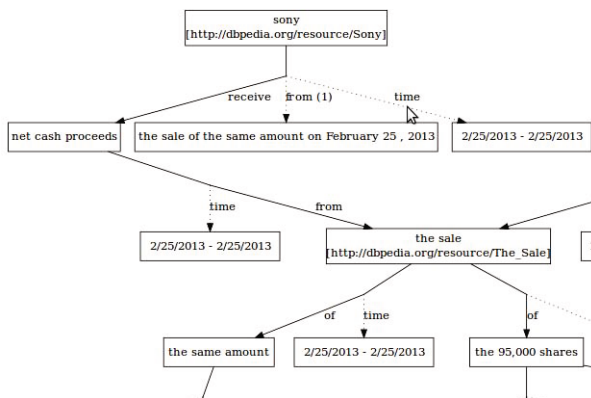


Fig. 1. Fragment of a graph extracted from a sentence on a *sec.gov* report: ... *Sony will receive net cash proceeds from the sale of the same amount on February 25, 2013.* Nodes represent *entities*, filled edges represent *relations* and dotted edges are *reifications* (i.e. context of relations).

The most common way of expressing relations in a information extraction system is the *subject-predicate-object* (*s,p,o*) triple, where the predicate expresses the relation between the subject and the object. Graphia builds on top of this representation model by using SDGs. The use of SDGs supports a representation which focuses on the capture of complex contextual dependencies without committing to a specific conceptual model. The system is designed to make use of syntactic information where possible, and helps to bridge the terminological, context, and semantic gap by connecting this information with the entities offered by linked open data such as DBpedia [2].

3 Demonstration and Workflow Description

Graphia is implemented as an open information extraction pipeline, in which each step does a well defined task and makes available a set of new data to the next step. The sequence of steps is described next, with the output for the sample sentence “*John Doe is a very important person in USA and he was born in the UK.*”:

Syntactic Analysis: The first step in the extraction process is the syntactic parsing of the natural language text into syntactic trees (C-Structures). This component uses the Probabilistic Context-Free Grammar (PCFG) implemented in the Stanford parser [3]. The C-Structures for the sentences are passed to the next components.

Named Entity Resolution: This component resolves named entities text references to existing DBpedia URIs. The first step consists in the use of the DBpedia Spotlight service¹ where the text is sent and is returned annotated with URIs. The second step consists in the use of Part-of-Speech tags together with C-Structures to aggregate words into entity candidates which were not resolved by the DBpedia Spotlight service. The entity candidates’ strings are resolved by using a local entity index which indexes all DBpedia URIs using TF/IDF over labels extracted from the URIs, and validated through a TF-IDF threshold check. The output of this component is the original text with a set of named entity terms annotated with URIs. In this step, *John Doe*, *USA* and *the UK* gets annotated with DBpedia URIs.

Personal Co-reference Resolution and Normalization: This component resolves pronominal co-references including personal, possessive and reflexive pronouns. Personal pronouns instances are substituted by the corresponding entities. Possessive and reflexive pronouns are annotated with the corresponding entities that will later define the co-reference links. The co-reference resolution process is done by the pronoun-named entity gender and number agreement (by taking into account gender information present in a name list from the public USA census data) and position-based heuristics. The output of this component are C-Structures with annotated named entities, co-reference substitutions for personal pronouns and possessive and reflexive pronouns annotated with named entities. In this step, *He* gets substituted by the annotated *John Doe*.

Graph Extraction: The graph extraction module takes as input the annotated C-Structures and generates the triple trees for each sentence by the application of a set of ten manually designed transformation rules based on syntactic conditions through a DFS traversal of the C-Structure. Instead of focusing on terminology-dependent patterns, these rules are based on syntactic patterns. In this step,

¹ <http://dbpedia.org/spotlight>

two graphs are extracted, as the sentence can be divided by the coordinating conjunction *and*: one centered on the relation *is* and the other on the relation *was born*.

Graph Construction: This component receives the triple trees from the previous component and outputs the final graph serialization. Additionally, local URIs are created for each resource which was not resolved to a DBpedia URI. The output is depicted in Figure 2.

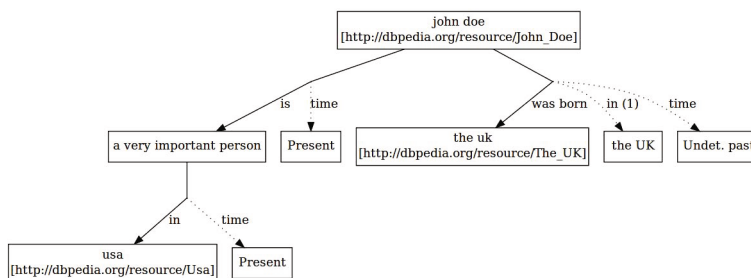


Fig. 2. Example of extracted graph

The Demonstration^{2,3} of the application running can be found on the web.

4 Related Applications

The three applications most closely related to Graphia are ReVerb⁴ [4], the relation extraction module of Alchemy API⁵ and FRED⁶ [5]. ReVerb algorithms are able to extract long relations, like the relation between Neil Armstrong and the Moon: “the first man to walk on”, while Alchemy API can only extract short (verbal) relations. Both focus on simple binary relations, generating a list of (s,p,o) triples. FRED produces RDF/OWL ontologies and linked data from natural language sentences, sharing the goal of integrating unstructured data with open web ontologies but focusing on fitting the text data on a ontology model and discarding what doesn’t fit. In contrast, Graphia tries to maximize the amount of extracted information, mapping contextual dependencies such as location, dates and quantities for the relations, and outputting graphs that admit the transformations: a) edges to simple relations and b) edge paths to long relations.

² <http://graphia.dcc.ufrj.br>

³ <http://graphia.dcc.ufrj.br/eswcdemo>

⁴ <http://reverb.cs.washington.edu>

⁵ <http://www.alchemyapi.com>

⁶ <http://wit.istc.cnr.it/stlab-tools/fred>

A extraction example for the three applications is shown on Figure 3.

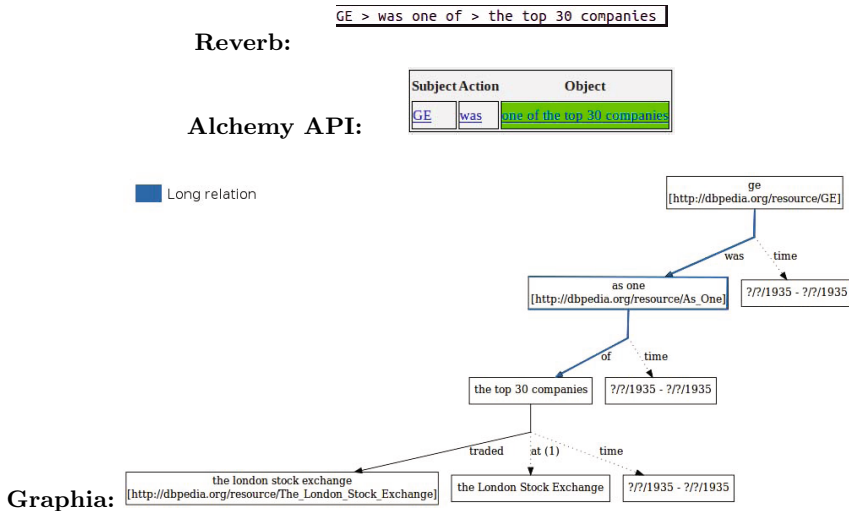


Fig. 3. Example of extractions on ReVerb, Alchemy API and Graphia for the sentence “In 1935, GE was one of the top 30 companies traded at the London Stock Exchange”

5 Conclusions and Future Work

Graphia builds on top of existing open information extraction technologies, offering more contextualized relation extraction results. The goal of enabling discourse representation as a set of contextualized relationships between entities is attained by the combination of linguistic information, linked open data and a flexible and extensible representation model. Broadening context capture is the main point of improvement for future work. Another major improvement is the inclusion of subordinate sentences. Currently integration with existing Linked Data resources is focused on instances and classes, by means of entities alignment with DBpedia.

References

- Freitas, A., Carvallho, D.S., da Silva, J.C.P., O’Riain, S., Curry, E.: A Semantic Best-Effort Approach for Extracting Structured Discourse Graphs from Wikipedia. In: Proc. of the 1st Workshop on the Web of Linked Entities (WoLE) at the 11th International Semantic Web Conf., ISWC (2012)
- Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia – A Crystallization Point for the Web of Data. Journal of Web Semantics: Science, Services and Agents on the World Wide Web (7), 154–165 (2009)

3. Klein, D., Manning, C.D.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423–430 (2003)
4. Fader, A., Soderland, S., Etzioni, O.: Identifying Relations for Open Information Extraction. In: Conf. on Empirical Methods in Natural Language Processing (2011)
5. Presutti, V., Draicchio, F., Gangemi, A.: Knowledge extraction based on discourse representation theory and linguistic frames. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d’Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 114–129. Springer, Heidelberg (2012)

Cross-Lingual Querying and Comparison of Linked Financial and Business Data

Seán O’Riain¹, Barry Coughlan¹, Paul Buitelaar¹, Thierry Declerck²,
Uli Krieger², and Susan Marie-Thomas³

¹ Digital Enterprise Research Institute, NUI Galway, Ireland
{sean.oriain, barry.coughlan, paul.buitelaar}@deri.org

² DFKI, Saarbrücken, Germany
{Declerck, krieger}@dfki.de

³ SAP Research, Karlsruhe, Germany
susan.marie.thomas@sap.com

Abstract. Cross lingual querying of financial and business data from multi-lingual sources requires that inherent challenges posed by the diversity of financial concepts and languages used in different jurisdictions be addressed. Ontologies can be used to semantically align financial concepts and integrate financial facts with other company information from multilingual, semi-structured and unstructured Open Data sources. Availability as Linked Data then allows cross-lingual interrogation of the interlinked multi-lingual data set. This paper presents how the use of semantics and Linked Data enables the alignment and integration of business and financial facts provided by the different European Business Registers. The demonstrator allows business users to query multilingual data, perform comparisons, and review generated financial metrics.

Keywords: Multilingual, Ontology Alignment, XBRL, Financial Analytics.

1 Business Motivation and Research Context

Business Registers throughout Europe are tasked with the collection, publishing exchange and provision of companies’ information and annual financial statements. Currently individual registers service specific information requests by returning populated pre-defined XBRL formatted templates with elements and content specific to the country of origin. Any further correlation or integrated analysis is left to the individual data requestor. To address this problem the Europe Business Registers Working Group (xEBR WG)¹ promotes registry information harmonization through the use of the eXtensible Business Reporting language (XBRL)² and the development of the xEBR Core Taxonomy, which was mapped to existing country-specific accounting taxonomies³, company identification taxonomies and extended with key financial ratios.

¹ <http://www.xbrleurope.org/working-groups/xebr-wg>

² <http://www.xbrl.org/>

³ Also referred to as Generally Accepted Accounting Principles or GAAP’s.

This paper introduces the Business Intelligence Cross-lingual XBRL (BIXL) demonstrator, developed in cooperation with the xEBR WG as one of the Monnet Projects⁴ use cases. Building on xEBR WG efforts, BIXL enables information dependent activities such as financial analysis and general business investigation. Analysts are able to locate, access, and compare key data and financial figures from reports and text, regardless of the XBRL jurisdiction taxonomy used to mark up the original report or the language used. BIXL functionality was developed by combining:

- i) Alignment of xEBR Core Taxonomy mappings with selected XBRL country taxonomies and extraction of financial data from XBRL filings as RDF triples.
- ii) Extraction, annotation and alignment of other key business concepts (e.g. stock exchange name, number of employees, and activity sector) which are extracted from multilingual web semi-structured/unstructured text sources as RDF triples.
- iii) Integration of alignment results into the Monnet Financial Ontology (MFO) [1] and its instantiation as Linked Financial Data. Use of SPARQL queries to generate financial data summaries, metrics and comparison tables.

The BIXL demonstrator builds on the notion of ontology localization [2] and emerges from previous attempts directed at financial term translation using statistical MT [3] and hybrid methods[4]. The demonstrators use of multilingual sources additionally progresses the availability and ability to deal with cross-jurisdictional financial Linked Data [5] as part of a move towards the Multilingual Semantic Web [6]. A recent monolingual commercial example offering a similar single entry point to company information, people, financial data, including ratios statement and disclosures, is 9W Search⁵, which targets mobile delivery of financial information.

A brief overview of the BIXL demonstrator architecture is first presented (Section 2) before a walk-through of the cross-lingual query and comparability functionality (Section 3).

2 Linked Data Integration and Processing Architecture

Figure 1 illustrates the 3 stage processing architecture and tools used to generate fact instances compatible with the Monnet Financial Ontology [1] (MFO). The data source layer outlines the XBRL jurisdictions and company information sources which are then converted (circular nodes)⁶ into MFO RDF compliant triples and axioms, as part of the MFO conversion process. OpenRDF Sesame was chosen as an RDF store due to its implementation of the SPARQL 1.1 syntax and the use of SPARQL property paths in simplifying tree structure navigation. The data layer is responsible for constructing and executing SPARQL queries against the RDF data store and marshalling of result sets into JSON Strings. Utility classes are available to convert result sets into a tree representation, needed later for traversal of the xEBR Core Taxonomy structure.

⁴ <http://www.monnet-project.eu>

⁵ <http://9wsearch.com/>

⁶ Available for download from

<https://github.com/monnetproject/rdfconverters>

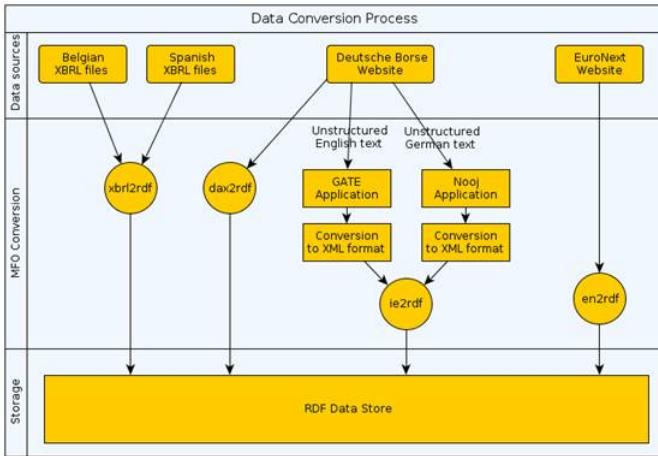


Fig. 1. Multi-lingual Financial and Business Data Integration, adapted from [5]

3 Cross-Lingual Querying and Comparison of Linked Financial and Business Data Demonstrator

The BIXL’s web interface allows business information querying from XBRL and stock exchange sources. Users can query, navigate and compare financial filing and company profile information across multiple languages and financial jurisdictions. Users are initially presented with a search form (Figure 2) to narrow selection based on data source (i.e. the type of query), the particular companies under investigation and the applicable time period.

Fig. 2. Business Information Source Selection Filter

Available query types are Company Profiles, Financial Filing Summaries/Metrics or a combination of both. Companies are selected using the auto-complete search box and company name, identifier (or ISIN number if unavailable), and the number of available filings/company profile snapshots in the RDF store, displayed. Users can select from the project supported languages of English, Spanish, German or Dutch, also used to specify the language for cross-lingual querying and interface labels. Search forms display results in an easy to access tabularized comparable format. The ‘Company Profile’ view shown in Figure 3, provides a side-by-side comparison of key data for two selected companies. Temporal specific snapshots for each company can be selected via the drop-down boxes. Displayed information extracted from unstructured text is underlined with dashes. When the user hovers over a fact extracted from unstructured text, a tool-tip displays the original text and outlines where the fact was extracted from, here in German.

Select Snapshot:		Deutsche Börse - 2012-06-06	Deutsche Börse - 2012-03-01
Name	Bayerische Motoren Werke Aktiengesellschaft BMW AG St BMW Group	DEUTZ AG Deutz AG	
Stock Exchange	Deutsche Börse	Deutsche Börse	
ISIN	DE0005190003	DE0006305006	
Total Capital Stock	€655,566,568	€308,978,241	
City	80788 München	51149 Köln	
Country	Germany	Germany	
Telephone	+ 49 (0)89 382-0	+49 (0)221 822-0	
Employees	95500	3839	
Internet	http://www.bmwgroup.com	http://www.deutz.com	
End of Business	Im Geschäftsjahr 2010 erzielte die BMW Group einen weltweiten Absatz von 1,46 Millionen Automobilen und über 110.000 Motorrädern. Das Ergebnis vor Steuern belief sich auf rund 4,8 Mrd. Euro, der Umsatz auf 60,5 Milliarden Euro. Zum 31. Dezember 2010 beschäftigte das Unternehmen weltweit rund 95.500 Mitarbeiterinnen und Mitarbeiter.		

Fig. 3. Company Profiles

The ‘Financial Filings Summary’ view shown in Figure 4, provides a comparative view of financial data extracted from XBRL filings.

The data, displayed in a tree table corresponds to the hierarchical structure of the xEBR Core Taxonomy. To assist users manage the level of information displayed, abstract concept nodes (aggregated financial figures) and their child nodes (individual financial figures) can be collapsed or expanded. Concepts found in the XBRL filings with corresponding instances are not shown with the displayed data. Colour coding is added to assist easy visual identification and comparison. Selecting the hyper-linked financial concept name (here “Assets”), provides, a detailed plot of values over time across the companies considered by the query. Source taxonomies provenance used to generate the xEBR data is also listed along with equivalent concept names in available project languages. Generated Financial Metrics (Figure 5) can be viewed by selecting “Metrics” from the drop down combo box in Figure 4.

The Metrics view displays both Business Registry specified financial metrics (e.g. Own Funds, Net Results) and generally applied investigative accounting ratios

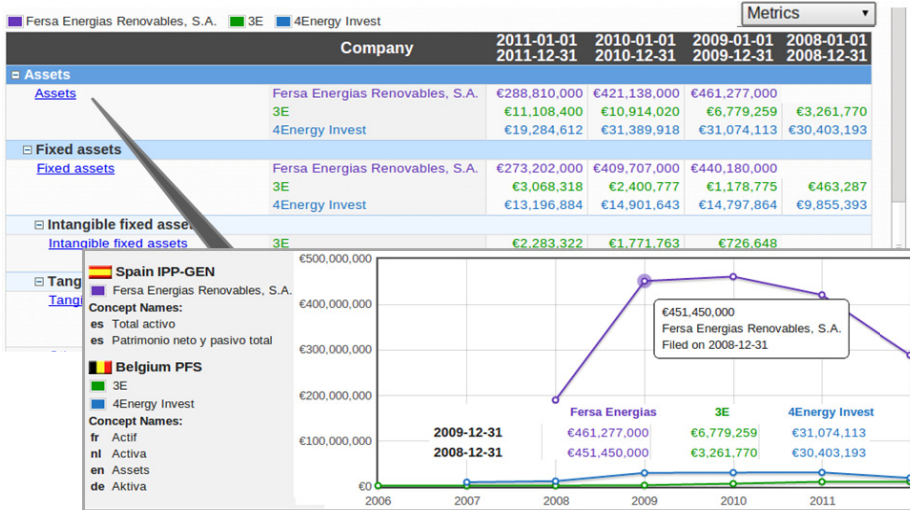


Fig. 4. Financial Filings Summary Comparison

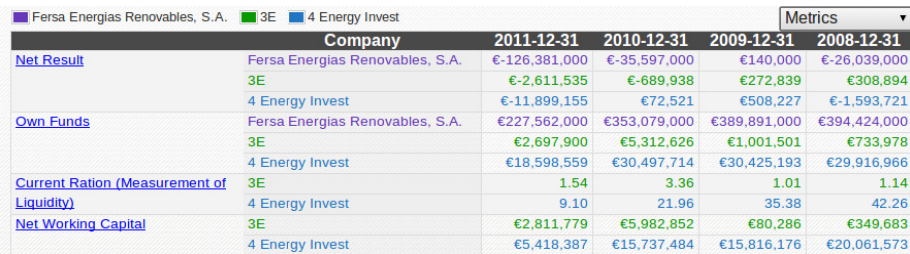


Fig. 5. Metrics Summary View

(e.g. Current Ration, Net Working Capital) calculated from the balance sheet of the XBRL filings aligned to the xEBR taxonomies. Here again the tabulated view allows quick side-by-side comparison of company figures or analysis ratios. The **BIXL demonstrator** is available from <http://monnet01.sindice.net:8080/financial>.

Acknowledgements. Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), and the EU FP7 MONNET project under Grant Agreement No. 248458.

References

1. Krieger, H.K., Declerck, T., Nedunchezian, A.K.: MFO - The Federated Financial Ontology for the MONNET Project. In: Proceedings of the 4th International Conference on Knowledge Engineering and Ontology Development, Barcelona, Spain (2012)
2. Cimiano, P., Montiel-Ponsoda, E., Buitelaar, P., Espinoza, M., Gomez-Peresz, A.: A note on ontology localization. Appl. Ontol. 5(2), 127–137 (2010)

3. Arcan, M., Federmann, C., Buitelaar, P.: Experiments with Term Translation in 24th International Conference on Computational Linguistics, Mumbai, India (2012)
4. Wunner, T., Buitelaar, P., O'Riain, S.: Semantic, Terminological and Linguistic Interpretation of XBRL. In: Workshop on Reuse and Adaptation of Ontologies and Terminologies, EKAW, Lisbon, Portugal (2010)
5. O'Riain, S., Edward, E., Harth, A.: XBRL and open data for global financial ecosystems: A linked data approach. IJAIS 13(2), 141–162 (2012)
6. Buitelaar, P., Choi, K.S., Cimiano, P., Hovy, E.: The Multilingual Semantic Web (Dagstuhl Seminar 12362) Dagstuhl Reports, Germany: 2/9, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2013)

R2RML by Assertion: A Semi-automatic Tool for Generating Customised R2RML Mappings

Luís Eufrazio T. Neto¹, Vânia Maria P. Vidal¹,
Marco A. Casanova², and José Maria Monteiro¹

¹Federal University of Ceará, Fortaleza, CE, Brazil
{vaniap.vidal, luis.eufrazio, jmmfilho}@gmail.com

²Department of Informatics – Pontifical Catholic University of Rio de Janeiro, RJ, Brazil
casanova@inf.puc-rio.br

Abstract. In this paper, we demonstrate the RBA (**R2RML By Assertion**) tool which automatically generates customized R2RML mappings based on a set of semantic mappings that model the relationship between the relational database schema and a target ontology in RDF. The semantic mappings are specified by a set of correspondence assertions, which are simple to understand.

Keywords: Linked Data, RDB-to-RDF user interface, R2RML mappings.

1 Introduction

The Linked Data initiative [1] promotes the publication of previously isolated databases as interlinked RDF triple sets, thereby creating a global scale data space, known as the Web of Data. However, the full potential of Linked Data depends on how easy it is to transform data stored in conventional, relational databases (RDB) into RDF triples. This process is often called RDB-to-RDF.

There are two main approaches for mapping RDB to RDF: direct mapping and customized mapping. The direct mapping approach relies on automatic methods that derive an ontology from a relational schema and transform data according to that schema. The customized mapping approach lets an expert create a mapping between the relational schema and an existing target ontology. In this approach, the RDB-to-RDF process can be divided into two steps: mapping generation and mapping implementation. The mapping generation step results in a specification of how to represent RDB schema concepts in terms of RDF classes and properties in a target vocabulary of the designer's choice. The mapping is conceptual and enables different implementation styles. For example, it can be used to materialize the RDF view or to offer virtual access through an interface that queries the underlying database.

In fact, quite a few tools that support customized mappings have been developed, such as Triplify, D2R Server [2], and OpenLink Virtuoso. Early surveys of RDB-to-RDF tools pointed out that the tools typically adopt different and proprietary mapping languages for the mapping process and do not provide a way to easily generate customized mappings between a RDB schema and a given domain ontology.

Recently, the W3C RDB2RDF Working Group proposed a standard mapping language, called R2RML [3], to express RDB-to-RDF mappings. However, R2RML is somewhat difficult to use, which calls for the development of tools to support the definition and deployment of mappings using R2RML. In this demo, we will show **RBA**, a tool that simplifies the task of generating and deploying customized R2RML mappings. The demo video is available at http://youtu.be/Un_UIrbHmqo.

2 Generating R2RML Mappings with RBA

R2RML is a language for expressing customized mappings from relational databases to RDF datasets. An R2RML mapping refers to logical tables to retrieve data from the input database. A logical table can be: (1) a base table; (2) a view; and (3) a valid SQL query (called an “R2RML view” because it emulates an SQL view without modifying the database). The use of views is a convenient solution to deal with complex mappings, which require data transformation, computation, or filtering before generating triples from the database.

In our approach, we use a three-level architecture for mapping RDB to RDF, which is depicted in Figure 1. The exported ontology (**EO**) models the RDF view exported by the data source. The vocabulary of the exported ontology is a subset of the target ontology vocabulary. The middle layer consists of a set of relational view schemas **VS**, where **VS** is a direct transformation for the exported ontology **E**. The view schemas in **VS** can be implemented as relational views or R2RML views.

In our framework, the use of the middle layer **VS**, help breaking the definition of the mappings into two stages: the definition of the *SQL mappings* and the definition of the *R2RML mappings*. The R2RML mapping from the views to the exported ontology is in fact one-to-one and is automatically generated based on **VS**. Also, it simplifies maintaining the R2RML mapping to reflect changes to the database schemas.

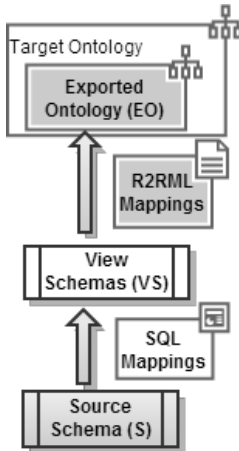


Fig. 1. 3-Level Schema Architecture

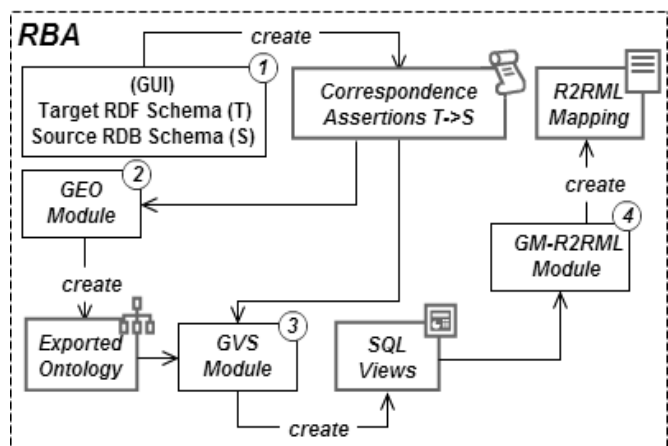


Fig. 2. RBA: main components

Figure 2 highlights the main components of *RBA*. The process for generating R2RML mappings with *RBA*, consists of four steps:

STEP 1: A session with *RBA* starts with the user loading a source and a target schema into the system. Then, the user can draw correspondence assertions (CAs) to specify the mapping between the target RDF schema and the source relational schema.

A CA can be: (i) a class correspondence assertion (CCA), which matches a class and a relation schema; (ii) an object property correspondence assertion (OCA), which matches an object property with paths (list of foreign keys) of a relation schema; or (iii) a datatype property correspondence assertion (DCA), which matches a datatype property with attributes or paths of a relation schema. CAs have a simple syntax and semantics, and yet suffice to capture most of the subtleties of mapping relational schemas into RDF schemas.

Figure 3 shows a snapshot of the GUI used for loading the schemas and defining the correspondence assertions. The target schema on the left contains 3 classes. The source relational schema on the right contains 3 tables. Given the two schemas, the user defines correspondence assertions from target to source.

As an example, consider the correspondence assertions shown in Figure 3:

CCA1: specifies that each tuple t in the *Authors* table produces an RDF triple (we omit the translations from attribute values to RDF literals for simplicity):

```
http://example.com/person/t.authorID rdf:type foaf:Person .
```

CCA2 specifies that each tuple t in *Papers* table produces the RDF triple:

```
<http://example.com/document/t.paperID> rdf:type foaf:Document .
```

DCA1 specifies that each tuple t in *Authors* produces the RDF triple:

```
<http://example.com/person/t.authorID> foaf:mbox "t.email" .
```

OCA1 specifies that for each pair $\langle t, t' \rangle$, where t is a tuple in *Papers*, t' is a tuple in *Authors*, and exists t'' in *Rel_Author_Paper* such that $t.paperID = t''.paperID$ and $t'.authorID = t''.authorID$, it generates one triple:

```
<http://example.com/document/t.paperID>
  dc:creator <http://example.com/person/t'.authorID> .
```

OCA1 is a little more complex, but it can be graphically defined. The user just navigates through the path by clicking over the FK nodes in the source schema and the *RBA* tool builds the assertions (see the text field at the bottom of Figure 3).

As we will show, step 1 is in fact the the only step which is not automatic.

STEP 2: The *GEO* (*Generate Exported Ontology*) module automatically generates the *exported ontology* (EO), which is induced by the correspondence assertions defined in Step 1. This task is very simple: GEO traverses the list of assertions created and includes in EO all classes and properties mapped. Figure 4 shows the exported ontology for the assertions in Fig 3.

STEP 3: The *GVS* (*Generate Views Schema*) module automatically generates the set of relational view schemas, which constitutes a direct transformation of the exported ontology generated in Step 2. The views schemas can be implemented as SQL views or R2RML views and *RBA* allows the user to decide the kind of views will be created; “Relational Views” or “R2RML Views” (see radio buttons in Figure 4). The first type creates the views in the source database and the second creates them directly in the R2RML mapping. In [4], we present an algorithm that automatically generates

the view schemas based on the correspondence assertions and exported ontology. Figure 5 shows the three SQL views generated for our case study.

STEP 4: The *GM-R2RML (Generate Mapping – R2RML)* module automatically generates a R2RML mapping from the views to the exported ontology. In [4], we present an algorithm that automatically generates the R2RML mappings from the CAs from the views to the exported ontology, which is one to one. The R2RML mapping is trivial, as shown in figure 6. After the creation of the R2RML mapping, probably the user needs to publish it to make some SPARQL queries over the RDF data. RBA emulates the D2RQ Server [2] with some customizations.

As discussed before, D2RQ has its own proprietary mapping language D2RM and until this moment there is not any production version that supports R2RML. To fill this gap, we developed a customized version of D2RQ (D2RQ-Ext) to support the main terms of R2RML and we embedded it as a component inside *RBA*. The limitation here is that D2RQ-Ext does not support “R2RML Views”, but it supports “Relational Views” which allows the publication of mappings when user chooses this kind of Views on Step 3.

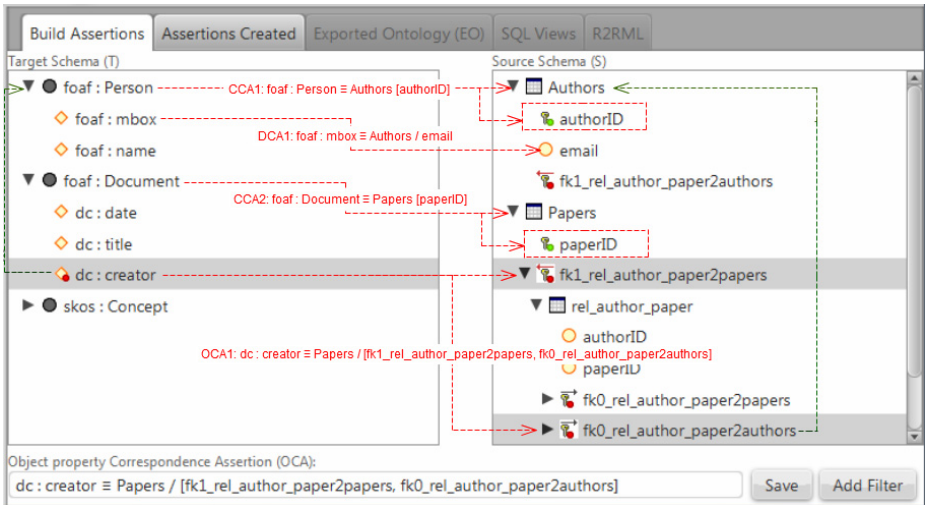


Fig. 3. RBA GUI snapshot

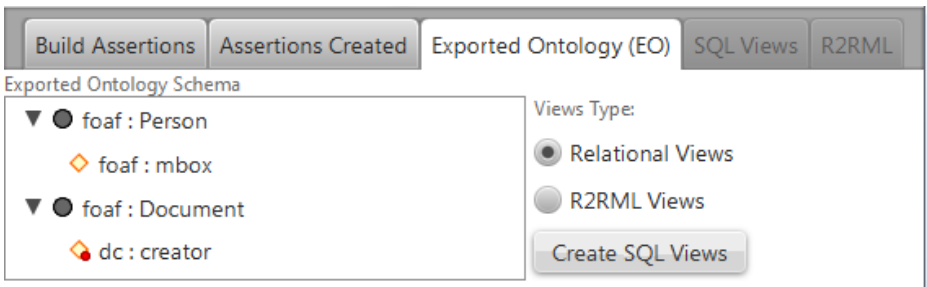


Fig. 4. Exported Ontology

```

CREATE OR REPLACE VIEW PERSON_VIEW AS
SELECT authors.AuthorID as ID, authors.Email as mbox
FROM authors;

CREATE OR REPLACE VIEW DOCUMENT_VIEW AS
SELECT papers.PaperID as ID
FROM papers

CREATE OR REPLACE VIEW DOCUMENT_CREATOR_VIEW AS
SELECT papers.PaperID as ID_DOCUMENT, authors.AuthorID as ID_PERSON
FROM papers, rel_author_paper, authors
WHERE papers.PaperID = rel_author_paper.PaperID
AND rel_author_paper.AuthorID = authors.AuthorID

```

Fig. 5. SQL Views

```

<#TriplesMapPerson>
  rr:logicalTable [ rr:tableName "PERSON_VIEW" ];
  rr:subjectMap [ rr:template "person/{ID}";
                  rr:class foaf:Person;];
  rr:predicateObjectMap [
    rr:predicate foaf:mbox;
    rr:objectMap [ rr:column "mbox" ];];
<#TriplesMapDocument>
  rr:logicalTable [ rr:tableName "DOCUMENT_VIEW" ];
  rr:subjectMap [ rr:template "document/{ID}";
                  rr:class foaf:Document;];
<#TriplesMapDocumentCreator>
  rr:logicalTable [rr:tableName "DOCUMENT_CREATOR_VIEW"];
  rr:subjectMap [rr:template "document/{ID_DOCUMENT}"];
  rr:predicateObjectMap [
    rr:predicate dc:creator;
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMapPerson>;
      rr:joinCondition [rr:child "ID_PERSON";
                       rr:parent "ID"];];];

```

Fig. 6. R2RML mapping

The demo video is available at http://youtu.be/Un_UirbHmqo. It shows how the *RBA* GUI helps the user graphically define CAs, and outlines the steps implemented by each module of the tool. It also shows some practical applications of the tool in a real world scenario. To the best of our knowledge, no commercial tool is available for generating customized R2RML mappings at the level of complexity of *RBA*.

References

1. Berners-Lee, T.: Linked Data (2006), <http://www.w3.org/DesignIssues/LinkedData.html>
2. Bizer, C., Cyganiak, R.: D2R Server – Publishing Relational Databases on the Semantic Web. In: ISWC (2006)
3. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language, W3C Working Draft (2012), <http://www.w3.org/TR/r2rml/>
4. Vidal, V., Casanova, M., Neto, L.: Towards Automatic Generation of R2RML Mappings (submitted, 2013), <http://goo.gl/fvZRt>

Tipalo: A Tool for Automatic Typing of DBpedia Entities

Andrea Giovanni Nuzzolese^{1,2}, Aldo Gangemi^{1,3}, Valentina Presutti¹,
Francesco Draicchio¹, Alberto Musetti¹, and Paolo Ciancarini^{1,2}

¹ STLab-ISTC Consiglio Nazionale delle Ricerche, Rome, Italy

² Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy

³ LIPN, Université Paris 13, Sorbone Cité, UMR CNRS, France

Abstract. In this paper we demonstrate the potentiality of Tipalo, a tool for automatically typing DBpedia entities. Tipalo identifies the most appropriate types for an entity in DBpedia by interpreting its definition extracted from its corresponding Wikipedia abstract. Tipalo relies on FRED, a tool for ontology learning from natural language text, and on a set of graph-pattern-based heuristics which work on the output returned by FRED in order to select the most appropriate types for a DBpedia entity. The tool returns a RDF graph composed of `rdf:type`, `rdfs:subClassOf`, `owl:sameAs`, and `owl:equivalentTo` statements providing typing information about the entity. Additionally the types are aligned to two lists of top-level concepts, i.e., Wordnet supersenses and a subset of DOLCE Ultra Lite classes. Tipalo is available as a Web-based tool and exposes its API as HTTP REST services.

1 Introduction

DBpedia [6] and YAGO [8] are, de facto, the two reference ontologies for DBpedia resources. Unfortunately, a large number of DBpedia resources is still untyped, or has a very specialized type, and types are taken from ontologies that have heterogeneous granularities or assumptions (e.g., 272 infobox-based types in the DBpedia ontology (DBPO) against almost 290,000 category-based in YAGO). While it is reasonable to have limited semantic homogeneity on the Web, it is highly desirable to bring a more organized and complete typing to DBpedia entities. Knowing what a certain entity is (e.g., a person, organization, place, instrument, etc.) is key for enabling a number of desirable functionalities such as type coercion [5], data pattern extraction from links [6], entity summarization (cf. Google Knowledge Graph), automatic linking, etc. In this paper we demonstrate how Tipalo and its Web API can be used for automatically typing DBpedia entities based on their natural language denitions as provided by their corresponding Wikipedia pages. Tipalo relies on FRED [7], a tool that implements deep parsing methods based on frame semantics for deriving RDF and OWL representations of natural language sentences. On top of FRED, Tipalo implements a set of graph-patterns-based heuristics that are used in order to extract the most appropriate types from the RDF representation derived from

the natural language definition of a DBpedia entity. Additionally, Tipalo gathers the correct senses of the extracted types by exploiting a word-sense disambiguation (WSD) tool, i.e., UKB [1], which automatically links them to WordNet’s synsets. Tipalo finally provides foundational grounding to extracted types by using an alignment between WordNet and two top-level ontologies: WordNet super senses, and a subset of DUL+DnS Ultralite ¹. We refer interested readers to [4] for a detailed description of the algorithm and the evaluation of Tipalo.

2 Related Work

The DBpedia project [6] and YAGO [8] are the most relevant approaches at typing DBpedia entities. DBpedia provides an ontology extracted from Wikipedia infoboxes based on hand-generated mappings of infoboxes to the DBpedia ontology. The DBpedia ontology counts of 359 concepts (version 3.8) but only 2.3M entities over more than 4M are classified with respect to this ontology. YAGO types are extracted from Wikipedia categories and aligned to a subset of WordNet. The YAGO ontology is larger than the DBpedia one and counts of 290K concepts, with a larger, but still incomplete, coverage of DBpedia entities. Other relevant work related to our method includes Ontology Learning and Population (OL&P) techniques [2]. Typically OL&P is implemented on top of machine learning methods, hence it requires large corpora, sometimes manually annotated, in order to induce a set of probabilistic rules. Such rules are defined through a training phase that can take a long time. The method presented in this paper differs from existing approaches, by relying on a component named FRED [7], which implements a logical interpretation of natural language represented based on Discourse Representation Theory (DRT). FRED is fast and produces an OWL-based graph representation of an entity description, including a taxonomy of types. We parse FREDs output graph, and apply a set of heuristics, so that we can assign a set of types to an entity in a very efficient way.

3 Tipalo

In this section we present an overview of the algorithm implemented by Tipalo and a scenario for demonstrating how to concretely use Tipalo. The purpose of this demo is to show the ability of Tipalo in gathering the most appropriate types of Wikipedia entities emerging from the interpretation of natural language definitions available in Wikipedia abstracts.

Our Approach. Tipalo is implemented as a pipeline of components and data sources as illustrated in figure 1. Each component in the pipeline implements a step of the computation: (i) extraction of an entity’s definition from its corresponding Wikipedia abstract; (ii) natural language deep parsing provided by FRED whose output is a RDF representation of the entity definition; (iii) selection of candidate types by means of the application of graph-pattern-based

¹ <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>

heuristics to FRED’s output; (iv) word-sense disambiguation of candidate types; and (v) type alignment to OntoWordNet [3], WordNet supersenses and to a subset of and DUL+DnS Ultralite.

More details about the pipeline and the evaluation of Tìpalo can be found in [4].

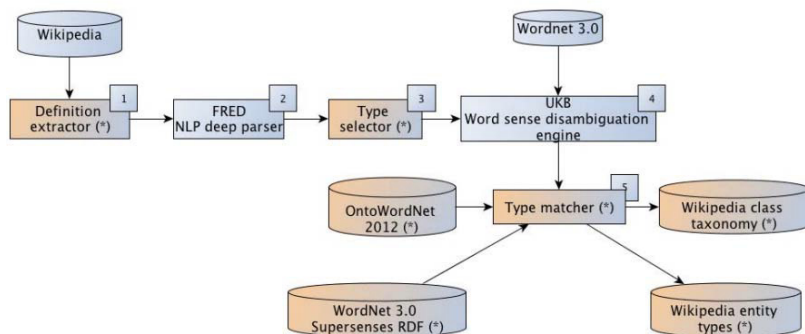


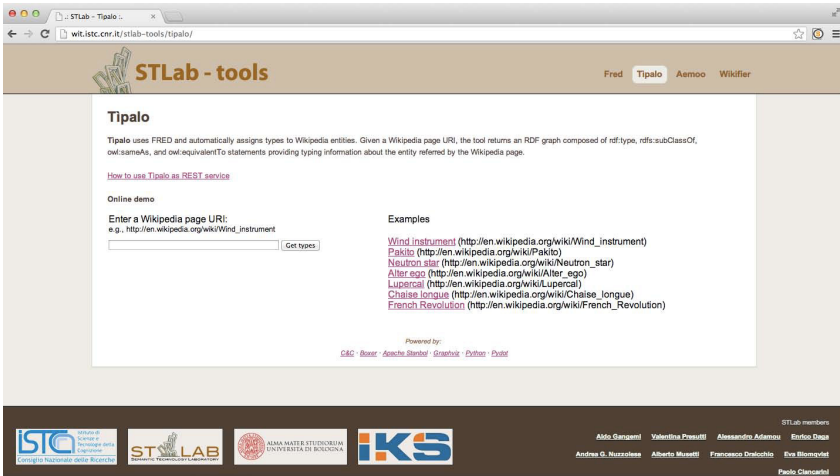
Fig. 1. The pipeline implemented by Tìpalo [4]

Tìpalo at Work. Tìpalo is available as a Web application. Figure 2(a) shows the on-line demo of Tìpalo ². Tìpalo demo works only on DBpedia resources: its input is a Wikipedia page URL (see the text input in the right side of the on-line demo in figure 2(a)). On the right side of the interface, users can find a list of samples to start from.

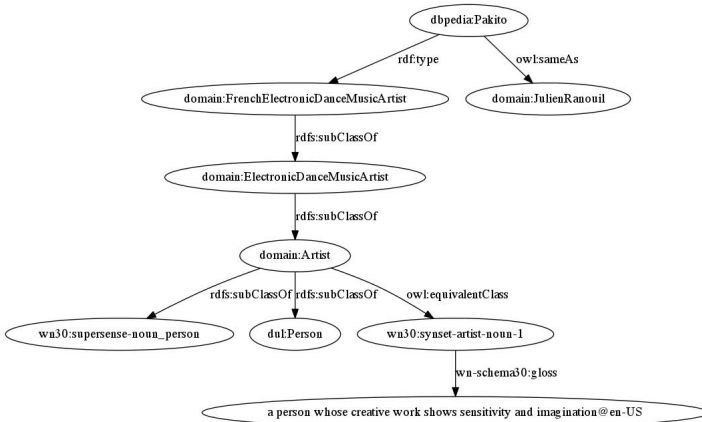
For demonstration purpose we show how to gather types for the entity `dbpedia:Pakito`. Once the Wikipedia URL of the entity has been provided in input to the system, Tìpalo returns (i) the natural language definition that has been used to infer the types of the entity by exploiting FRED and (ii) a RDF representation composed of `rdf:type`, `rdfs:subClassOf`, `owl:sameAs`, and `owl:equivalentTo` statements that provides typing information about the entity. In our example the definition that Tìpalo extracts from the Wikipedia abstract for `dbpedia:Pakito` is: “*Pakito is the alias of French Electronic Dance Music artist Julien Ranouil.*”. As can be observed in figure 2(b) the entity `dbpedia:Pakito` is typed as `domain:FrenchElectoronicDanceMusicArtist` ³. Tìpalo correctly recognizes the type `domain:FrenchElectoronicDanceMusicArtist` as subclass of the type

² The homepage of Tìpalo is available at <http://wit.istc.cnr.it/stlab-tools/tipalo>

³ The prefix `domain` can be customized to any namespace desired by the user. The prefixes `dbpedia`, `dul`, and `wn30` stand for <http://dbpedia.org/resource>, <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>, and <http://www.w3.org/2006/03/wn/wn30/instances/> respectively.



(a) The home page of Tipalo on-line at <http://wit.istc.cnr.it/stlab-tools/tipalo/>.



(b) The output of Tipalo for the resource **Pakito** derived from the natural language definition “*Pakito is the alias of French Electronic Dance Music artist Julien Ranouil.*”

Fig. 2.

domain:ElectroronicDanceMusicArtist, which is in turn recognized as subclass of domain:Artist. Tipalo also disambiguate each extracted type through WSD. For this task Tipalo uses UKB [1]. The IDs of WordNet synsets returned by UKB are resolved to corresponding OntoWordNet [3] URIs and are aligned to the extracted types by means of owl:equivalentClass axioms. In our example the type domain:Artist has been disambiguated with the OntoWordNet synset wn30:synset-artist-noun-1. This synset expresses the meaning “a person whose creative work shows sensitivity and imagination”. Finally, the entity

dbpedia:Pakito has been classified as owl:sameAs domain:JulienRanouil.
 Tpaló service is exposed as HTTP REST API ⁴.

4 Conclusions

We have presented Tipalo, a tool that formalizes entity definitions extracted from Wikipedia for automatically typing DBpedia entities and linking them to other DBpedia resources, WordNet, and foundational ontologies. As ongoing work, we are deploying the tool on a more robust cluster for improving time performances and experimenting on a large-scale resource such as the whole Wikipedia. The medium-term goal is to incrementally build a Wikipedia ontology that reflects the richness of terminology expressed by natural language, crowd sourced definitions of entities.

References

1. Agirre, E., Soroa, A.: Personalizing pagerank for word sense disambiguation. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009). The Association for Computer Linguistics, Athens (2009)
2. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer (2006)
3. Gangemi, A., Navigli, R., Velardi, P.: The ontowordnet project: extension and axiomatization of conceptual relations in wordnet. In: WordNet, Meersman, pp. 3–7. Springer (2003)
4. Gangemi, A., Nuzzolese, A.G., Presutti, V., Draicchio, F., Musetti, A., Ciancarini, P.: Automatic Typing of DBpedia Entities. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 65–81. Springer, Heidelberg (2012)
5. Kalyanpur, A., Murdock, J.W., Fan, J., Welty, C.: Leveraging community-built knowledge for type coercion in question answering. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 144–156. Springer, Heidelberg (2011)
6. Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - A Crystallization Point for the Web of Data. Journal of Web Semantics 7(3), 154–165 (2009)
7. Presutti, V., Draicchio, F., Gangemi, A.: Knowledge extraction based on discourse representation theory and linguistic frames. In: EKAW: Knowledge Engineering and Knowledge Management that Matters. Springer (2012) (to appear)
8. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th International World Wide Web Conference (WWW 2007). ACM Press, New York (2007)

⁴ Details about how to use Tipalo’s API can be found at <http://stlab.istc.cnr.it/stlab/tipalo>

Tracking and Analyzing The 2013 Italian Election

Vuk Milicic, José Luis Redondo García, Giuseppe Rizzo, and Raphaël Troncy

EURECOM, Sophia Antipolis, France

{vuk.milicic,redondo,giuseppe.rizzo,raphael.troncy}@eurecom.fr

Abstract. Social platforms open a window to what is happening in the world in near real-time: (micro-)posts and media items are shared by people to report their feelings and their activities related to any type of events. Such an information can be collected and analyzed in order to get the big picture of an event from the crowd point of view. In this paper, we present a general framework to capture and analyze micro-posts containing media items relevant to a search term. We describe the results of an experiment that consists in collecting fresh social media posts (posts containing media items) from numerous social platforms in order to generate the story of the “2013 Italian Election”. Items are grouped in meaningful time intervals that are further analyzed through deduplication, clusterization, and visual representation. The final output is a storyboard that provides a satirical summary of the elections as perceived by the crowd. A screencast showing an example of these functionalities is published at <http://youtu.be/jIMdnwMoWnk> while the system is publicly available at <http://mediafinder.eurecom.fr/story/elezioni2013>.

Keywords: Storytelling, Storyboard Creation, Visual Summarization, Topic Generation.

1 Introduction

The massive amount and steady increase of heterogeneous data shared on social platforms has attracted the interest of different research communities. Micro-posts such as status updates or tweets enable people to share their activities, feelings, emotions and conversations, opening a window to the world in real-time. Making sense out this amount of data is an extremely challenging task due to its heterogeneity (media items mixed with textual data) and dynamics making often short-lived phenomena. A growing number of commercial tools and academic research approaches try to partially collect and analyze this data in order to make sense of it. Capturing life moments and building narratives using social platforms is, for example, the goal of Storify¹ where the creators aim to investigate the interaction between event stories and the role of social networks that tell them: *(i)* sorting and organizing the items of an experience similar to the elements of a story, *(ii)* communicating and discussing strategies

¹ <http://storify.com>

on how to guide a user towards an intended experience. The overall storytelling creation is supervised by the user who composes a story based on streams of news coming through social platforms such as Twitter and YouTube. Generating the big picture from these streams is also the objective of Storyful². This application enables the user to navigate through the story created by other users or to create his own, aggregating content from different social platforms. While these two approaches position the role of a social platform as a container of fresh and breaking news items, they are leveraging on the user interaction that defines the summary creation as a supervised task. A disruptive innovation has been recently revealed by Mahaya³ which proposes an automatic crowd storyfication of the 12/12/12 concert⁴. In this example, the highlights of the concert corresponding to social media spikes when performers appeared are emphasized with images collected from Instagram, and microposts collected from Twitter. Inspired by the idea of automatic summarization through visual galleries, we focus more on the automatic sorting and clustering of media items for topic visualization. In [2], we proposed a generic media collector for retrieving media items that illustrate daily life moments shared on social platforms. In particular, we proposed a common schema in order to align the search results of numerous social platforms. This demonstration extends [1], adding to the codebase, the temporal feature to the cluster operations.

2 Use Case: The 2013 Italian Election

The 2013 Italian Election will be remembered as the *messy*⁵ Italian political election because of the absence of a clear winner. Such a result triggered lots of discussion on numerous Italian and international newspapers, and especially on different social platforms. We have tracked and analyzed media posts tagged as *elezioni2013* from 2013-02-26 to 2013-03-03. In particular, we have automatically extracted the main named entities in these posts and perform different automatic clustering operations. We have then compared those results with a baseline made of the facts occurring during those six days, taking as input the news appearing in the daily Italian and international newspapers.

2.1 Event Streaming Tracking

We performed an event stream processing from all media posts shared on the following social platforms: Twitter and its ecosystem (TwitPic, TwitterNative, MobyPicture, Lockerz or yfrog), GooglePlus and YouTube, Facebook and Instagram, Flickr and FlickrVideos. Leveraging on the search API of those platforms, we applied a cron job composed of different searches using the term *elezioni2013*.

² <http://storyful.com>

³ <http://mahaya.co>

⁴ <http://121212.mahaya.co>

⁵ <http://online.wsj.com/article/SB10001424127887323384604578325992879185934.html>

Each search operation proceeds as follows: first, each social platform is queried and the resulting microposts containing at least one media item (image or video) are collected. The output of those search queries result in a heterogeneous collection of items, varying in terms of serialization formats, schemas, data types, and topics (hidden or declared), that have been harmonized to a common schema. We applied a near-deduplication process to group microposts which contain the same image. By using the Hamming Distance over the discrete cosine similarity image fingerprints, we created a collection of items where each one is attached to a list of microposts that contain this media resource. Finally, for each micropost, we extract named-entity using the NERD framework [3]. A multi-lingual entity extraction is performed and the output result is a collection of entities annotated using the NERD Ontology⁶ and attached to each micropost. We repeated this search operation every 30 minutes for the 6 days.

2.2 Temporal Grouping

One of the key aspect of an event is the time dimension. Hence, we have sliced the *elezioni2013* timeline in 24 hours slots where the different social media posts were aligned to. Our assumption is that people react and share opinions and feelings following the daily newspapers reporting. Even though it is a simple assumption, it fits well (as we have observed in our use case) this particular political scenario. The period studied is therefore divided into six slots (one per day), and each search is aligned to the corresponding interval according to the time it was queried. We have then calculated the union between all the searches belonging to the same time interval and we have discarded the microposts published outside the boundaries of the interval. At the end of this step, social media posts were grouped in consecutive intervals of times which were relevant in the context of the current event.

2.3 Topic Generation

Once the entire set of social media posts is divided into chronologically ordered time intervals, we have further analyzed the details of every of those generated groups in order to automatically find the main highlights. For each temporal group, our approach identifies the topics that best describe the result set using clustering operations. Specifically, it implements four clustering methods working exclusively on the textual features from the microposts: (i) *named entity* based cluster algorithm which groups microposts according to the most frequent named entities extracted in the microposts; a further distance process is applied to compute the label similarity among the extracted entity. (ii) *named entity type* based cluster where microposts are grouped according to the dominant types of extracted named entities, such as Thing, Amount, Animal, Event, Function, Location, Organization, Person, Product, Time. (iii) *generative model* that extracts hidden topics from the large set of microposts collected, using the

⁶ <http://nerd.eurecom.fr/ontology/nerd-v0.5.n3>

latent Dirichlet allocation (LDA) model. *(iv) density* based cluster that exploits micropost proximity based on several micropost features such as temporal distance, text similarity and entity label similarity. Once the clustering operation is completed, we generate a Bag of Entities (BOE) that best describes the cluster while the most representative entity is disambiguated using a DBpedia URI⁷ and chosen to be the label of this cluster. Ultimately, the final output of this processing step is a set of clusters (limited to ten for visualization purpose) that best describes the topics extracted from one time interval. In our case study, we identified the following clusters at the end of this step: *Monti*, *Bersani*, *Italia*, *Berlusconi*, *Grillo*, and *Stelle* which basically correspond to the main entities and actors of the elections.

2.4 Visualization and Storyboarding

Once computed, these clusters are displayed according to a chronological timeline. The visualization of those results is crucial since the user has to be able to identify what are the main facts for a period of time for an event. We generate a stacked bar chart where the horizontal axis contains the time intervals (six days) and the vertical axis depicts the number of social media posts available. Each bar is decomposed into different portions that corresponds to a cluster inside this particular time range. To easily differentiate them, they are labeled and displayed using different colors. The user can, at anytime, show or hide a particular cluster in the diagram by interacting with the total list of clusters available in the left panel. There are two main indicators that help users to decide about the importance of a particular cluster: (i) minimal background knowledge about the context of the event attached to each portion (or cluster); (ii) the number of social media posts inside the cluster, and intuitively, if a cluster contains a higher number of items, it is because this topic is trending and it is therefore most probably important for the topic generation. Finally, by clicking on a particular cluster, a more detailed representation with the corresponding media items are shown, giving the user an insight about the relevance of that particular fact. This way of displaying an event can go further, being possible not only to figure out what is the most relevant fact for a particular day, but also to track the way this topic has evolved over the time of the event.

3 Discussion

During the first day (February 26th), no party was a clear winner of the election, but Monti's defeat stood out. International newspapers reported about it, fueling the discussion on social platforms. Similarly, the 27th is the day after Bersani's speech. His words about the dramatic situation became popular and viral. Another example of the usefulness of our automatic approach is the day 28th, when the world started to investigate about *The cult of Silvio Berlusconi* and the reaction from the crowd (caught by our storyboard) is immediately depicted. On

⁷ <http://dbpedia.org>

March 1st, the Italian president visited Germany and met the German chancellor Merkel to discuss, amongst others, the Italian elections. On March 2nd, Grillo claimed no collaboration with any party. Grillo is the *Movimento 5 Stelle* leader, incorrectly identified by our approach as *Stelle*. This is due to the lack of knowledge of our entity spotter. In the same day, both a famous Italian talk show and an Economist article triggered a lot of discussion about the current Italian situation, Grillo and Berlusconi. Again, Grillo and Berlusconi were trending. On March 3rd, the kick-off meeting of the Beppe Grillo's party took place. The main action lines were rejecting possible alliance with any parties and especially with the Bersani's party.

The election use case showed that our approach provides useful insights about events while leveraging from the crowd. In a wide sense, it provides a snapshot of how the crowd is experiencing an event. We focus on generating visual summaries in order to illustrate those phenomena. We argue that the need of human intervention in the collection and interpretation processes has been drastically reduced. The different clustering operations and the user interface give to the user a tool for interpreting the story, even if some background knowledge about the event is required. Although the crowd talked about the same main entities reported by daily newspapers, we observe that different satirical expressions for talking about politics are used. Actually, numerous images and posts are biased towards being satirical. Tracking and understanding an event from social data should also include the popularity of microposts. We have planned to use this measure as an additional mean for weighting the importance of a social peak. We will further demonstrate how this tool can be used to generate visual summaries of highly viral memes such as *Harlem Shake*.

Acknowledgments. This work was partially supported by the European Union's 7th Framework Programme via the project LinkedTV (GA 287911).

References

1. Milicic, V., Rizzo, G., Garcia, J.L.R., Troncy, R., Steiner, T.: Live Topic Generation from Event Streams. In: 22nd International World Wide Web Conference (WWW 2013), Rio de Janeiro, Brazil (2013)
2. Rizzo, G., Steiner, T., Troncy, R., Verborgh, R., Redondo García, J.L., Van de Walle, R.: What Fresh Media Are You Looking For? Retrieving Media Items from Multiple Social Networks. In: International Workshop on Socially-Aware Multimedia (SAM 2012), Nara, Japan (2012)
3. Rizzo, G., Troncy, R.: NERD: A Framework for Unifying Named Entity Recognition and Disambiguation Extraction Tools. In: 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012), Avignon, France (2012)

FRED: From Natural Language Text to RDF and OWL in One Click

Francesco Draicchio¹, Aldo Gangemi^{1,3},
Valentina Presutti^{1,2}, and Andrea Giovanni Nuzzolese^{1,2}

¹ STLab-ISTC Consiglio Nazionale delle Ricerche, Rome, Italy

² Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy

³ LIPN, Université Paris 13, Sorbone Cité, UMR CNRS, France

Abstract. FRED is an online tool for converting text into internally well-connected and quality linked-data-ready ontologies in web-service-acceptable time. It implements a novel approach for ontology design from natural language sentences. In this paper we present a demonstration of such tool combining Discourse Representation Theory (DRT), linguistic frame semantics, and Ontology Design Patterns (ODP). The tool is based on Boxer which implements a DRT-compliant deep parser. The logical output of Boxer enriched with semantic data from Verbnet or Framenet frames is transformed into RDF/OWL by means of a mapping model and a set of heuristics following ODP best-practice [5] of OWL ontologies and RDF data design.

1 Introduction

The problem of knowledge extraction from text is still only partly solved, this is particularly true if we consider it as a means for populating the Semantic Web. Being able to automatically and fastly produce quality linked data and ontologies from an accurate and reasonably complete analysis of natural language text would be a breakthrough: it would enable the development of applications that automatically produce machine-readable information from Web content as soon as it is edited and published by generic Web users.

Existing Ontology Learning and Population (OL&P) approaches can be used as drafting ontology engineering tools, but they show some limitations when used to produce linked data on the Web:

- they usually need a training phase, which can take a long time;
- their output form needs further, non-trivial elaboration to be put in a logical form;
- they only partially exploit OWL expressivity, since they typically focus on specific aspects e.g. taxonomy creation, disjointness axioms, etc.;
- in most cases, they lack implementation of ontology design good practices;
- linking to existing linked data vocabularies and datasets is usually left as a separate task or not considered at all.

In other words, existing tools focus mainly on helping users identifying key elements for ontology drafting, assuming that they would transform and substantially refine and enrich the output. Our approach instead focuses on producing ontologies and linked data ready for the Web.

We present a novel approach implemented in an online tool named FRED, which performs deep parsing of natural language, extracts complex relations based on Discourse Representation Theory (DRT), and produces linked data graphs.

An important aspect of OL&P is the design quality of the resulting ontology: this is related to representing the results in a logical form such as OWL by ensuring that some best-practices are followed.

Based on this consideration, we summarize a set of requirements that FRED follows in order to enable robust OL&P:

- ability to capture accurate semantic structures;
- representing complex relations;
- exploiting general purpose resources;
- no need of large-size domain-specific text corpora and training sessions;
- minimal time of computation;
- ability to map natural language to RDF/OWL representations.

2 Related Work

OL&P is concerned with the (semi-)automatic generation of ontologies from textual data (cf. [2]). Typical approaches to OL&P are implemented on top of Natural Language Processing (NLP) techniques, mostly machine learning methods, hence they require large corpora, sometimes manually annotated, in order to induce a set of probabilistic rules. Such rules are defined through a training phase that can take long time. OL&P systems are usually focused on either ontology learning (OL) for TBox production, or ontology population (OP) for ABox production. Examples of OL systems include [6], which describes an ontology-learning framework that extends typical ontology engineering environments by using semiautomatic ontology-construction tools, and Text2Onto [3], which generates a class taxonomy and additional axioms from textual documents. Examples of OP systems include: [9], which presents a GATE plugin that supports the development of OP systems and allows to populate ontologies with domain knowledge as well as NLP-based knowledge; [8] describes a weakly supervised approach that populates an ontology of locations and persons with named entities; [7] introduces a sub-task of OP restricted to textual mentions, and describes challenging aspects related to named entities. The method and tool (FRED) that we present in this paper differs from most existing approaches, because it does not rely on machine learning methods.

3 FRED at Work

In this section we present an overview of the pipeline implemented by FRED and an example of FRED execution. FRED reuses Boxer[1], a linguistic tool

based on DRT that generates formal semantic representations of text based on event semantics. Identifying the correct event in a sentence means identifying the frame behind it, which in turns leads to improve the design quality of the resulting ontology, as shown in [4]. This is motivated by the fact that frames match to ontology design patterns.

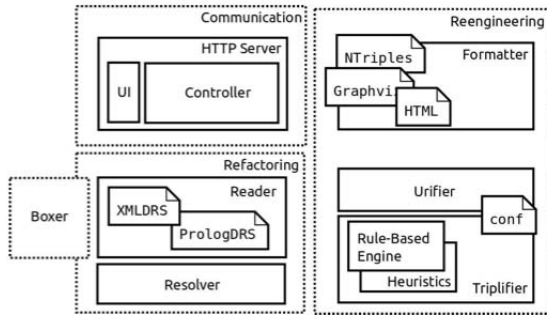


Fig. 1. Block architecture and workflow

Figure 1 depicts FRED architecture that is composed of four main components, implementing the following computational steps:

- capture user input text (e.g., via an HTML interface) and pass it to Boxer;
- retrieve Boxer XML output and transform it into a convenient data structure;
- run a collection of ad-hoc transformation rules and heuristics for producing OWL/RDF triples;
- transform and return triples into human as well as machine readable format.

Although Boxer produces a logical form from natural language - syntactically, lexically, and semantically - this logical form differ from RDF or OWL, and the heuristics that it implements for interpreting a natural language and transforming it to a DRT-based structure can be sometimes awkward when directly translated to ontologies for the Semantic Web. For this reason, FRED implements a set of rules for transforming Boxer output to OWL/RDF ontologies.

DRT construct	Boxer syntax	FOL construct	OWL construct
Predicate	pred(x)	Unary predicate ϕ	rdf:type
Relation	rel-name(x,y)	Binary relation	owl:ObjectProperty
Eq Rel	eq(x,y)	Identity	owl:sameAs
Named Entity	named(<var>, <name>, <type>)	Unary predicate ϕ	owl:NamedIndividual
Discourse Referent	<var>	Quantified Variable	(generated) owl:NamedIndividual
DRS	<drs> with event E	Proposition P with predicate ϕ_E	RDF graph G_P with class E
Negated DRS	not(<drs>)	Negated Proposition $\neg P$	G_P with NotE owl:disjointWith E

Fig. 2. A sample subset of quality ontology production heuristics

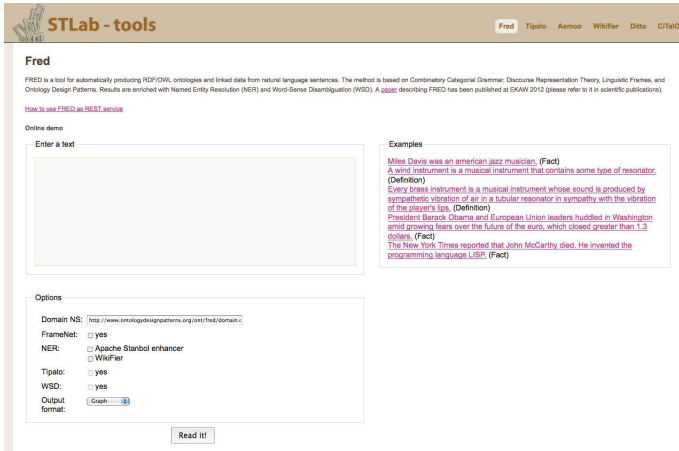


Fig. 3. Screenshot of FRED on-line demo

We distinguish two set of rules: (i) translation rules, which define global transformations from DRS constructs to OWL constructs (figure 2); and (ii) heuristics rules, which deal with adapting the results to the design needs of a Semantic Web ontology.

Demo. FRED on-line demo¹. (see Figure 3) provides a simple user interface that accepts a natural language text and returns a RDF/OWL graph either as a graphics or as a RDF serialization. A list of sample sentences are provided (on the right side) as a starting point for testing the tool. For demonstration purpose let us consider the following sentence “*The New York Times reported that John McCarthy died. He invented the language LISP.*”. By clicking on the “Read me!” button, a user would obtain the graph shown in Figure 4.

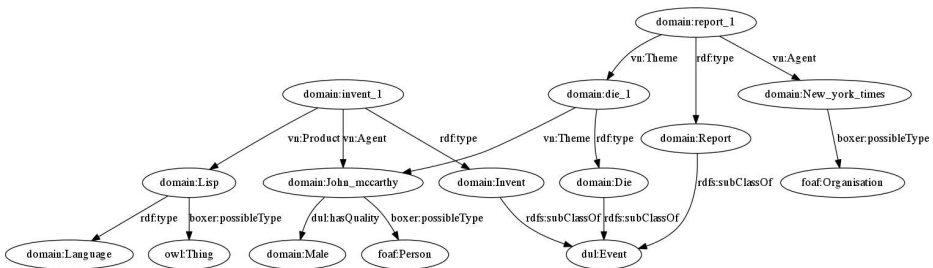


Fig. 4. Resulting graph for the sentence “The New York Times reported that John McCarthy died. He invented the language LISP.”

¹ FRED on-line demo <http://wit.istc.cnr.it/stlab-tools/fred>. FRED is exposed as HTTP REST API.

Two events (typed as DOLCE events) are correctly recognized: **Report**² and **Die**. An individual **John_mccarthy** of type **foaf:Person** is created, which has the role **vn:Theme**³ in the **Die** event, which in turn is the theme of the **Report** event performed by (role **vn:Agent**) the new created individual **New_york_times** of type **foaf:Organisation**. Additionally, FRED creates an individual for the language **LISP** and is able to recognize a third event **Invent** that involves the individual **John_mccarthy**, as agent this time, and **LISP**, as theme. This last situation is correctly represented by performing co-reference resolution.

4 Conclusion and Future Work

We have presented FRED, a novel implemented method that transforms natural language texts to OWL/RDF according to a frame-based design approach. Currently we are working at a rigorous evaluation of the resulting ontology and linked data.

References

1. Bos, J.: Wide-Coverage Semantic Analysis with Boxer. In: Bos, J., Delmonte, R. (eds.) *Semantics in Text Processing. STEP 2008 Conference Proceedings. Research in Computational Semantics*, vol. 1, pp. 277–286. College Publications (2008)
2. Cimiano, P.: *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer (2006)
3. Cimiano, P., Vlker, J.: *Text2onto - a framework for ontology learning and data-driven change discovery* (2005)
4. Coppola, B., Gangemi, A., Gliozzo, A., Picca, D., Presutti, V.: Frame detection over the semantic web. In: Aroyo, L., et al. (eds.) *ESWC 2009. LNCS*, vol. 5554, pp. 126–142. Springer, Heidelberg (2009)
5. Gangemi, A., Presutti, V.: *Ontology Design Patterns*. In: Staab, S., Studer, R. (eds.) *Handbook on Ontologies*, 2nd edn., Springer (2009)
6. Maedche, A., Staab, S.: *Ontology learning for the semantic web*. *IEEE Intelligent Systems* 16, 72–79 (2001)
7. Magnini, B., Pianta, E., Popescu, O., Speranza, M.: *Ontology population from textual mentions: Task definition and benchmark*. In: *Proceedings of the OLP2 workshop on Ontology Population and Learning*, Sidney, Australia (2006)
8. Tanev, H., Magnini, B.: *Weakly supervised approaches for ontology population*. In: *Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, Amsterdam, The Netherlands, pp. 129–143. IOS Press (2008)
9. Witte, R., Khamis, N., Rilling, J.: *Flexible ontology population from text: The owl exporter*. In: Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., Tapias, D. (eds.) *LREC. European Language Resources Association* (2010)

² We omit the prefix **domain:** as it is the local namespace and can be customized according to users needs.

³ prefix **vn:** refers to VerbNet.

Trusted Facts: Triplifying Primary Research Data Enriched with Provenance Information[★]

Kai Schlegel, Sebastian Bayerl, Stefan Zwicklbauer,
Florian Stegmaier, Christin Seifert, Michael Granitzer, and Harald Kosch

University of Passau, Germany
Innstrasse 41, 94032 Passau
{forename.surname}@uni-passau.de

Abstract. A crucial task in a researchers' daily work is the analysis of primary research data to estimate the evolution of certain fields or technologies, e.g. tables in publications or tabular benchmark results. Due to a lack of comparability and reliability of published primary research data, this becomes more and more time-consuming leading to contradicting facts, as has been shown for ad-hoc retrieval [1]. The CODE project [2] aims at contributing to a Linked Science Data Cloud by integrating unstructured research information with semantically represented research data. Through crowdsourcing techniques, data centric tasks like data extraction, integration and analysis in combination with sustainable data marketplace concepts will establish a *sustainable, high-impact ecosystem*.

1 Triplifying Primary Research Data

This paper demonstrates a triplification processing chain for semantic lifting of primary research data. To follow the idea of a Linked Science Data Cloud, data must be openly accessible by sophisticated retrieval possibilities to foster a dynamic interaction by the community. Figure 1 highlights essential components of this process from a conceptional point of view. If the primary research data in scope is encapsulated in unstructured data sources, a preprocessing step extracts it. For a unified view on the data, HTML table specification serves as pivot format during the semantic enrichment phase. This phase is forked in two parts: disambiguation of existent concepts and the description of the overall table structure, e.g., its dimensions or its measure type (e.g., nominal or ordinal). Both parts are crowdsourced to manage impreciseness of automatic routines applied. The initial extracted table will be enriched and annotated with semantic information using a proprietary microformat¹. The enriched table will be finally transformed into OLAP compliant data cubes using the W3C RDF Data Cube Vocabulary². Publication is handled via a linked data endpoint following the five star deployment scheme³.

[★] The presented work was developed within the CODE project funded by the EU Seventh Framework Programme, grant agreement number 296150.

¹ <http://www.code-research.eu/dataextraction/microformat>

² <http://www.w3.org/TR/vocab-data-cube/>

³ <http://www.5stardata.info/>

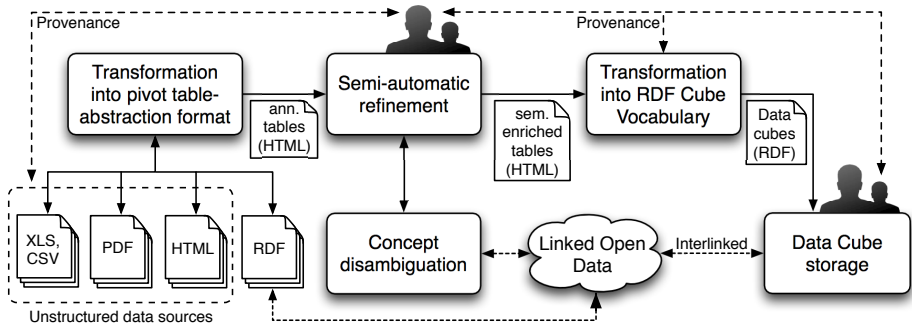


Fig. 1. Conceptual overview of data triplication chain

2 Establishment of Data Provenance Chains

A Linked Data aware publication of primary research data is not the end of the story due to arising questions, such as: Who generated the data? Who interacted with it? The answer to these questions lead to the definition of data provenance chains enabling justification of data with respect to its impact and quality fostering trust between peers in a marketplace. As shown in Figure 1, such information is omnipresent within the triplication pipeline. To model these chains, three atomic objects are distinguished: An *entity* describes the objects whose provenance should be specified. Interaction with an entity is modeled by an *action*. Finally, the *agent* points out who is responsible for the action. Those information are then aggregated and stored by the W3C PROV ontology⁴. To add workflow specific semantics to PROV-O, the proprietary CODE Provenance Vocabulary⁵ can be used.

3 Prototypic Implementation

The outlined processing chain has been implemented in the CODE Data Extraction prototype hosted at the University of Passau⁶. In the future this service will be combined with the *Testbed for Information Retrieval Algorithms (TIRA)* [3] to manage the evaluation results of CLEF challenges.

References

1. Armstrong, T.G., Moffat, A., Webber, W., Zobel, J.: Improvements that don't add up: ad-hoc retrieval results since 1998. In: Proceedings of the Conference on Information and Knowledge Management, pp. 601–610 (2009)

⁴ <http://www.w3.org/TR/prov-o/>

⁵ <http://www.code-research.eu/ontology/code-prov-vocabulary>

⁶ Further details at <http://www.code-research.eu/results/data-extractor>

2. Stegmaier, F., Seifert, C., Kern, R., Höfler, P., Bayerl, S., Granitzer, M., Kosch, H., Lindstaedt, S., Mutlu, B., Sabol, V., Schlegel, K., Zwicklbauer, S.: Unleashing semantics of research data. In: Proceedings of the 2nd Workshop on Big Data Benchmarking (2012)
3. Gollub, T., Stein, B., Burrows, S., Hoppe, D.: Tira: Configuring, executing, and disseminating information retrieval experiments. In: Proceedings of the 23rd International Workshop on Database and Expert Systems Applications, pp. 151–155 (2012)

Semantic Hyperlocal Search for Parlance Mobile Spoken Dialogue System

Panos Alexopoulos¹, Marie-Aude Aufaure², Nesrine Ben-Mustapha²,
Hugues Bouchard³, José Manuel Gomez-Pérez¹, James Henderson⁴,
Beibei Hu², Joel Lang⁴, Peter Mika³, and Yves Vanrompay²

¹ iSOCO Madrid Spain

² Ecole Centrale Paris

³ Yahoo! Research Barcelona

⁴ University of Geneva

1 Introduction

Current spoken dialogue systems (SDS) for mobile search are mostly domain-specific and make use of static knowledge. Consequently they do not take into account the interests, location and contextual situation of the concrete user. We propose PARLANCE, which is a more dynamic and personalized SDS that incorporates 1) a dynamic knowledge base consisting of modular ontologies that are enriched incrementally with information extracted from the Web; 2) and an evolving user profile. This allows the system to provide answers that are more tailored to the concrete user and to exploit the Web as a source of information, which can improve the quality of experience for the user. The PARLANCE SDS aims to guide the user in his search for information by providing answers that are: (1) Hyperlocal: The current geographical location of the user is taken into account to provide points of interest (POIs) in the neighborhood; (2) Dynamic: New concepts and entities are learned at runtime and included in the appropriate modular ontologies; (3) Personalized: Potential relevant answers adapted to user's queries are selected and ranked according to user preferences. Complementary, a form of social search is performed by looking at interests of similar user in the neighborhood (i.e. collaborative filtering). The central component in the PARLANCE architecture is the Interaction Manager (IM) which probabilistically decides on the most appropriate next answer to be provided to the user. The IM exploits information from the Semantic Web by interacting with the Knowledge Base (KB), the Web Content Analyzer (WCA) and the Local Search (LS) components, which will be detailed in the next section.

2 Hyperlocal, Dynamic and Personalized Spoken Dialogues

Modular Ontologies and User Profile. The Knowledge Base consists of the Ontology Manager (OM) and User Model (UM) and is informed by the Web Context Analyzer. The OM is responsible for the construction and enrichment of the

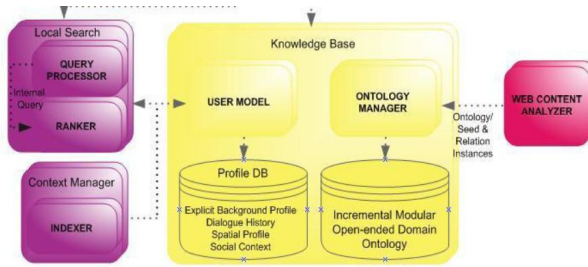


Fig. 1. Hyperlocal semantic search components of PARLANCE

modular ontologies in the form of xsd-schemas specifying RDF documents. In addition, we propose a method of ontology enrichment by searching web snippets related to each entity associated with a core ontology module schema, focusing on one attribute at a time. Web snippets are considered as short sentences, which can be parsed to retrieve linguistic patterns by the WCA. These patterns are then ranked according to web-based co-occurrence measures by querying the Web using the extracted linguistic patterns. The User Model contains user interests, social and contextual information that is exploited to provide tailored answers to the user.

Web Content Analyzer. The goal of the WCA is to extract information from web content. The extracted information is written to the Knowledge Base. Specifically, the WCA is concerned with extracting relation tuples for relations in the domains of interest. In our approach the target concept (either an entity class or a relation) is defined in terms of a relatively small number of seed instances (seed set), which are then used to extract further instances belonging to the concept. We developed a novel graph-based seed set expansion method, in which candidate instances are ranked according to their distance/similarity to the seed instances in the graph, where distance is measured in terms of truncated random walk hitting times.

Local Search. The LS consists of an Indexer, Query Processor and Ranker. The ontology provides information at index time to the Indexer to determine the metadata fields to be applied to the POIs, for example, that the POI is a restaurant, which serves Chinese food and is cheap. The Query Processor determines the geographical scope and personalization and the index is organized into small geographical partitions, populated with data instances and embedded with metadata associated with the Knowledge Base. The Indexer builds the necessary index structures for scalable matching and ranking. The Indexer is an offline component that builds an index from RDF data on disk, which is used at runtime for matching results. Thus the Indexer does not support dynamic updates of the data or the schema but instead rebuilds the index periodically from the data feeds received from the WCA. The Ranker performs runtime ranking based on the internal query received from the Query Processor. The goal of the Ranker is to order the results retrieved from the index according to the relevance of the original query.

Representation of Complex Expressions in RDF

Sébastien Ferré

IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
ferre@irisa.fr

Abstract. Complex expressions, as used in mathematics and logics, account for a large part of human knowledge. It is therefore desirable to allow for their representation and search in RDF. We propose an approach¹ that fulfills three objectives: (1) the accurate representation of expressions in standard RDF, so that expressive search is made possible, (2) the automated generation of human-readable labels for expressions, and (3) the compatibility with legacy data (e.g., OWL/RDF, SPIN).

1 Introduction

Complex expressions account for a large part of human knowledge. Common instances of expressions are mathematical equations, logical formulae, regular expressions, or parse trees of natural language sentences. In the domain of the Semantic Web, they can be OWL axioms, SWRL rules, or SPARQL queries. It is therefore desirable to allow for their representation in RDF so that they can be mixed with other kinds of knowledge. For example, it should be possible to describe a theorem by its author, its discovery date, its informal description as a text, and its formal description as a mathematical and logical expression, all in RDF. An expected advantage of the formal representation of expressions is the ability to search those expressions by their content: e.g., *mathematical search* [1]. For example, we want to retrieve all expressions that are an integral in some variable x and whose body contains the sub-expression x^2 . Correct answers are $\int x^2 + 1 dx$ and $\int y^2 - y dy$. This example exhibits two difficulties in expression search: (1) to take into account the nested structure of expressions (x^2 is in the scope of the integral), (2) to abstract over the name of bound variables (x is bound by the integral $\int dx$).

Textual search methods that work by linearizing expressions cannot correctly account for the above difficulties [1]. In the above example, a textual search would have false positives such as $\int 2x dx = x^2 + c$ (x^2 is not in the scope of f), and would have false negatives such as $\int y^2 - y dy$ (y instead of x). On the contrary, structured query languages [1] correctly account for them by reasoning directly on the structure of expressions, and by using *jokers* as place-holders for variables and sub-expressions. However, those query languages are limited to mathematical expressions, and are not interoperable with Semantic Web languages.

¹ A long version of this paper is available at <http://hal.inria.fr/hal-00812197>

A number of RDF vocabularies have been defined to represent complex expressions with blank nodes and *ad-hoc* classes and properties: e.g., OWL/RDF for OWL class expressions, SPIN for SPARQL queries, and Robbins' proposal [4] for MathML strict content. Each of them is good in isolation, but they kind of reinvent the wheel each time, and a semantic web tool would have to be configured for each of them in order to nicely render the different kinds of expressions. A number of proposals have been made to use RDF for Mathematical Knowledge Management (MKM) [3], but few of them allow for complete RDF representations. In addition to Robbins' proposal, N3 has a syntax for expressions but its goal is to extend RDF in a non-standard way to express *computations*, whereas we are interested in the representation of *syntax trees* in standard RDF.

We propose to represent expressions as RDF containers, using custom *constructors* in addition to `rdf:Seq`, `rdf:Bag`, `rdf:Alt`: e.g., `[a math:Power; rdf:_1 _x; rdf:_2 2]` represents the expression x^2 , using constructor `math:Power`. With a simple syntactic extension of Turtle and SPARQL for containers, the formula $\int x^2 + 1 dx$ can be concisely represented as `math:Integral(math:Plus(math:Power(_:x,2),1),_:x)`; and the query that retrieves integrals in x whose body contains x^2 can be represented as `SELECT ?e WHERE { ?e is math:Integral(...math:Power(?x,2)... ,?x) }` By annotating constructors with notation expressions (e.g., `math:Plus expr:hasNotation expr:LeftAssociativeInfixOperator("+",math:PlusPriority)`), *human-readable labels* can be automatically generated for each blank node representing an expression: e.g., `"∫ x2+1 dx"` for the above example. To allow for the generation of such labels for legacy data, patterns of *ad-hoc* classes and properties can be mapped to *implicit constructors*. For example, OWL existential restrictions `[a owl:Restriction; owl:onProperty ?p; owl:someValuesFrom ?c]` can be mapped to expressions `owl:Some(?p,?c)`, by defining the implicit constructor `owl:Some: owl:Some expr:hasImplicitClass owl:Restriction ; expr:hasImplicitProperties rdf:Seq(owl:onProperty,owl:someValuesFrom)`. From there, both DL and Manchester notations can be generated as labels.

The above ideas have been implemented in Sewelis [2], a Semantic Web tool for the guided exploration and edition of RDF graphs, and applied to math formulas and OWL axioms. Expressions are only displayed and edited through their human-readable form (generation of labels), and can be searched through expressive query-based faceted search [2].

References

1. Altamimi, M.E., Youssef, A.S.: A math query language with an expanded set of wildcards. *Mathematics in Computer Science* 2(2), 305–331 (2008)
2. Ferré, S., Hermann, A.: Semantic search: Reconciling expressive querying and exploratory search. 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. 177–192. Springer, Heidelberg (2011)
3. Lange, C.: Ontologies and languages for representing mathematical knowledge on the semantic web. *Semantic Web Journal* (to appear)
4. Robbins, A.: *Semantic MathML* (2009), <http://straymindcough.blogspot.fr/2009/06/>

Representing and Querying Negative Knowledge in RDF

Fariz Darari*

Faculty of Computer Science, Free University of Bozen-Bolzano
Dominikanerplatz 3, Bozen-Bolzano, Italy
fadirra@gmail.com

Abstract. Typically, only positive data can be represented in RDF. However, negative knowledge representation is required in some application domains such as food allergies, software incompatibility and school absence. We present an approach to represent and query RDF data with negative data. We provide the syntax, semantics and an example. We argue that this approach fits into the open-world semantics of RDF according to the notion of certain answers.

1 Syntax

We present the syntax for representing positive and negative RDF data using Turtle serialization. We represent positive data (s, p, o) and negative data $\text{not}(s, p, o)$ as reified statements of types `:posStatement` and `:negStatement` respectively¹, and they have properties `:subj`, `:pred` and `:obj` with the corresponding values s, p and o . In our work, s, p and o cannot be blank nodes.

Since negative knowledge can be represented, it is possible for data to be inconsistent. Thus, inconsistency is defined when the intersection between positive and negative data is not empty.

There will be a pre-processing step to de-reify positive and negative RDF data, and put it into the graphs `:posGraph` and `:negGraph` respectively, for querying.

As for the SPARQL syntax, we focus on SELECT queries with BGP (Basic Graph Pattern), which is of the form $(\text{SELECT } W \ P)$, where W is a set of variables and P is a set of positive (of the form (s, p, o)), and negative triple patterns (of the form $\text{not}(s, p, o)$), which are the new feature.

2 Semantics

We present the semantics for positive and negative RDF data. The positive RDF data has the semantics as in W3C's RDF Specification². Moreover, following the specification, we define the semantics of negative RDF data as: if E is a ground negative

* The author was supported by the European Master's Program in Computational Logic and the project Querying Incomplete Data (QUID) funded by the FU Bozen-Bolzano.

¹ We assume the default RDF namespace is <http://example.com/>

² <http://www.w3.org/TR/rdf-mt/>

triple $\text{not}(s, p, o)$, then $I(E) = \text{true}$ if s, p and o are in V , $I(p)$ is in IP and $\langle I(s), I(o) \rangle$ is not in $IEXT(I(p))$, otherwise $I(E) = \text{false}$.

As for the semantics of SPARQL, during evaluation, all positive triple patterns in the body of queries will be delegated into positive graph `:posGraph`, whereas all negative triple patterns will be delegated into negative graph `:negGraph` using GRAPH graph patterns. Then, the transformed SPARQL query is evaluated using the standard SPARQL semantics [1]. We provide an example in Section 3.

Theorem. For a SPARQL query Q with positive and negative triple patterns and a consistent RDF data D , the evaluation of query Q over D gives certain answers over all models of D .

Proof Idea. The proof is based on the semantics of BGP and GRAPH graph pattern that will give certain answers [1], with the idea that for every negative triple $\text{not}(s, p, o)$, then it must be the case that in all models of D , s, p and o are in V , $I(p)$ is in IP and $\langle I(s), I(o) \rangle$ is not in $IEXT(I(p))$. This is reflected by the evaluation of negative query part into all the triples in `:negGraph` that gives certain answers.

3 Example

Consider an example in the food allergies domain. Let D be RDF data containing $\{(john, \text{eats}, \text{egg}), (john, \text{eats}, \text{nut}), (tom, \text{eats}, \text{egg}), \text{not}(john, \text{eats}, \text{fish})\}$. Given a query $Q = \text{SELECT } \{?x\} \{(?x, \text{eats}, \text{egg}), \text{not}(?x, \text{eats}, \text{fish})\}$, the evaluation of Q over D gives $\{?x/john\}$.

Regarding the implementation, for example, the negative triple $\text{not}(john, \text{eats}, \text{fish})$ will be represented as $\{(_:b1 \text{ a } :negStatement), (_:b1 :subj :john), (_:b1 :pred :eats), (_:b1 :obj :fish)\}$ and thus, after dereification, the triple `:john :eats :fish` is stored in the graph `:negGraph`. Also note that the query Q during evaluation has the following form:

```
SELECT ?x WHERE { GRAPH :posGraph {?x :eats :egg}
                  GRAPH :negGraph {?x :eats :fish}}
```

4 Future Work

For future work, we will investigate the roles and interaction of blank nodes, RDF Schema and more expressive SPARQL queries with respect to negative RDF data.

Reference

1. Arenas, M., Gutierrez, C., Pérez, J.: On the Semantics of SPARQL. In: De Virgilio, R., Giunchiglia, F., Tanca, L. (eds.) Semantic Web Information Management: A Model Based Perspective. Springer (2010) ISBN: 978-3-642-04328-4

The Semantic Evolution of General and Specific Communities

Matthew Rowe¹ and Claudia Wagner²

¹ School of Computing and Communications, Lancaster University, Lancaster, UK

² Institute for Information and Communication Technologies,

JOANNEUM RESEARCH, Graz, Austria

m.rowe@lancaster.ac.uk, claudia.wagner@joanneum.at

Abstract. Content injection methods rely on understanding community dynamics (i.e. attention factors) in order to publish content that community users will engage with (e.g. product-related posts), however such methods require re-training should the community’s discussed topics change. In this paper we present an examination of the semantic evolution of community forums by measuring the topical specificity of online community forums and then tracking changes in the concepts discussed within the forums over time. Our results indicate that general discussion communities tend to diverge in their semantics, while topically-specific communities do not. These findings inform content injection methods on model longevity and the need for adaptation for general communities.

1 Introduction

The life cycles of online communities have been examined in [2], where the authors identified disparate evolution stages (e.g. *creation, growth, etc.*); and in [1], where Mozilla dev communities exhibited such stages. Although such works examine the changing nature of online communities, they do not examine how the topics of online communities evolve over time. Previously [3], we found a relation between the topical specificity of online communities and their attention patterns: single topic communities (e.g. Golf) require content to match this topic exactly. Therefore changes in a community’s topics would require any models (e.g. content injection) that rely on the topical dynamics to be re-adapted. In this paper we examine the following question: *How is the topical specificity of an online community forum related to its evolution?* First, we describe a method to measure the topical specificity of community forums, before second, explaining how the community specificity is related to changes in discussed concepts.

2 Measuring Topical Specificity

For our analysis we used a dataset of 230 community forums from the Irish community message board Boards.ie.¹ Our method for measuring the topical

¹ <http://www.boards.ie>

specificity of community forums took all posts published during a one-week window from 23/3/2005² and extracted entities using Zemanta.³ We then used the DBpedia Ontology to identify the *type* of each entity to provide the set of concepts for a given forum: $A_f^{t't''}$. We measured the specificity of each community forum using different combinations of abstraction measures (Network Entropy, Degree Centrality, Eigenvector Centrality, Hits Authority/Hub Score, Statistical Subsumption, Key Player Problem) and composite functions (Most Specific Concept, Mean Specificity, Most Frequent Concept, Concept Frequency-Inverse Forum Frequency) that function over the DBpedia Ontology represented as a concept graph. For each model, we ranked the forums by their specificity scores and compared this to the ground truth ranking of forums ordered by hierarchical levels. We found the best model to be Concept Frequency with Eigenvector Centrality (Kendall $\tau_b = 0.075$), surpassing the random Knuth Shuffle baseline.

3 Semantic Evolution

To examine the relation between topical specificity and semantic evolution we divided the forums up into 10-equal frequency bins based on their specificity value (derived using the above model) and selected forums in the top and bottom bins to form *high* and *low*-specificity forums respectively. For these forums (23 in each bin) we randomly chose 4 weekly periods following 30/03/2005, extracted entities from forum posts during these periods, and derived concepts using the DBpedia Ontology. We then derived a concept frequency vector \mathbf{c} for each forum for the four randomly-inspected analysis periods: $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}$, and calculated the cosine similarity of the concept vectors between consecutive time periods (i.e. $\text{cosine}(\mathbf{c}_i, \mathbf{c}_{i+1})$). We examined the differences between the cosine similarity distributions of the *low* and *high* forums and found a significant difference in their means ($\mu_{low} = 0.783$ and $\mu_{high} = 0.854$ with $p < 0.05$ using Student's t-test). This result indicates that the initial specificity of a community forum is related to changes in concepts: *general communities exhibit greater semantic drift than specific communities*. Such results inform content injection models on the potential changes in topical dynamics and for model adaptation.

References

1. Hong, Q., Kim, S., Cheung, S.C., Bird, C.: Understanding a developer social network and its evolution. In: Proceedings of the 2011 27th IEEE International Conference on Software Maintenance, ICSM 2011, pp. 323–332. IEEE Computer Society, Washington, DC (2011)
2. Iriberry, A., Leroy, G.: A life-cycle perspective on online community success. *ACM Comput. Surv.* 41(2), 11:1–11:29 (2009)
3. Wagner, C., Rowe, M., Strohmaier, M., Alani, H.: Ignorance isn't bliss: an empirical analysis of attention patterns in online communities. In: AES Conference on Social Computing (2012)

² The median number of posts (4,455) were published on Boards.ie on this date.

³ <http://www.zemanta.com>

Experiments Varying Semantic Similarity Measures and Reference Ontologies for Ontology Alignment

Valerie Cross, Prमित Silwal, and Xi Chen

Computer Science and Software Engineering Department,
Miami University, Oxford, OH 45056
crossv@miamioh.edu

Abstract. Semantic similarity measures within a reference ontology have been used in a few ontology alignment (OA) systems. Most use a single reference ontology, typically WordNet, and a single similarity measure within it. The mediating matcher with semantic similarity (MMSS) was added to AgreementMaker to incorporate the selection of a semantic similarity measure and the combination of multiple reference ontologies in an adaptable fashion. The results of experiments using the MMSS on the anatomy track of the Ontology Alignment Evaluation Initiative (OAEI) are reported. A variety of semantic similarity measures are applied within multiple reference ontologies. Using multiple reference ontologies with the MMSS improved alignment results. All information-content based semantic similarity measures produced better alignment results than a path-based semantic similarity measure.

The AgreementMaker OA system [1] has a variety of matchers configured in a hierarchical linear weighted combination (LWC) with weights derived from quality measures on their produced mappings. In OAEI 2011, AgreementMaker introduced the mediating matcher (MM) with Uberon as a reference ontology but did not use semantic similarity measures within Uberon. In [2] the MMSS replaces the MM in its OAEI 2011 configuration. Their performances are compared on the anatomy track running both on the same cluster computer. The MM result in Table 1 differs some from that reported in OAEI 2011. With a threshold of 0.90 for semantic similarity between concepts, the same number of correct mappings was produced for both MM and MMSS as shown in Table 1. Investigating the alignment results showed although the same number is correct, they were different ones. More experiments revealed the parametric string matcher (PSM) of AgreementMaker had several of its correct

Table 1. Experimental Results on the OAEI Anatomy Track

OAEI 2011	Produced	Correct	Precision	Recall	F-measure
<i>MM, Uberon</i>	1439	1348	93.7	88.9	91.2
<i>MMSS, 0.90, Uberon</i>	1441	1348	93.5	88.9	91.1
<i>MMSS, 0.95 PSM kept, Uberon</i>	1443	1353	93.8	89.2	91.4
<i>MMSS, 0.95 PSM kept, Uberon+FMA</i>	1453	1364	93.9	90.0	91.9

mappings overridden by the MMSS matcher. A new LWC of matchers kept PSM mapping and the similarity threshold was raised [3]. Row 3 shows these results.

AgreementMaker did not participate in 2012 OAEI. GOMMA-bk (Generic Ontology Matching and Mapping Management with background knowledge) [4] uses a composition-based approach by mapping to three reference ontologies (UMLS, Uberon, and FMA). It was the best performer for the anatomy track with respect to f-measure with a value of 0.923. Its precision and recall were 0.917 and 0.928.

The next experiment combined Uberon and the FMA as reference ontologies to see if MMSS performance could improve. The results in the last row of Table 1 show the f-measure of 91.9, almost as good as GOMMA's which required the complete UMLS. The precision of 93.9 was better than GOMMA's but the recall was not as good. All previous experiments with the MMSS used the Lin semantic similarity measure. The two major measure categories are path-based and information-content (IC) based, which have been more prevalent in bioinformatics research. A detailed overview of current semantic similarity measures used in bioinformatics research along with their formulas and references can be found in [5]. One path based measure, the Wu and Palmer (WP) and three IC-based semantic similarity measures, the Resnik, the Lin, and the Jiang-Conrath (JC) are compared. IC measures how specific a concept is in a given ontology. The more specific the higher its IC is. Corpus-based or an ontology-based IC measures can be used. Here the ontology-based method in [6] is used. To compare the performance of these measures without the effects of the other matchers, only the MMSS is used. Table 2 shows the results with both Uberon and FMA.

Table 2. Semantic Similarity Measures within reference ontologies

MMSS only, 0.95	Produced	Correct	Precision	Recall	F-measure
WP	1300	1227	94.4	80.9	87.1
Resnik	1283	1226	95.6	80.9	87.6
Lin	1288	1227	95.3	80.9	87.5
JC	1295	1228	94.8	81.0	87.4

The WP measure produced the most mappings with the worst f-measure although all produced nearly identical number of correct mappings. The JC measure had highest recall yet lowest precision. The Resnik measure had the highest f-measure because of it producing less incorrect mappings. More work needs be done to determine if differences exist in the correct mappings.

References

1. Cruz, I.F., Stroe, C., Caimi, F., Fabiani, A., Pesquita, C., Couto, F.M., Palmonari, M.: Using AgreementMaker to Align Ontologies for OAEI 2011. In: International Semantic Web Conference on Ontology Matching Workshop (2011)
2. Cross, V., Silwal, P., Morell, D.: Using a Reference Ontology with Semantic Similarity in Ontology Alignment. In: International Conference on Biomedical Ontologies (ICBO), Graz, Austria, July 22-25 (2012)

3. Cross, V., Silwal, P.: Measuring Semantic Similarity within Reference Ontologies to Improve Ontology Alignment. In: *Ontology Matching Workshop, 11th Int. Semantic Web Conference (ISWC 2012)*, Boston, USA (2012)
4. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA Results for OAEI 2012. In: *Seventh International Workshop on Ontology Matching, ISWC 2012* (2012)
5. Cross, V.: Ontological Similarity. In: *Data Mining in Biomedicine Using Ontologies*, pp. 23–43. Artech House, Norwood (2009) ISBN-13:978-1-59693-370-5
6. Seco, N., Veale, T., Hayes, J.: An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In: *ECAI*, pp. 1089–1090 (2004)

Market-Based SPARQL Brokerage: Towards Economic Incentives for Linked Data Growth

Mengia Zollinger, Cosmin Basca, and Abraham Bernstein

DDIS, Department of Informatics, University of Zurich, Switzerland
{lastname}@ifi.uzh.ch

Abstract. The growth of the Web of Data (WoD) has primarily been funded by subsidies. New datasets are financed via public funding or research programs. This may eventually limit the growth and could hamper data quality for lack of clear incentives. We propose **MaTriX**, a market-based SPARQL broker over the WoD as an economically viable growth option. Similar to others, queries are associated with a budget and minimal result-set quality. The broker then employs auction mechanisms to find a set of data providers that jointly deliver the results. Preliminary results shows that mixing free and commercial providers exhibits superior: consumer surplus, producer profit, total welfare, and recall.¹

1 Introduction

A fast growing decentralized repository of knowledge, the WoD consists of many data tenants that publish and manage interlinked RDF datasets. In contrast to the traditional Web the vast majority of datasets are freely available forming the Linked Open Data (LOD) cloud. Most exist by means of subsidies from research projects or governments. Unlike the Web, where the provision of advertising drives much of the monetization, traditional incentive mechanisms to publish data do not work in the WoD, as datasets are algorithmically queried. Consequently, results of WoD queries often do not contain attributions to the original source removing most non-monetary benefit for the publisher. Centralizing all data in one big database akin to Google's index although technically possible would not work as Van Alstyne *et al.* [4] argue since incentive misalignments would lead to enormous data quality problems and inefficiencies, removing this approach from consideration. In this poster we propose *the use of a global price-market for data* to address the questions of economic viability of the WoD. Such a market could close the gap between applications needing access to diverse data and providers whilst providing a well-defined set of incentive mechanisms to all parties. End-users would query the marketplace that would negotiate with providers for answers. Early research on microeconomic-based scheduling focused on the efficiency of computational resources allocation [2].

¹ The poster explains both the procedure in detail and provide the empirical results.

Stemming from the interaction between data management and microeconomics, the Mariposa [3] RDBMS drops the traditional cost-based optimizer in favor of a market-based one. This poster, in contrast, explores economic methods as incentive mechanisms for publishing.

2 System Design

An extension of AVALANCHE [1] MaTriX requires that providers register prior to query execution by supplying simple information like the SPARQL endpoint URL and simple statistics like vocabularies used. Then the quality of a provider² is assessed. An actual query A with budget B and quality constraint Q is executed in 3 phases as mandated by AVALANCHE. During the *planing* Avalanche generates the ordered list of plans U_A for query A . Then, during *bidding and plan selection* MaTriX starts a reverse auction for each of the top k plans $P_i \in U_A$ by issuing requests for bids to the providers who have information pertinent to any query fragment $F_{P_i} \in P_i$. From all successful auctions MaTriX picks the plan P_i that minimizes the monetary cost and is still under budget whilst providing sufficient quality (i.e., $i : \min_i[\sum(cost(F_{P_i}))] < B \wedge \min(quality(F_{P_i})) \geq Q$). Finally, MaTriX passes the winning plan P_i to AVALANCHE for execution and pays out the fees to the providers.

3 Findings

In a preliminary evaluation we compare settings where data is offered commercially only to settings where there are both free and commercial data providers. Note that we omit the setting with only free providers as it is only viable given subsidies. To generalize our findings we vary the quality of various providers between high, medium, and low as well as vary the kind of auction used between first-price and second price sealed bid auction.

We find that a reverse, sealed-bid second price auction mechanism provides consumers with a higher consumer surplus, in a mixed commercial and free provider setting. Whilst this may not be surprising, we also show that producers will be able to reap higher profits in the mixed setting. We even find, that for profit producers might be enticed by these higher profits to cross-subsidize the free information providers – a surprising result. Furthermore, another positive aspect is denoted by the general incentive for providers to expose high(er) quality data which in turn drives profits up. Whilst our findings are clearly preliminary and burdened by a number of limitations our papers presents the first systematic study trying to provide a sound economic foundation for the WoD. Indeed, the study lays out the foundation and agenda for such economic studies of data on the web. As such it paves the way to a financially healthy WoD.

² A procedure beyond the scope of this poster.

References

1. Basca, C., Bernstein, A.: Avalanche: Putting the Spirit of the Web back into Semantic Web Querying. In: The 6th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2010) (November 2010)
2. Malone, T.W., Fikes, R.E., Howard, M.T.: Enterprise: a market-like task scheduler for distributed computing environments (November 1983)
3. Stonebraker, M., Aoki, P.M., Litwin, W., Pfeffer, A., Sah, A., Sidell, J., Staelin, C., Yu, A.: Mariposa: a wide-area distributed database system. *The VLDB Journal The International Journal on Very Large Data Bases* 5(1), 48–63 (1996)
4. Van Alstyne, M., Brynjolfsson, E., Madnick, S.: Why not one big database? principles for data ownership. *Decis. Support Syst.* 15(4) (December 1995)

A Shared Vocabulary for Audio Features

Alo Allik, György Fazekas, Simon Dixon, and Mark Sandler

Queen Mary University of London

{alo.allik,gyorgy.fazekas,simon.dixon,mark.sandler}@eecs.qmul.ac.uk

Abstract. The aim of the Shared Open Vocabulary for Audio Research and Retrieval project is to foster greater agreement on the representation of content-based audio features within music research communities. The Audio Feature Ontology has been developed for this purpose as part of a library of modular ontologies in order to increase interoperability, reproducibility and sustainability in music information retrieval workflows. The ontology provides a descriptive framework for expressing different conceptualisations of the audio features domain and allows for publishing content-derived information about audio recordings.

Introduction. Due to profusion of digital audio content there is a growing demand for vocabularies that facilitate interoperability between music related data sources. This includes content-based audio feature data, which can be utilised for various commercial and research purposes such as automatic genre classification, query by humming services and many others. Researchers in audio and music information retrieval increasingly use common sets of features to characterise audio material, and large data sets are released for commercial and scientific use. The development of data sets and research tools however is not governed by shared open vocabularies and common data structures. Current formats for describing content-based audio features, including standard as well as generic structured data formats are limited in their extensibility, modularity and interoperability. This limits their ability to support reproducible research, sustainable research tools, and the creation of shared data sets.

The Audio Feature Ontology provides a model for describing acoustical and musicological data and allows for publishing content-derived information about audio recordings. It was initially created within the Music Ontology framework[1]. The Music Ontology has been adopted by researchers and user communities, as well as the industry, including the BBC and its music website (<http://bbc.co.uk/music>). The audio feature ontology subsumes concepts from other ontologies in this framework, including the Event and Timeline ontologies. With regards to different conceptualisations of feature representations, the Audio Feature ontology deals with data density and temporal characteristics[1]. However, its vocabulary is incomplete with regards to user needs within the audio research communities as a number of popular features extracted in research and commercial software are not supported.

Updated Audio Feature Ontology. The aim of this work is to foster greater agreement on the representation of audio features within research communities and to extend the Audio Feature Ontology. It may not be feasible however that

a single ontology may completely represent the different conceptualisations of the domain that exist in the research communities. For example, in some contexts, audio features are categorised according to musicological concepts, such as pitch, rhythm and timbre, while in others the computational workflow used in calculating the features determines the taxonomic hierarchy. The main scope of the present ontology is to provide a framework for communication, feature representation, and describe the association of features and audio signals.

Approach. In order to gain a better understanding of the domain and user needs, a catalogue of audio features was first compiled based on a thorough review of relevant literature, existing feature extraction tools and vocabularies, and research workflows. The catalogue was created in linked data format listing feature objects and their attributes and serves as the foundation for a hybrid ontology engineering process, combining manual and semi-automatic approaches[2]. This catalogue facilitates conceptual scaling of feature attributes to generate concept lattices for Formal Concept Analysis (FCA)[3]. Concept lattices facilitate the extraction of conceptual hierarchies and thereby can provide a foundation for shared vocabularies. Subsets of popular features computed in feature extraction tools can be queried from the linked data graph to help define the scope and domain boundaries of the ontology. The catalogue identifies approximately 400 different features and significantly increases the scope of the ontology.

Conclusion. A shared vocabulary for audio features has the potential to increase research interoperability, reproducibility and sustainability in music information retrieval research communities. Leveraging open linked data formats, it provides a flat vocabulary to facilitate task and tool specific ontology development and serves as a descriptive framework for audio feature extraction. Future work includes developing specific ontologies based on the existing tools and vocabularies, and updating Sonic Annotator, a command line application for automatically analysing large audio collections. SAWA, an open access demonstrator will also be updated to adhere to the revised ontology. Finally, Sonic Visualiser, a desktop application for visualising audio features will be updated to handle RDF files adhering to the updated ontology.

Acknowledgements. This research was funded by JISC (<http://jisc.ac.uk>) through the Shared Open Vocabulary for Audio Research and Retrieval project.

References

1. Fazekas, G., Raimond, Y., Jakobson, K., Sandler, M.: An overview of semantic web activities in the OMRAS2 project. *Journal of New Music Research (JNMR)* 39(4) (2010)
2. Cimiano, P., Hotho, A., Stumme, G., Tane, J.: Conceptual knowledge processing with formal concept analysis and ontologies. In: Eklund, P. (ed.) *ICFCA 2004*. LNCS (LNAI), vol. 2961, pp. 189–207. Springer, Heidelberg (2004)
3. Wille, R.: Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications* 23, 493–515 (1992)

Exploratory Search on the Top of DBpedia Chapters with the Discovery Hub Application

Nicolas Marie^{1,2}, Fabien Gandon¹, and Myriam Ribière²

¹ INRIA Sophia-Antipolis, Wimmics Team,
2004 Route des Lucioles, Sophia-Antipolis, 06410 BIOT
{nicolas.marie, fabien.gandon}@inria.fr

² Alcatel-Lucent Bell Labs,
7 Route de Villejust, 91260 NOZAY
{nicolas.marie, myriam.ribiere}@alcatel-lucent.com

Abstract. Discovery Hub is a novel application that processes DBpedia for exploratory search purpose. It implements on-the-fly semantic spreading activation and sampling over linked data sources to suggest ranked topics of interest to the user.

Motivation. Exploratory search [2] systems help users learn or investigate a topic. They provide a high level of assistance during the search task and advanced explanations about the results. In the past few years several works put in evidence the interest of using linked data, and especially DBpedia¹, for exploratory search systems. In these systems the user query is matched with the DBpedia graph. Then relevant related results are selected and ranked. These results are then presented through interfaces optimized for exploratory tasks. These systems use various methods but all depend on a partial or total results pre-processing. As it is a young research area there are many possible improvements:

- No work deals with the data freshness issue. Indeed, linked data datasets are evolving over the time. The continuous update of the data and its impact on the preprocessing need to be addressed.
- No work proposes a lightweight method that is applicable on remote SPARQL endpoints. Indeed the pre-processing phases used in the state-of-the-art works are specific to the knowledge base addressed and sometimes require a copy of the base to be performed. Consequently existing systems often only consider one data source e.g. the English-speaking version of DBpedia.

These limitations are mainly due to the preprocessing step that is strongly conditioning the type and the range of results the applications are able to retrieve. To overtake these limits, we propose a method that selects and ranks on-the-fly a meaningful subset of resources in a targeted LOD dataset. We use the term “*on-the-fly*” to stress that the method does not need any pre-processing.

¹ <http://dbpedia.org>

Approach. To reach our objective of an on-the-fly linked data processing we propose a novel method [3] based on a semantic spreading activation coupled with a sampling process. The spreading activation technique [1] consists in associating a value to the node(s) representing the user’s interest and then spreading this value to the neighborhood iteratively with various heuristics. It was applied over RDF for various objectives (see [4]). Our approach differs as the propagation controlling pattern is a type-based semantic weight that is function of the stimulated origin node. In other words the origin node semantics plays a significant role in the distribution of activation even in “*distant*” parts of the graph:

- When a query is entered a local triple store instance is created. It imports the neighbourhood of the seed node(s) filtered with a semantic pattern taking in consideration the nodes’ types. This pattern aims to concentrate the activation on a consistent subset of nodes in order to increase the relevance. The most prevalent types of the seed’s neighbours are included in the pattern. For each neighbour only its deepest type(s) in the class hierarchy is/are taken in account.
- As the propagation spreads along the iterations the neighbourhoods of the most activated nodes are imported till a limit (a maximum number of triples) is reached. The semantic pattern is re-used for the imports at every iteration.
- The propagation stops when the maximum number of iterations is reached. The most activated nodes are suggested to the user in decreasing order of activation.

We performed extensive analysis on a large set of queries to understand the behaviour of the algorithm. It helped us to set its main parameters correctly in order to get a fast response time without degrading the results too much. We obtained an average response time of 2031ms with a standard deviation of 1952 ms on a set of 100.000 queries having one node as input i.e. ego-centric queries. The 100.000 nodes were selected randomly thanks to a random walker. With the proposed method it is possible to address remote data sources using their endpoints, e.g. local DBpedia chapters.

Prototype. Discovery Hub² is an exploratory search engine that helps its users to explore topics of interest. It uses DBpedia to perform the semantic spreading activation, render (e.g. label, description, pictures) and organize the results space (e.g. filters, facets). It also offers various explanations about the results based on DBpedia and Wikipedia. Discovery Hub is able to address localized DBpedia chapters: using the Italian data for the query “*Leonardo Da Vinci*” for instance.

References

- [1] Crestani, F.: Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review* 11(6), 453–482 (1997)
- [2] Marchionini, G.: Exploratory search: From finding to understanding. *ACM* (2006)
- [3] Marie, N., Corby, O., Gandon, F., Ribière, M.: Composite interests’ exploration thanks to on-the-fly linked data spreading activation. In: *Hypertext 2013* (to appear, 2013)
- [4] Rodríguez, J.M.Á., Gayo, J.E.L., Ordoñez de Pablos, P.: An Extensible Framework to Sort out Nodes in Graph-Based Structures Powered by SA Technique: The ONTOSPREAD Approach. In: *International Journal of Knowledge Society Research* (2012)

²<http://semreco.inria.fr>

The Finnish Law as a Linked Data Service

Matias Frosterus, Jouni Tuominen, Mika Wahlroos, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University and University of Helsinki
{firstname.lastname}@aalto.fi
<http://www.seco.tkk.fi/>

Abstract. Juridical information is important to organizations and individuals alike and is linked to from all walks of life. The Finnish government has published the Finlex Data Bank¹ for searching and browsing legislation documents. However, the data there is not yet open, is based on a traditional XML schema, and does not conform to new semantic metadata standards. There are many difficulties in maintaining and using the site in, e.g., data harvesting, interoperability, querying, and linking that could be mitigated by the Semantic Web technologies. This paper presents an approach and a project—including first results—for publishing and using the Finnish legislation as a 5-star Linked Open Data service.

Publishing and using juridical information is challenging in many ways. It is produced by different parties, such as governmental bureaus, ministries, different levels of courts, research organizations, and media. The content is heterogeneous and produced using differing tools, data formats, and practices. The links between documents are often informal and/or not made explicit. The law in general is a dynamic, changing entity: for example, it is important to be able to refer to different versions of a law at different points of time. These challenges can be addressed through the use of linked data techniques [1]. Further consideration should be paid to offering the linked juridical data in the form of ready-to-use services and end-user applications.

Our goal² is to create a 5-star linked open data service of the Finnish legislation, a semantic version of Finlex. Based on the service, a next generation of public and commercial legal systems can be built. The work involves 1) juridical ontology development, 2) metadata modeling of legislative documents, 3) the publication of the content as linked data, and 4) the evaluation of the system via application demonstrators.

Ontologies. We have begun the work by harmonizing different in-use thesauri and building an ontology of legal concepts. A project by the Finnish Ministry of Justice combined ca. 30 vocabularies and thesauri used by different courts in Finland, and we further added two vocabularies used by commercial legal information services. The collected ca. 9,000 different terms were then mapped with the larger KOKO ontology cloud (ca. 45,000 concepts) that is used in Finland. In addition, the anticipated ontology infrastructure includes two classification schemes of law.

¹ <http://www.finlex.fi/en/>

² The work is a collaboration with the Ministry of Justice, the Ministry of Communications, Edita Publishing Ltd, and Talentum Corp. under the national Linked Data Finland project (<http://www.seco.tkk.fi/projects/ldf/>) funded mainly by Tekes.

Metadata Model. A first version of a metadata model for representing the structure and linking in the Finnish law has been designed in RDF, based on the Finlex XML schema.

Automatic Annotation. We aim at annotating the subject matter of legislation, case documents, and legal news using the presented ontologies and metadata models. In this way, the documents could be interlinked also in terms of their content (e.g., materials related to family crimes), not only in terms of the legislative structures now used in Finlex (e.g., that a law X is based on or replaces another law Y). The first dataset, a collection of 35,000 legal news items, is now being annotated.

Publication. The data will be published as linked data, including datasets for the laws, court cases, and preparatory works, which are often critical when interpreting the laws. Aside from a standard SPARQL endpoint publication, the laws will be published also in the ONKI ontology service, providing not only a browsing facility but a selector widget with autocompletion for annotation, and APIs for machine use [3]. Each law and its version has a URI that can be found and fetched as an unambiguous reference from ONKI.

Evaluation via Demonstrators. New semantic search and recommendation facilities for Finlex will be demonstrated using our SAHA-HAKO tool³ [2]. This tool provides means for data editing and analysis of data quality against the metadata models. The system provides a SPARQL endpoint to its data store, a download facility, and incorporates a faceted search engine on top of SPARQL. In this case, useful facets include the laws and their sections, time, subject matter, as well as the law classification systems in use. The recommendation system would allow, e.g., studying a given case with links to similar cases that reference to the same laws. This can be further refined by noting the cases where the law has changed, so that these can be viewed separately. Finally, users can be offered a possibility of subscribing to the laws and specific sections of laws of interest to them, so that they would receive a notification, if a law in their interest has changed. This is especially pertinent in situations where, e.g., an organization's internal regulation rests upon a specific law whose changes can be critical to the validity of the regulation.

References

1. Hoekstra, R.: The MetaLex Document Server legal documents as versioned linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 128–143. Springer, Heidelberg (2011)
2. Kurki, J., Hyvönen, E.: Collaborative metadata editor integrated with ontology services and faceted portals. In: Workshop on Ontology Repositories and Editors for the Semantic Web (ORES 2010) at ESWC 2010. CEUR Workshop Proceedings, vol. 596 (2010)
3. Viljanen, K., Tuominen, J., Hyvönen, E.: Ontology libraries for production use: The Finnish ontology library service ONKI. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 781–795. Springer, Heidelberg (2009)

³ <http://code.google.com/p/saha/>

A Distributed Entity Directory^{*}

Fausto Giunchiglia and Alethia Hume

Department of Information Engineering and Computer Science

University of Trento, Italy

{fausto,hume}@disi.unitn.it

<http://www.disi.unitn.it>

Abstract. We see the local content from peers organized in directories (i.e., on local ordered lists) containing local representations of entities from the real world (e.g., persons, locations, events). Different local representations can give different “versions” of the same real world entity and use different names to refer to it (e.g., Fausto Giunchiglia, Giunchiglia F., Prof. Giunchiglia). Although the names used in these directories connect data that could complement each other, there are no links that allow peers to share and search across them. We propose a Distributed Directory of Entities that makes explicit these connecting links and allows peers to: (i) maintain their data locally and (ii) find the different versions of a real world entity based on any name used in the network. The model we present exploits the name as the central (multi-value) attribute of entities and aims to convince readers of the importance of such names in a peer-to-peer scenario.

1 A Distributed Entity Directory

We see the internet as a network of peers maintaining local directories of entities that are interconnected. An example of this is shown in the first part of Figure 1, where different local representations are seen as pieces of information about a particular entity that are stored in a distributed manner in the network. Entities can be of different types, they have a name, and are described by attributes (e.g., latitude-longitude, size, birth date). However, the names of entities play a key role as identifiers, which are used to distinguish them from others (e.g., F. Giunchiglia, Italy, University of Trento) and behave similarly to keywords. Moreover, real world entities can be called by multiple names as a consequence of different types of variations and errors (e.g., format variations, partial and full translations, misspellings and pseudonyms).

We formalize these characteristics in a Distributed Directory¹ that distinguishes between a Digital Entity (*DE*) and a Real World Entity (*WE*):

$$DE = \langle URL, \{N\} \rangle \quad (1) \quad WE = \langle URI, \{URL\} \rangle \quad (2)$$

A *DE* is a local representation of an entity that exists in the real world, while a *WE* represents the real world entity (modeled as a class of *DEs*). A *URL* is

^{*} This work has been partially supported by the EU-FET grant SmartSociety 600854.

¹ The interested reader can find an extended version of the approach in [1].

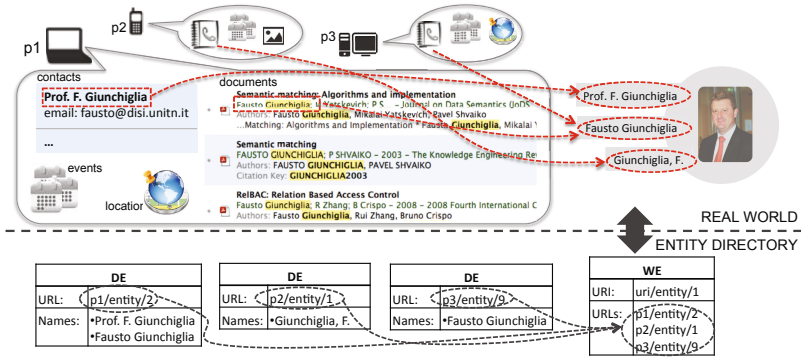


Fig. 1. Example of a network of interconnected directories

used to uniquely identify a *DE* and it can be dereferenced to obtain the local description based on attributes; $\{N\}$ is the set of names used in *DEs* to refer to a *WE*. A *URI* is used to uniquely identify a *WE* and a non-empty set $\{URL\}$ contains the identifiers of different *DEs* that describe *WE* (see Figure 1).

The notions introduced in (1) and (2) define the *many-to-many* relation between *Names* and *WEs*. Different names can be used in *DEs* to identify the same *WE*. At the same time, the names used in *DEs* that refer to different *WEs* can overlap. We exploit these notions by building two indexes:

1. A *DEindex* maps *WEs* (i.e., *URIs*) to *DEs* (i.e., *URLs*) containing the different versions that locally represent them.
2. A *WEindex* maps the *names* (given in *DEs*) to *WEs* (i.e., *URIs*) that are candidates to be called by such names.

The two indexes allow finding all the different versions of an entity based on any name used in the network to refer to it. The fact that the indexes contain only identifiers (i.e., names, *URIs* and *URLs*) allows peers to always maintain their attribute-based descriptions locally. Furthermore, in our approach the indexes are stored using a Distributed Hash Table (DHT) built on top of the same network of peers that share and consume the data, which gives us scalability.

We evaluated the approach on networks of 50, 100, and 150 peers running on PlanetLab. The average query time (as a measure of the performance) is 2.7 seconds for the different network sizes. We also noticed that more than 55% of the queries are answered in less than a second, almost 70% in less than 2 seconds and only 9% of queries takes more than 5 seconds to be answered. We believe these results are promising in terms of scalability because they show that performance can be stable with the network growth.

Reference

1. Giunchiglia, F., Hume, A.: A distributed directory system. Technical Report DISI-13-005, University of Trento, Italy (2013)

Optique: OBDA Solution for Big Data[★]

D. Calvanese³, Martin Giese¹⁰, Peter Haase², Ian Horrocks⁵, T. Hubauer⁷,
Y. Ioannidis⁹, Ernesto Jiménez-Ruiz⁵, E. Kharlamov⁵, H. Kllapi⁹, J. Klüwer¹,
Manolis Koubarakis⁹, S. Lamparter⁷, R. Möller⁴, C. Neuenstadt⁴, T. Nordtveit⁸,
Ö. Özcep⁴, M. Rodriguez-Muro³, M. Roshchin⁷, F. Savo⁶, Michael Schmidt²,
Ahmet Soylu¹⁰, Arild Waaler¹⁰, and Dmitriy Zheleznyakov⁵

¹ Det Norske Veritas, ² Fluid Operations AG, ³ Free University of Bozen-Bolzano,
⁴ Hamburg University of Technology, ⁵ Oxford University, ⁶ Sapienza University of Rome,
⁷ Siemens Corporate Technology, ⁸ Statoil ASA, ⁹ University of Athens, ¹⁰ University of Oslo

1 Motivations and Challenges

Accessing the *relevant* data in Big Data scenarios is increasingly difficult both for end-user and IT-experts, due to the *volume*, *variety*, and *velocity* dimensions of Big Data. This brings a high cost overhead in data access for large enterprises. For instance, in the oil and gas industry, IT-experts spend 30–70% of their time gathering and assessing the quality of data [1]. The Optique project (<http://www.optique-project.eu/>) advocates a next generation of the well known *Ontology-Based Data Access* (OBDA) approach to address the Big Data dimensions and in particular the data access problem. The project aims at solutions that reduce the cost of data access dramatically.

OBDA systems address the data access problem by presenting a general ontology-based and end-user oriented query interface over heterogeneous data sources. The core elements in a classical OBDA systems are an *ontology* describing the application domain and a set of *mappings*, relating the ontological terms with the schemata of the underlying data source. OBDA is natural for addressing the 3V of Big Data: the ontology covers the *variety* of data sources, on the fly data access for query evaluation allows to obtain fresh data regardless the *velocity* of its changes, and the virtual nature of data integration allows to manage large *volumes* of data.

The important limitations of the state of the art OBDA systems are as follows:

- The *usability* of OBDA systems is hampered by the need to use a formal query language which is difficult for end-users even if they know the ontological vocabulary.
- The *prerequisites* of OBDA, i.e., ontology and mappings, are in practice expensive to obtain. Additionally, they are not static artefacts and should evolve according to the new end-users' information requirements. In current OBDA systems bootstrapping of ontologies and mappings are in a premature stage at the best.
- The *scope* of existing systems is too narrow. The chosen expressiveness of the ontology and mapping language are focused on very concrete solutions. Management of *streaming data* is essentially ignored despite their importance for industry.
- The *efficiency* of the translation process and the execution of the queries is too low.

[★] This research was financed by the Optique project with the grant agreement FP7-318338.

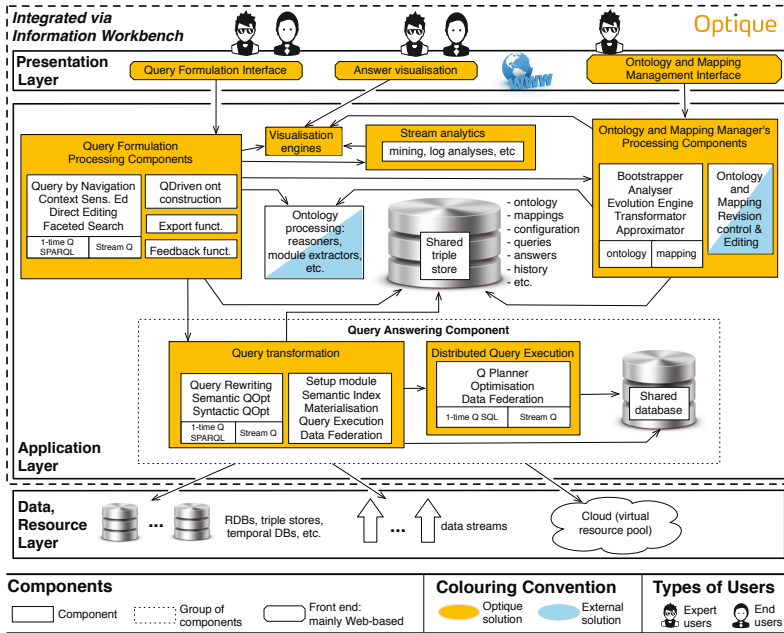


Fig. 1. Optique's general architecture

Some of these items were addressed by the Semantic Web community, but the solutions are limited and there is no unified approach to deal with all these aspects in one system.

2 Optique Solution

Figure 1 shows an overview of the Optique architecture and its components which aim at overcoming the limitations above. We now briefly introduce the main components:

- *The query formulation component* aims at providing a friendly interface for non-technical users combining multiple representation paradigms for ontologies (query by navigation, faceted search, context sensitive editing, etc.). *Query-driven ontology extension* subcomponent will allow to extend the ontology on the fly.
- *The ontology and mapping management component* will provide tools to (i) semi-automatically bootstrap an initial ontology and mappings and (ii) maintain the consistency between the evolving mappings and the evolving ontology.
- *The query answering component* is decomposed into several layers in order to achieve efficiency: transformation, planning and execution. The transformation layer will exploit the mappings and the ontology and will apply optimised query rewriting techniques. The planning and execution layers will distribute queries to individual servers and use massively parallelised (cloud) computing.

To sum up, the Optique system will provide a novel end-to-end OBDA solution for Big Data access which will be integrated in the Information Workbench platform

(www.fluidops.com/information-workbench/) and address a number of important industry requirements. The technology and system will be developed in a close cooperation of six universities, two industrial partners, and use cases: Statoil and Siemens. The system will be deployed and evaluated in our use cases. It will provide valuable insights for the application of semantic technologies to Big Data integration in industry.

Reference

1. Crompton, J.: Keynote talk at the W3C Workshop on Sem. Web in Oil & Gas Industry (2008), <http://www.w3.org/2008/12/ogws-slides/Crompton.pdf>

Linked Open Ontology Cloud KOKO—Managing a System of Cross-Domain Lightweight Ontologies

Matias Frosterus*, Jouni Tuominen, Sini Pessala*, Katri Seppälä, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University and University of Helsinki
{firstname.lastname}@aalto.fi
<http://www.seco.tkk.fi/>

1 Introduction

The Linked Data movement has focused on building cross-domain interoperability by creating and using (typically) `owl:sameAs` mappings between the datasets in the Linked Data Cloud (LOD). However, when linking data, ontologies would allow for deeper interoperability. Because different ontologies have been used when annotating different datasets, we argue that the LOD cloud needs to be complemented by developing a lightweight “Linked Open Ontology Cloud” (LOO). Aligning the ontologies requires more refined techniques than mapping data instances for the LOD.

In a LOO, special care must be taken to ensure that the (subclass) reasoning remains valid when performed over mappings between ontology boundaries. Furthermore, when an ontology is changed, reasoning over ontologies mapped to it may become invalid and therefore coordination and collaboration is needed when developing the ontologies.

We have created a LOO cloud called KOKO made up of a single general upper ontology and 15 domain ontologies (e.g., health, cultural heritage, agriculture, government, defense; with more being integrated into the system) as part of the national FinnONTO project (2003–2012). The system is based on transforming a set of 15 legacy thesauri in use into lightweight SKOS ontologies and interlinking them with each other through the general upper ontology. The result is KOKO, a harmonized global ontology cloud of some 45,000 concepts aligned into a single hierarchy. From the end-user’s viewpoint, KOKO cloud is seen and used like a single ontology without boundaries. For the developers, the cloud is seen as a collection of interlinked ontologies in order to easily divide responsibilities based on domain expertise. KOKO has been published and is in use as a service in the national ontology library system ONKI¹ [2].

2 A Model for Managing a Linked Open Ontology Cloud

Based on our work with KOKO, we created and propose the following model depicted in Fig. 1 for developing a LOO cloud in a coordinated and collaborative manner.² The model with related tools is in field testing at the moment.

* Also affiliated with The National Library of Finland, <http://www.nationallibrary.fi/>

¹ online browsing: <http://onki.fi/en/browser/overview/koko-beta>
download: <http://tinyurl.com/koko-download>

² This work is part of the FinnONTO and Linked Data Finland projects, funded mainly by Tekes.

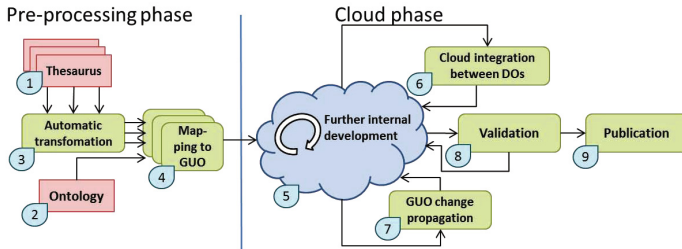


Fig. 1. The model of Linked Open Ontology Cloud formation and management

In the *Pre-processing Phase*, thesauri (1) and ontologies (2) are selected for building blocks of the ontology cloud. A thesaurus is converted into RDF format using a shared ontology schema (3) and aligned with a general upper ontology (GUO) (4). Aligning domain ontologies with a GUO forms a basis for interoperability by providing a complete hierarchy and is much easier to maintain than direct mappings between domain ontologies [1]. A similar approach was used by the IEEE Standard Upper Merged Ontology SUMO³ working group. In our case, a natural basis for the GUO was the General Finnish Thesaurus YSA that was transformed into a General Finnish Ontology YSO⁴.

The *Cloud Phase* begins after the domain ontologies have been aligned with the GUO (5). Since the alignments were made between domain ontologies and the GUO only, there may be mutually overlapping parts between the domain ontologies. To facilitate the integration of domain ontologies (DO) (6), processes and tools for discovering overlapping parts of the ontologies are needed. Based on the analysis, it is possible to eliminate redundant development work by deciding between domain ontology developers which ontologies maintain the overlapping parts.

Changes in the GUO create pressure for the domain ontologies to be updated accordingly to ensure the consistency of the cloud. For example, in our system, some 2,000 changes are made annually to the upper ontology YSO. These should be taken into account in the development process of the domain ontologies (7). Fortunately, only changes to concepts linked to the domain ontology via equivalency or subclass-of relations are likely to be relevant to the domain ontology. After editing the ontology, its logical consistency and other quality aspects should be validated (8) and fixed as necessary. Finally, the ontology cloud can be published as services for humans and machines, e.g., as user interfaces, APIs, and downloadable files (9).

References

1. Hameed, A., Preece, A., Sleeman, D.: Ontology reconciliation. In: Staab, S., Studer, R. (eds.) *Handbook on Ontologies*, pp. 231–250. Springer, Germany (2004)
2. Tuominen, J., Frosterus, M., Viljanen, K., Hyvönen, E.: ONKI SKOS server for publishing and utilizing SKOS vocabularies and ontologies as services. In: Aroyo, L., et al. (eds.) *ESWC 2009*. LNCS, vol. 5554, pp. 768–780. Springer, Heidelberg (2009)

³ <http://suo.ieee.org/SUO/SUMO/index.html>

⁴ <http://www.seco.tkk.fi/ontologies/yso/>

A Semantic-Enabled Engine for Mobile Social Networks*

Ronald Chenu-Abente, Ilya Zaihrayeu, and Fausto Giunchiglia

Dept. of Information Engineering and Computer Science, University of Trento, Italy
{chenu,ilya,fausto}@disi.unitn.it

Abstract. Despite their success in general applications, social networks also present a series of challenges like attracting user signups, keeping a healthy contribution level, user privacy concerns, etc. This paper introduces the Social Core – a social network engine that adds semantic-based functionalities like semantic annotations, semantic search and semantic-enhanced access control; as a way to enhance and answer to the current challenges of social applications. The Social Core was integrated as part of the SmartCampus mobile platform, which is currently being live tested by around one hundred students.

Social Core Overview

Online social networks are Internet-based services that people use for managing their social relations, sharing their content and having access to the content shared by others. Beyond these most common features each social network also offers its own specific features like blogging capabilities or document creation/editing. However, despite the usefulness of their services and their general adoption, several studies have discussed concerns and issues related to them. Some common issues include the lack of clear and easy-to-understand privacy settings, and a wide-spread perception that social networks are more a distraction than useful services to be taken seriously.

To try to address the common limitations of state of the art social networks, the Social Core uses a semantic storage (introduced enabling concept search [1]) in place of a regular database. Furthermore, the Social Core defines users as an extension that allows entities to become a subject of social interactions. These subjects along with the already existing entities used as objects become the two main components for the definition of the semantic-enabled social network backend (as shown Figure 1):

- *Semantic Storage:* the Social Core defines a set of commonly used entity types such as people, places, events and media. Each entity type provides a structured definition of the metadata used to describe entities of the entity type. Note that different entity bases are defined and used as separate containers belonging to users.
- *Users:* each user is linked by id to an entity (a person entity in this version), which allows it to participate in social interactions representing the entity. Some examples of these social interactions include sharing, tagging, commenting and messaging.

The following are some of the semantic-based services provided:

* This work is partially supported in part by the EU FP7 Project Smart Society n. 600854.

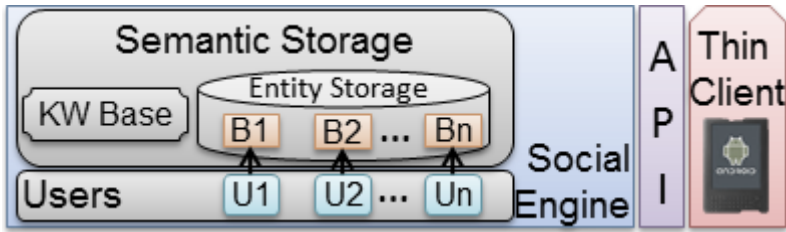


Fig. 1. Social Core as part of the SmartCampus platform

Semantic Services and Platform

- *Access Control*: based on the ReIBAC (Relation Based Access Control) theory [2]. By exploiting the translation from classifications and web directories to lightweight ontologies, we can represent both user networks (subjects) and entity organizations (objects) as lightweight ontologies. This allows computing access control and even suggesting new rules by using Description Logics. Furthermore, multiple subject, object and permission ontologies are made interoperable by Semantic Matching.
- *Semantic Annotations*: Social Core allows its users to annotate entities given that they are granted the necessary permissions to do so. The engine supports three kinds of the annotations: i) text, an arbitrary sequence of characters provided by the user e.g., “party”, “xyz”; ii) semantic, a concept from the Knowledge Base that enables automated reasoning (e.g., searching for ANIMAL would find entities annotated with concepts DOG or CAT); and iii) Entity relations, a pointer to other entity (e.g., a photo entity that is annotated with the people entities that appear in it).
- *Live Topics*: a mechanism that allows its users to find and follow entities or concepts. Essentially the live topic is a query based on three main parameters: i) the source or groups of users that may be the source of the news (i.e., from who?); ii) the topic in form of a free text, a concept or an entity reference (i.e., about what?) and; iii) the type of the entities that will be returned (i.e., how is the desired information represented). Examples include: “Pictures shared by my friends from their mountain hiking” and “Tell me when anyone posts a happy hour in bar Ponte”

The Social Core is integrated through an http APIs to each of the more than one hundred Android clients in the hands of students running the SmartCampus Platform applications. The main objective of the project is to harmonize all the existing services available to the students from the University of Trento and initial data obtained from the first months of usage shows promising results.

References

1. Giunchiglia, F., Kharkevich, U., Zaihrayeu, I.: Concept search. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 429–444. Springer, Heidelberg (2009)
2. Giunchiglia, F., Crispo, B., Zhang, R.: Access control via lightweight ontologies. In: 2011 Fifth IEEE International Conference on Semantic Computing (ICSC), p. 352 (2011)

The Birds of the World Ontology AVIO

Jouni Tuominen, Nina Laurenne, Mikko Koho, and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University and University of Helsinki
`firstname.lastname@aalto.fi`
<http://www.seco.tkk.fi>

Abstract. We present an ontology for managing the scientific and common names of birds. The ontology is based on the TaxMeOn meta-ontology model for biological names. The ontology is in use as an ontology service and it has been applied in a bird watching system.

Biologists refer to organisms by using scientific names although most people use common (vernacular) names. Modeling and managing scientific and common names is not trivial [2,3] because names change over time. For example, a species may be shifted into another genus, which is reflected in the scientific name.

We present “The Birds of the World Ontology AVIO”¹ encompassing all bird taxa of the world, including their scientific names and common names in English, Finnish, and Swedish. The ontology is an application of our earlier work, the meta-ontology model TaxMeOn for biological names and classifications [3]. The aim of the AVIO ontology is to provide a reference to the species names of the birds, e.g., for indexing bird related information.

The AVIO ontology is available as open data and as an operational ontology service in the Finnish Ontology Library Service ONKI [4]. The ontology has been used in an online mobile birding system BirdWatch² [1].

AVIO Ontology. The AVIO Ontology is based on the Finnish names of the birds of the world published by BirdLife Finland³, containing the common names for all the species of the world including their higher taxa—9,740 species, 1,227 genera, and 194 families. The English names and the scientific names are included, too. A CSV formatted species list was provided by BirdLife, and was converted into an RDF-based ontology conforming to the TaxMeOn schema. The classification of taxa was completed with additional higher level taxa.

The CSV formatted species list contained a scientific name and its common name equivalents per line. The hierarchy between the scientific names was expressed using hierarchically numbered identifiers for the names (e.g., 061.0449. *Sterna* and 061.0449.1664 *Sterna paradisaea*). In the RDF conversion, every scientific and common name was given a URI and the relations between them

¹ This work is part of the FinnONTO project (<http://www.seco.tkk.fi/projects/finnonto/>) funded mainly by Tekes.

² <http://www.demo.seco.tkk.fi/birdwatch/>

³ <http://www.birdlife.fi/>

were made explicit: 1) the hierarchical relations between the scientific names, and 2) the equivalence relations between the scientific and common names.

The scientific names are modelled as instances of the TaxMeOn classes *TaxonInChecklist* and an appropriate taxonomic rank (e.g., *Species*). The common names are expressed as instances of the class *VernacularName*. Giving separate URIs for scientific and common names supports temporal tracking of the names as the individual names can be assigned statuses with time stamps.

The resulting ontology is published in the ONKI service⁴. AVIO is available also in SKOS⁵ to support SKOS-based applications. Here the preferred and alternative scientific and common names of a taxon are represented simply as literals that cannot have further properties, unlike in TaxMeOn. This version also includes Swedish common bird names obtained from BirdLife.

Ontology Service and a Use Case. ONKI provides ready-made services for accessing the AVIO ontology. End-users can browse and search the ontology, e.g., to get an overview of the classification of birds or to find a scientific name for a taxon that they know only by the common name. The ONKI selector widget can be integrated into legacy CMS systems to provide an autocomplete and URI fetching features. The ontology is published for machines as Linked Data, as a SPARQL endpoint, and there are several APIs to use, e.g., for query expansion.

AVIO ontology is used as a conceptual hub of the BirdWatch system based on linked data. This system helps citizen ornithologists in species identification, based on the ontological knowledge and the statistics of earlier observations. The URIs are used for linking bird data from different services, such as Wikipedia, the multilingual NatureGate⁶, and 1.25 million bird observations in a SPARQL endpoint based on the GBIF data⁷.

References

1. Hyvönen, E., Alonen, M., Koho, M., Tuominen, J.: BirdWatch—supporting citizen scientists for better linked data quality for biodiversity management. In: Workshop on Semantics for Biodiversity (S4BIODIV), ESWC 2013, Montpellier, France. CEUR Workshop Proceedings (2013)
2. Page, R.: Taxonomic names, metadata, and the semantic web. *Biodiversity Informatics* 3, 1–15 (2006)
3. Tuominen, J., Laurene, N., Hyvönen, E.: Biological names and taxonomies on the semantic web – managing the change in scientific conception . In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 255–269. Springer, Heidelberg (2011)
4. Viljanen, K., Tuominen, J., Hyvönen, E.: Ontology libraries for production use: The Finnish ontology library service ONKI. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 781–795. Springer, Heidelberg (2009)

⁴ online browsing: <http://onki.fi/en/browser/overview/linnut>
download: <http://tinyurl.com/linnut-download>

⁵ <http://onki.fi/en/browser/overview/avio>

⁶ <http://www.naturegate.fi>

⁷ <http://www.gbif.org/>

Exploring the Dynamics of Linked Data

Tobias Käfer¹, Ahmed Abdelrahman², Jürgen Umbrich²,
Patrick O’Byrne², and Aidan Hogan²

¹ Institute AIFB, Karlsruhe Institute of Technology, Germany

² Digital Enterprise Research Institute, National University of Ireland, Galway

Abstract. Little is known about the dynamics of Linked Data, primarily because there have been few, if any, suitable collections of data made available for analysis of how Linked Data documents evolve over time. We aim to address this issue. We propose the Dynamic Linked Data Observatory, which provides the community with such a collection, monitoring a fixed set of Linked Data documents at weekly intervals. We have now collected eight months of raw data comprising weekly snapshots of eighty thousand Linked Data documents. Having published results characterising the high-level dynamics of Linked Data, we now wish to disseminate results: we wish to investigate how results from our experiment might benefit the community and what online services and statistics (relating to Linked Data dynamics) would be most useful for us to provide.

Summary

The Web of (Linked) Data is dynamic. Knowledge about Linked Data dynamics is important for a wide range of applications and can help to answer a wide variety of practical questions, including (but far from limited to):

For warehouses: Which remote datasets need to be updated most frequently?

Which are static and do not need to be refreshed? Are the updates mostly additions or mostly deletions? How often do new documents appear?

For “live query” engines: Which documents are static and can be cached?

Which query patterns involve dynamic patterns (e.g., are `foaf:knows` triples more dynamic than `foaf:name`)? For link traversal methods, how often do links change between documents?

For reasoning engines: Do schema data change more or less often than instance data? Which ontologies are the most dynamic? What kinds of semantics change? Are changes “monotonic” with respect to entailments? Which data are static and subject to materialisation? Which are dynamic and subject to query-rewriting?

For publishers: How often do target documents go offline? How often should deadlinks be checked and pruned? How often should certain link-types (e.g., `owl:sameAs`) be revised for updated remote content? How often do the semantics of vocabularies change?

Few works have looked at the nature of Linked Data dynamics or have tried to answer the types of questions posed above; this is probably due to a lack

of suitable collections tracking changes in Linked Data documents over time. Thus, we proposed the Dynamic Linked Data Observatory (DyLDO) [2,1] for monitoring weekly changes in 86,696 Linked Data documents.

We recently published results (in the main track of ESWC) based on initial analyses of the first six months of data that we have collected. These results begin to answer *some* of the questions listed above [1]. For example, therein:

1. We showed that 5% of documents went offline in six months, establishing an initial estimated death-rate for Linked Data documents.
2. We showed that the content of 62.2% of the monitored documents did not change in six months, and that most of the remaining documents either changed very infrequently (23.2%) or very frequently (8.4%).
3. We identified four types of data-sites: STATIC involving few infrequently changing documents, including `linkedmdb.org`; DUAL involving a few frequently updated documents, including `loc.gov`; BULK involving many infrequently updated documents, including `dbpedia.org`; and ACTIVE involving many frequently updated documents, including `dbtropes.org`.
4. Of those documents that changed at least once, we showed that one quarter only ever updated individual values (often an object literal), one quarter only ever added new triples, and that the other half contained a mix of change types. We also showed, e.g., that the schema signature of a document (set of class and property terms used) rarely changes.
5. Links in the monitored documents are quite static over time, with the exception of a few domains such as `sec.gov`, `identi.ca`, `zitgist.com`, `dbtropes.org` and `freebase.com` that regularly contribute a small volume of fresh links.

We would now like to disseminate our collection and our results for the benefit of the community. We wish to collect use-cases for statistics on Linked Data dynamics that we can extract from our collection and provide online. We are currently working on a live site that visualises changes in Linked Data sites and allows interested users to see changes happen on a weekly basis, where they occur, and what types of changes they are (<http://swse.deri.org/dyldo>). We are also in the process of creating a SPARQL service that indexes details of weekly changes represented as RDF. This would enable external systems to find out which Linked Data sites were updated in the previous week(s) and what types of changes they exhibited. We hope that warehouses, live query engines, reasoners and publishers could then use our API to directly answer many of the questions about Linked Data dynamics highlighted at the outset.

References

1. Käfer, T., Abdelrahman, A., Umbrich, J., O’Byrne, P., Hogan, A.: Observing Linked Data Dynamics. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 213–227. Springer, Heidelberg (2013)
2. Käfer, T., Umbrich, J., Hogan, A., Polleres, A.: DyLDO: Towards a Dynamic Linked Data Observatory. In: LDOW at WWW. CEUR-WS, vol. 937 (2012)

MAD SWAN: A Semantic Web Service Composition System

George Markou and Ioannis Refanidis

Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
{gmarkou, yrefanid}@uom.gr

Abstract. This paper describes our work towards the implementation of **MAD SWAN**, an open-source web-based application that comprises a web service registry, an XML editor, as well as manual and automatic web service composition modules. **MAD SWAN** supports various stages of web service composition and tackles the inherent non-determinism of the domain.

Keywords: Web Service, PPDDL, non-determinism, OWL-S, composition.

1 Proposed Approach

MAD SWAN is an online application related to the composition of Web Services (WSC). The users of **MAD SWAN** (an anagram of “**M**anual **A**ND **A**utomatic **S**emantic **W**sc”) are currently able to advertise a new web service (WS) in a registry, retrieve and edit the ones already in it, and manually create workflows based on OWL-S control constructs that can be bound to already stored WS descriptions. The ultimate goal is to be able to generate a composite process model automatically, based on contingency planning and a prior translation of the original WSC domain, described in OWL-S, to the widely adopted AI planning language PPDDL [1]. The whole WSC process will be evaluated against pre-defined use case scenarios and quantitative criteria, as well as a comparison between the automatic and semi-automatic modules.

Our work is motivated by the fact that very few WSC systems tackle the problem of non-determinism, whereas the recent literature suggests a gap in their evaluation. To solve the WSC problem in a non-deterministic environment, we adopt a complete search algorithm that exhaustively generates solution plans for the various contingency combinations, until a time limit set by the users is reached. A, possibly suboptimal, contingency plan can then be constructed by linking these plans through searching for natural join points. To tackle the latter problem, we present an evaluation approach, which can be used and reproduced by other systems, comprising of detailed use case scenarios that are based on an existing, open test collection.

MAD SWAN abides by the main goal of WSCs, that is, maximizing the reuse of loosely coupled components; for our implementation we make use of customized versions of already freely available components, such as iServe [2] and PetalsBPM [3]; the former is used as the basis for an online registry, and the latter is modified in order to be used for the manual composition of WSCs (instead of BPMN diagrams). Furthermore, **MAD SWAN** follows the de facto WSC and planning standards, i.e., OWL-S and (P)PPDDL. The system’s lifecycle can be summarized as follows: OWL-S

TC [4] v.4 is used throughout all the WSC stages, that is, the registry contains all of its WS descriptions, which the user can search for, edit and use for the creation of a composite WS. For that purpose, the WSs comprising the registry, or a subset of those, are translated to PPDDL. The solution to it is produced by the contingent planner, and it is translated to an OWL-S description.

We designed three use case scenarios, each with an increasing amount of non-determinism and complexity. All the WSs used are taken from OWL-S TC, with minor modifications in some of the descriptions. As a result, the evaluation process presented here can be used by a variety of WSC approaches, as a common test bed. The display format of the resulting workflow for one of the scenarios is shown in Figure 1b. A detailed description of the use case scenarios, and an analysis of the modules used in MAD SWAN can be found in [5]. Figure 1a presents part of the application, specifically the registry (on top), along with the XML editor.

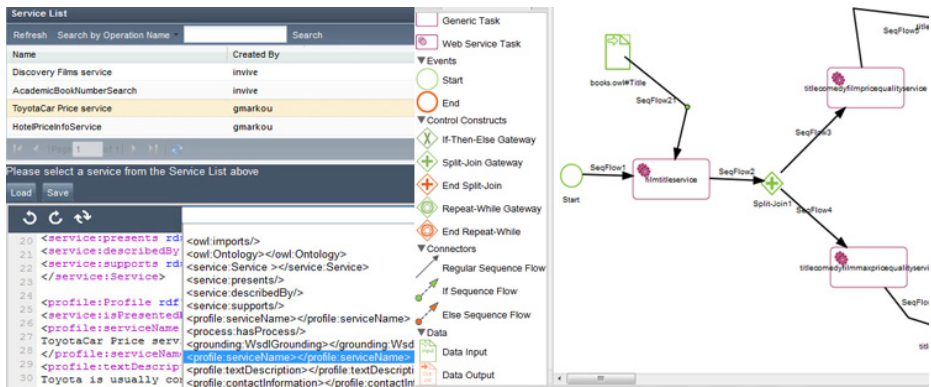


Fig. 1. a. Application's interface and editor **Fig. 1b.** Manual WSC module and sample workflow

The full size images of Figure 1 are available in [6], along with the output workflows for all the use case scenarios, a graphical overview of the presented system, and an example of the planning algorithm.

References

1. Younes, H., Littman, M.: PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report, Carnegie Mellon University (2004)
2. Pedrinaci, C., Liu, D., Maleshkova, M., et al.: iServe: A Linked Services Publishing Platform. In: ORES 2010, Hersonissos, Crete, Greece (2010)
3. Petals BPM architecture overview, <http://goo.gl/647UF>
4. OWL-S Service Retrieval Test Collection, <http://goo.gl/ehmCG>
5. Markou, G., Refanidis, I.: Towards an Automatic Non-Deterministic Web Service Composition Platform. In: NWeSP 2012, Sao Carlos, Brazil (2012)
6. MAD SWAN - ESWC 2013 Images (2013), <http://goo.gl/ZzEOG>

Longitudinal Queries over Linked Census Data

Albert Meroño-Peñuela^{1,2}, Rinke Hoekstra¹, Andrea Scharnhorst²,
Christophe Guéret², and Ashkan Ashkpour³

¹ Department of Computer Science, VU University Amsterdam, NL
`albert.merono@vu.nl`

² Data Archiving and Networked Services, KNAW, NL

³ International Institute of Social History, Amsterdam, NL

Abstract. This paper discusses the use of semantic technologies to increase quality, machine-processability, format translatability and cross-querying of complex tabular datasets. Our interest is to enable longitudinal studies of social processes in the past, and we use the historical Dutch censuses as case-study. Census data is notoriously difficult to compare, aggregate and query in a uniform fashion. We describe an approach to achieve this, discussing results, trade-offs and open problems.

1 Introduction

Census data have great value in the historical study of society. In the Netherlands, the Dutch historical censuses (1795-1971) are among the most frequently consulted statistic sources of the Central Bureau of Statistics [1]. During the period they cover, it is the only regular statistical population study performed by the Dutch government, and the only not strongly distorted historical dataset on population characteristics. Various efforts have been taken to make these data digitally available. Digitized images aside, the original censuses have been manually entered into 507 Excel workbooks and 2,288 tables.

The dataset is very difficult to use in its current form. The tables are not well self-described, and researchers must spend hours of manual analysis. The variety of variables in the dataset, on the one hand, and concept drift [3] through time (i.e. changes in the meaning of concepts, e.g. shoemakers evolved from *leather workers* to *company owners*), on the other hand, make automated, longitudinal analyses impracticable. We discuss issues, implementations and open problems on format transformations, quality checking and dataset interlinking.

2 Longitudinal Linked Census Data

Our approach is based on the Linked Data paradigm. We apply a set of standard vocabularies to make census data queryable and inter-linkable with other hubs of historical socio-economic and demographic data.

We translate the dataset to RDF with a supervised XLS/CSV to RDF converter, TabLinker¹. TabLinker produces an RDF named graph per census table

¹ <https://github.com/Data2Semantics/TabLinker>

expressed using the Data Cube Vocabulary [2]. This is a semi-automatic process: first experts need to mark manually the tables, and then TabLinker produces the T/A-Box according to table styles and data. Secondly, the census dataset contains 33,283 annotations attached to original values with data corrections, descriptions and interpretations that we translate to RDF using the Open Annotation Data Model. Query responses are provided in multiple formats (tabular, tree-structured, graph-structured, relational) as requested on demand by users.

However, the straightforward conversion of census tables into RDF does not solve cross querying across multiple tables *per se*. A valid longitudinal query would be, e.g., *get the evolution on the number of shoemakers in Amsterdam from 1795 to 1971*. But what counts as a *shoemaker* varies in meaning and labeling in different tables. Solving these queries requires additional, more abstract knowledge on these variations. We propose a two-fold approach to integrate this knowledge. First, as a top-down strategy, we transform existing taxonomies like the Historical International Standard Classification of Occupations (HISCO) into RDF/SKOS, and map disparate census table values with similar meaning into this general scheme. Second, as a bottom-up approach, we consider the specific local T-Boxes created by TabLinker for each table and we gradually generalize them. We do this by ways of mapping (e.g. using Silk), and manually creating upper ontologies with more abstract entities. Our evaluation relies on replicating results of previous socio-historical work on the census [1].

3 Conclusion

In this paper we present an overview of our efforts to improve quality, machine-processability, format translatability and cross-querying of complex historical tabular datasets. We advocate a supervised conversion tool to translate these datasets into RDF. We illustrate an hybrid top-down/bottom-up approach to build multi-layered ontologies, allowing us to query data longitudinally.

Acknowledgements. The work on which this paper is based has been partly supported by the Computational Humanities Programme of the Royal Netherlands Academy of Arts and Sciences, under the auspices of the CEDAR² project. For further information, see <http://ehumanities.nl>.

References

1. Boonstra, O., et al.: Twee eeuwen Nederland geteld. Onderzoek met de digitale Volks-, Beroeps- en Woningtellingen 1795–2001. DANS en CBS (2007)
2. Meroño-Peñuela, A., Ashkpour, A., Rietveld, L., Hoekstra, R., Schlobach, S.: Linked humanities data: The next frontier? a case-study in historical census data. In: Proceedings of the 2nd International Workshop on Linked Science (LISC 2012). International Semantic Web Conference, ISWC (2012), <http://ceur-ws.org/Vol1-951/>
3. Wang, S., Schlobach, S., Klein, M.C.A.: What is concept drift and how to measure it? In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 241–256. Springer, Heidelberg (2010)

² <http://www.cedar-project.nl/>

Deciphering Location Context – A Semantic Web Approach

Zhenning Shangguan^{1,2} and Deborah L. McGuinness²

¹ Pitney Bowes, 27 Waterview Drive, Shelton, CT 06484, USA
zhenning.shangguan@pb.com

² Rensselaer Polytechnic Institute, 105 8th Street, #2104, Troy, NY 12180, USA
d1m@cs.rpi.edu

1 Introduction

Mobile user location data has been commercially exploited and studied due to the commoditization of GPS position sensors and the popularity of Location Based Services (LBS). Context researchers have already studied how to understand human mobility using location histories [1], and how to model location context using ontologies [2]. However, these studies make surprisingly little use of rich geospatial data and knowledge about the world to a) explicitly describe user locations, and b) possibly infer implicit contexts. In this paper, we demonstrate that openly accessible geospatial data can facilitate both a) and b), thus resulting in improved understanding of mobile user location context.

2 Approach

We aim to understand the user's location context leveraging rich semantic geospatial data about the world. More concretely, we seek 1) *explicit* information that describes the user's current location point and surroundings, and 2) *implicit* insights that describe the user's activity status, situation, or intent. We call the former *static location context*, and the latter *dynamic location-informed user context*. Being critical information that can be used to characterize the situation of the user, both are considered to be the constituent parts of what we refer to as *location context*. An example scenario includes a particular mobile user's current location with timestamp, e.g., `<latitude=41.055861, longitude=-73.517434, timestamp=2013-02-15 08:30:00 EST-0500>`, and optionally some prior knowledge about the user, such as his home city of Stamford, CT, and work city of Shelton, CT. In this case, we know that his current position is on Highway Interstate 95 (I-95) North Bound (*static location context*), and we can also infer that he is on his way to work (*dynamic location-informed user context*). We build a geospatial knowledge graph to aid in resolving static location context. Dynamic location-informed context is obtained using SPARQL queries extended with geospatial functionalities.

Static Location Context. Fig. 1 shows the graph representation of a small part of Linked Geo Data (LGD) [3] used in context understanding. We also integrate proprietary commercial geospatial data with LGD, resulting in a much larger knowledge base

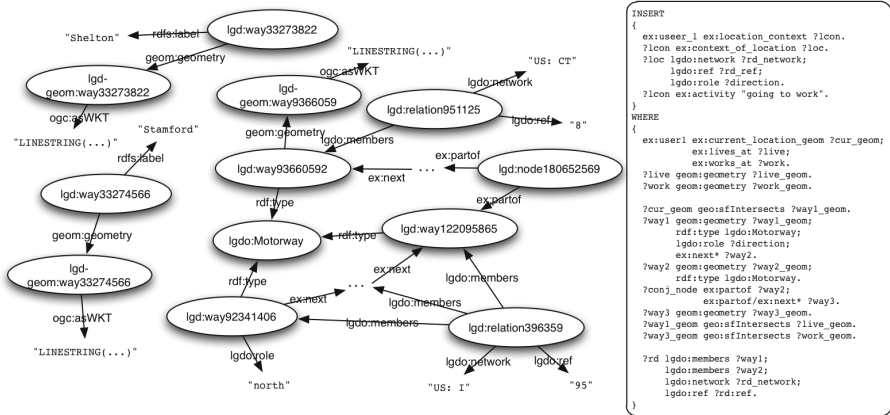


Fig. 1. Example Geospatial Knowledge Graph and SPARQL Query

(currently over 700 million triples) with socioeconomic data and geographic boundaries of neighborhoods, businesses, and residential properties.

Explicit Location-Derived User Contexts. The geospatial knowledge graph enables various common sense heuristics for understanding user contexts to be formally expressed in the form of SPARQL queries. Fig. 1 shows a sample SPARQL query to understand the user’s contexts discussed in the previous example. Notice how geospatial operations and property paths are used to retrieve his static location context (i.e., Highway Interstate 95 (I-95) North Bound) and to infer dynamic context (i.e., “going to work”) given the former. Please note that, for illustration purposes, geospatial operations are expressed using GeoSPARQL, which is currently not standardized.

3 Conclusion and Future Work

We proposed and demonstrated the feasibility of a novel approach to understand mobile user location context leveraging SPARQL and geospatial data available in the web of Linked Open Data. Future work includes a semantic model of a person’s location context, and spatial-temporal context inference mechanisms.

References

1. González, M.C., Hidalgo, C.A., Barabási, A.-L.: Hidalgo, and Albert-László Barabási: Understanding Individual Human Mobility Patterns. *Nature* 453, 779–782 (2008)
2. Chen, H., Finin, T., Joshi, A.: An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review* 18, 197–207 (2004)
3. Stadler, C., Lehmann, J., Höffner, K., Auer, S.: *LinkedGeoData: A Core for a Web of Spatial Open Data*. *Semantic Web Journal* (2012)

Comparative Classifier Evaluation for Web-Scale Taxonomies Using Power Law

Rohit Babbar, Ioannis Partalas, Cornelia Metzigg,
Eric Gaussier, and Massih-reza Amini

Laboratoire d'Informatique de Grenoble, Université Joseph Fourier, Grenoble, France
{firstname.lastname}@imag.fr

Abstract. In the context of web-scale taxonomies such as Directory Mozilla(www.dmoz.org), previous works have shown the existence of power law distribution in the size of the categories for every level in the taxonomy. In this work, we analyse how such high-level semantics can be leveraged to evaluate accuracy of hierarchical classifiers which automatically assign the unseen documents to leaf-level categories. The proposed method offers computational advantages over k -fold cross-validation.

1 Introduction

The existence of power law for explaining the underlying phenomena has been observed in a variety of domains including sizes of cities [3], internet topologies [1]. In large scale textual hierarchies, the distribution of documents in nodes at individual levels, is also shown to exhibit similar behavior [2]. Directory Mozilla, a rooted tree taxonomy, lists over 5 million websites among 1 million categories and is maintained by close to 100,000 human editors. Hence, there is a need for automatic classification of unseen documents to the desired categories. In this work, we propose an approach which captures the high level semantics of such taxonomies to efficiently evaluate the performance of a hierarchical classifier.

2 Proposed Approach

For the purpose of our work, we assume taxonomies in the form of rooted-tree in which the training documents are at leaves. The training documents for a classifier at a particular node in the taxonomy consist of all the documents of the leaves in the subtree rooted at that node. The number of training documents in the j -th ranked category (according to number of documents) for level i , denoted by N_{ij} , can be represented by [2]: $N_{ij} \approx N_{i1}j^{-\alpha_i}$, where $\alpha_i > 0$ is a level specific parameter. This distribution is shown in Figure 1, for level-5 of the DMOZ dataset extracted from the LSHTC-2011 challenge (<http://lshtc.iit.demokritos.gr/>).

The proposed strategy relies on the common assumption in machine learning that training and test data come from the same distribution. As shown in Figure 1, the curves for training and test data represented by plus and square signs receptively are parallel to each other. Using this intuition to compare the relative

accuracies of two hierarchical classifiers, the curve for the better classifier is likely to be more parallel to that of training data. Suppose C^* , represents the current best classifier, for a new classifier C' to be better than C^* :

Condition: The power law exponent of C' should be more closer to that of training data than the power law exponent of C^* .

The commonly used method of k -fold cross-validation, on other hand, is computationally expensive especially for web-scale data having tens of thousands of categories. Using a separate held-out set for the purpose of classifier evaluation leads to wasting training data which is scarce for rare categories as shown in [2].

Experiments. We performed experiments with 50,538 training documents among 4,787 leaves in a tree of depth 6. Support Vector Machine (SVM) and Multinomial Naive Bayes (MNB) classifiers were trained to evaluate the performance on a test set of size 13,057. Figures 1(a) and 1(b) show the distribution of test documents among categories after classification by MNB and SVM respectively. Two conclusions are apparent by observing Figure-1:

- 1) The plot for SVM in Figure 1(b) is much more parallel to training data (and test data) than MNB, supporting the fact that SVM with accuracy of 52.3% is better than MNB whose accuracy is 36.5%.
- 2) The number of documents in the highest ranked category are 136 and 299 for SVM and MNB respectively, and the true value is 94.

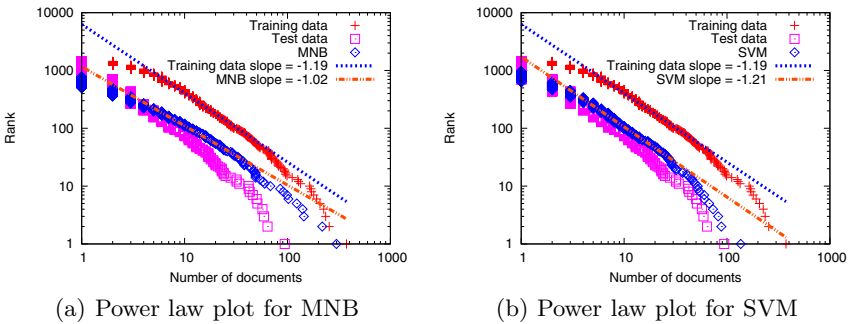


Fig. 1. Power Plots for MNB and SVM

References

1. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On power-law relationships of the internet topology. In: SIGCOMM
2. Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z., Ma, W.-Y.: Support vector machines classification with a very large-scale taxonomy. In: SIGKDD (2005)
3. Newman, M.E.J.: Power laws, Pareto distributions and Zipf's law. Contemporary Physics (2005)

NERITS - A Machine Translation Mashup System Using Wikimeta and DBpedia

Kamel Nebhi, Luka Nerima, and Eric Wehrli

LATL, Department of linguistics
University of Geneva
Switzerland
`firstname.name@unige.ch`

Abstract. Recently, Machine Translation (MT) has become a quite popular technology in everyday use through Web services such as Google Translate. Although the different MT approaches provide good results, none of them exploits contextual information like Named Entity (NE) to help user comprehension.

In this paper, we present NERITS, a machine translation mashup system using semantic annotation from Wikimeta and Linked Open Data (LOD) provided by DBpedia. The goal of the application is to propose a cross-lingual translation by providing detailed information extracted from DBpedia about persons, locations and organizations in the mother tongue of the user. This helps at scaling the traditional multilingual task of machine translation to cross-lingual applications.

Keywords: Mashup, Machine Translation, Named Entity Recognition, Linked Open Data.

1 Motivation

Machine Translation (MT) has become increasingly commonplace in everyday use through Web services such as BabelFish, Bing Translator or Google Translate. Although the different MT approaches provide good results, none of them exploits contextual information like named entity - such as the names of persons, organizations or locations - to help user comprehension. In addition, communication across languages and cultures has become vitally important, especially now with the globalization of Internet. In this context, cross-lingual knowledge bases like DBpedia can be used to connect structured information across languages. This helps at scaling the traditional multilingual task of machine translation to cross-lingual applications.

In this paper, we present NERITS, a machine translation system using semantic annotation provided by Wikimeta and LOD [5] from DBpedia. Initially, the mashup application translates a sentence using our MT Web service ITS-2¹(Interactive Translation System). Then, we use the Wikimeta API² in

¹ <http://latlapps.unige.ch/Translate>

² <http://www.wikimeta.com/>

order to extract named entities in the source sentence and to establish a link to the DBpedia databank. Finally, we give detailed information about the named entities using the DBpedia triplestore.

The goal of NERITS is to propose a cross-lingual translation by providing a set of information about persons, locations and organizations in the mother tongue of the user.

This article is structured as follows: section 2 describes our MT system ITS-2; section 3 provides details on the proposed mashup approach. We conclude and give some perspectives in section 4.

2 ITS-2 : An Interactive Translation System

2.1 Overview

ITS-2 is a large-scale translation system developed in our laboratory, LATL, in the last couple of years [11,13]. The language pairs supported are: English-French, German-French, Italian-French, Spanish-French, French-German and French-English.

At the software level, an object-oriented design has been used, similar to the design adopted for the Fips multilingual parser on which it relies [12]. To a large extent, ITS-2 can be viewed as an extension of the parser. It relies heavily on the detailed linguistic analysis provided by the parser for the supported languages, and exploits the lexical information of its monolingual lexicons. Both systems aim to set up a generic module which can be further refined to suit the specific needs of, respectively, a particular language or a particular language-pair.

The system is based on the familiar transfer architecture, with its three main components, parser, transfer and generation. First, the input sentence is parsed, producing an information-rich phrase-structure representation with associated predicate-argument representations. The parser also identifies multi-word expressions such as idioms and collocations [8] – crucial elements for a translation system .

Then, the transfer module maps the source-language abstract representation into the target-language representation. Given the abstract nature of this level of representation, the mapping operation is relatively simple and can be sketched as follows: recursively traverse the source-language phrase structure in the order: head, right sub-constituents, left subconstituents. Lexical transfer (the mapping of a source-language lexical item to an equivalent target-language item) occurs at the head-transfer level (provided the head is not empty); it yields a target-language equivalent term, often (but by no means always) of the same category. Following the projection principle used in the parser, the target-language structure is projected on the basis of the lexical item which is its head.

However, the projections (i.e., constituents) which have been analyzed as arguments of a predicate undergo a slightly different transfer process, since their precise target-language properties may be in part determined by the subcategorization features of the target-language predicate. To take a simple example, the direct object of the French verb *regarder* in (1-a) will be transferred to English

as a prepositional phrase headed by the preposition *at*, as illustrated in (2-a). This information comes from the lexical database. More specifically, the French-English bilingual lexicon specifies a correspondence between the French lexeme [VP regarder NP] and the English lexeme [VP look [PP at NP]]. For both sentences, we also illustrate the syntactic structures as built by the parser and/or the generator of ITS-2:

- (1) a. Paul regardait la voiture.
 b. [TP [DP Paul] regardait_i [VP e_i [DP la [NP voiture]]]]
- (2) a. Paul was looking at the car.
 b. [TP [DP Paul] was [VP looking [PP at [DP the [NP car]]]]]

2.2 Evaluation

The last evaluations of ITS-2 were for the Fifth and Seventh Workshop on Statistical Machine Translation [2,3]. The LATL participated in the French-English and English-French tasks. The table 1 shows the best results obtained by ITS-2 in terms of BLEU³ [7] and Translation Edit Rate⁴ (TER) [10] using the newstest2010 and newstest2012 corpus as evaluation set.

Table 1. Translation results from French to English and English to French measured on newstest2010 and newstest2012

Pair of language	BLEU	TER
French-English	16.5	0.785
English-French	21.8	0.684

3 The Proposed Approach

NERITS (Named Entity Recognition for the Interactive Translation System) is a Web application plugged on top of various tools such as ITS-2 machine translation, Wikimeta semantic annotation platform and DBpedia triplestore.

3.1 Architecture

For our mashup, we used a layered architecture that includes: data retrieval using ITS-2 and Wikimeta Web services, data integration using DBpedia knowledge base, and the user interface.

³ BLEU is currently the standard in MT evaluation. It calculates n-gram precision and a brevity penalty, and can make use of multiple reference translations as a way of capturing some of the allowable variation in translation.

⁴ TER calculates the number of edits required to change a hypothesis translation into a reference translation. The possible edits in TER include insertion, deletion, and substitution of single words, and an edit which moves sequences of contiguous words.

Actually, the mashup application provides French to English and English to French translation. The figure 1 shows the architecture of our system.

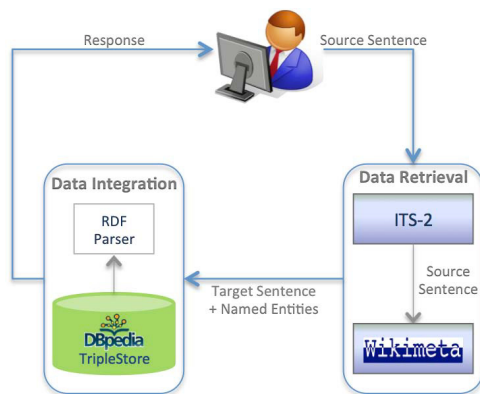


Fig. 1. NERITS Mashup Architecture

Data Retrieval using Web Services. The following Web services were used to build the mashup:

1. **ITS-2**: our ReST Web service [6] generates a translation using the machine translation ITS-2 (cf. section 2);
2. **Wikimeta** [4]: the source sentence is analyzed by the semantic annotation platform which performs the traditional task of named entity recognition and also the task of named entity linking⁵ [9]. Wikimeta proposes an approach based on a resource of contextual words called *Linked Data Interface* (LDI). The LDI associates several metadata⁶ to each entity described in Wikipedia. The disambiguation task uses the *Semantic Disambiguation Algorithm* (SDA), which identifies the item in the LDI that is most similar to the context (the context is represented by the set of words that appear around the NE) of the named entity. The system shows promising results with 90 percent of recall for French and 86 percent of recall in English. The listing 1.1 shows the Wikimeta categorization for the NE “Jean-Marc Ayrault”.

```

<extraction>
  <NE>Jean-Marc Ayrault</NE>
  <type>PERS.HUM</type>
  <LOD>http://www.dbpedia.org/resource/Jean-Marc_Ayrault</LOD>
</extraction>
  
```

Listing 1.1. Wikimeta categorization for the NE “Jean-Marc Ayrault”

⁵ Entity linking is the task to link the entity mention in text with the corresponding entity in the existing knowledge bases like DBpedia or GeoNames.

⁶ A set of surface forms, a set of words that are contained in the entity description and an URI that points to some entity in the LOD Cloud.

Data Integration. NERITS integrates data by using Semantic Web technology, in particular LOD as DBpedia. DBpedia [1] defines LOD URIs for millions of concepts by extracting structured information from Wikipedia.

The DBpedia databank contains:

- Data about persons, locations, organizations, music albums, etc.
- 4 million datasets described by 1,2 billions of triples
- 7 million of external RDF links to Freebase, GeoNames, YAGO, etc.

The listing 1.2 is a part of the RDF/XML representation for the DBpedia entry “Jean-Marc Ayrault”. These data provide several informations about “Jean-Marc Ayrault” such as his birth date, his place of birth, his occupation, etc.

```

<rdf>
  <rdfs:label>Jean-Marc Ayrault</rdfs:label>
  <dbpedia-owl:orderInOffice>Prime Minister of France</dbpedia-owl:orderInOffice>
  <dbpprop:birthDate>1950-01-25
</dbpprop:birthDate>
  <dbpprop:placeOfBirth>Maulevrier , Maine-et-Loire France</dbpprop:placeOfBirth>
  <dbpprop:spouse>Brigitte Terrien</dbpprop:spouse>
  <foaf:depiction rdf:resource="http://upload.wikimedia.org/wikipedia/commons/b/be/Jean-Marc_Ayrault_-_mars_2012.jpg" />
</rdf>

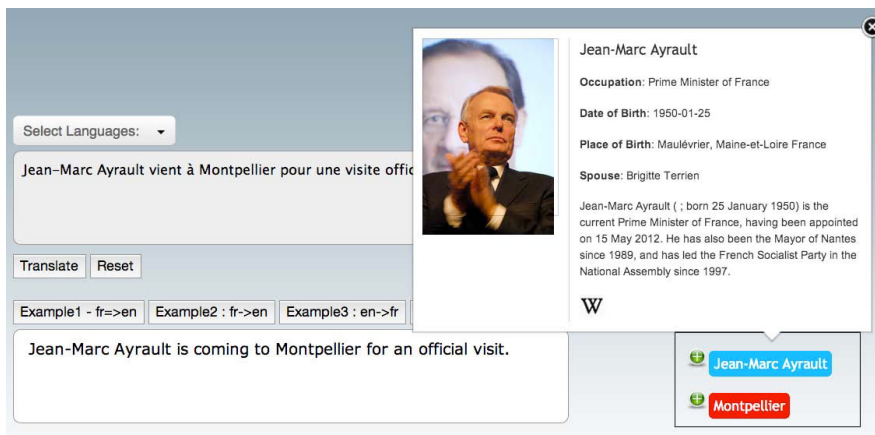
```

Listing 1.2. Part of the RDF/XML representation for the DBpedia entry “Jean-Marc Ayrault”

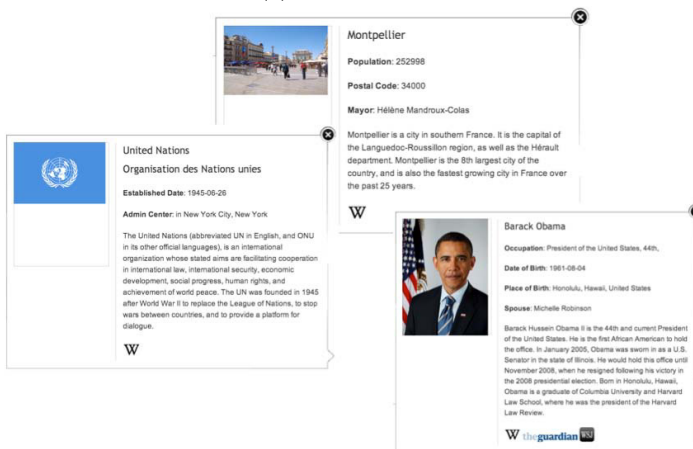
The processing of the data integration module relies on two main steps : when the Wikimeta API recognizes and annotates entities in the source sentence, a query is issued to the DBpedia Triplestore; then a PHP module parses and converts the RDF file into HTML. For our mashup, informations extracted from DBpedia are available in English and French.

User Interface. The user interface is freely available at <http://cms.unige.ch/lettres/linguistique/nebhi/nerits/>.

The figure 2a is a screenshot of the user interface for the translated sentence “Jean-Marc Ayrault vient à Montpellier pour une visite officielle”. In a first step, NERITS provides the translation of the source sentence. Then, if the source sentence contains NE, the mashup application provides detailed information about persons, locations and organizations. For the example “Jean-Marc Ayrault”, it gives the following information: occupation, date of birth, place of birth, spouse, a brief biography and a link to Wikipedia. For the city “Montpellier”, the figure 2b shows the following information: population, country, mayor, a brief description and a link to Wikipedia.



(a) User Interface



(b) Other Examples

Fig. 2. NERITS screenshots

The figure 2b also shows other examples provided by NERITS. For the organization “UN”, it gives information such as the established date, the location of the admin center, a brief description and a link to Wikipedia. For the example “Barack Obama”, in addition to conventional information, it also provides links to a set of newspaper articles from the Guardian and/or the Wall Street Journal.

4 Conclusion - Further Work

In this paper, we have presented a machine translation mashup system using semantic annotation provided by Wikimeta and Linked Open Data. We have given a first glance on how semantic annotation and information from DBpedia can be combined to help user comprehension.

In future work, we plan to incorporate recognition of events (such as *September 11 attacks* or *Fall of the Berlin Wall*). We'll also try to make better use of Linked Open Data in order to provide the points of interest of a Location (for example "Tour Eiffel" in Paris) or people around a Person (for example "Hillary Clinton" is in Barack Obama's entourage). Finally, we'll add other machine translation systems (such as Google Translate, Bing Translator and Babelfish) and languages as German-French and French-German.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia – A Crystallization Point for the Web of Data. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* (7), 154–165 (2009)
2. Callison-Burch, C., Koehn, P., Monz, C., Peterson, K., Przybocki, M., Zaidan, O.: Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation. In: *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics* MATR, Uppsala, Sweden, pp. 17–53. *ACL* (2010)
3. Callison-Burch, C., Koehn, P., Monz, C., Post, M., Soricut, R., Specia, L.: Findings of the 2012 Workshop on Statistical Machine Translation. In: *Proceedings of the Seventh Workshop on Statistical Machine Translation*, Montréal, Canada, pp. 10–51. *ACL* (2012)
4. Charton, E., Gagnon, M., Ozell, B.: Automatic semantic web annotation of named entities. In: *Canadian Conference on AI*, pp. 74–85 (2011)
5. Heath, T., Bizer, C.: *Linked Data: Evolving the Web into a Global Data Space*. *Synthesis Lectures on the Semantic Web*. Morgan & Claypool Publishers (2011)
6. Nebhi, K.: A ReSTful Web Service for multilingual LRT. In: *3rd International Conference on the Future of Information Sciences (INFuture): Information Sciences and e-Society*, Zagreb, Croatia (2011)
7. Papineni, K., Roukos, S., Ward, T., Zhu, W.: Bleu: A method for automatic evaluation of machine translation. In: *Proceedings of ACL* (2002)
8. Seretan, V., Wehrli, E.: Accurate collocation extraction using a multilingual parser. In: *ACL* (2006)
9. Shen, W., Wang, J., Luo, P., Wang, M.: Linden: linking named entities with knowledge base via semantic knowledge. In: *Proceedings of the 21st International Conference on World Wide Web, WWW 2012*, pp. 449–458. *ACM* (2012)
10. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: *Proceedings of Association for Machine Translation in the Americas* (2006)
11. Wehrli, E.: Translating Idioms. In: *Proceedings of COLING 1998*, Montreal, pp. 1388–1392 (1998)
12. Wehrli, E.: Fips, a deep linguistic multilingual parser. In: *ACL 2007 Workshop on Deep Linguistic Processing*, pp. 120–127. Czech Republic, Prague (2007)
13. Wehrli, E., Nerima, L., Scherrer, Y.: Deep linguistic multilingual translation and bilingual dictionaries. In: *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pp. 90–94. *ACL* (2009)

SNARC - An Approach for Aggregating and Recommending Contextualized Social Content

Ahmad Assaf¹, Aline Senart¹, and Raphael Troncy²

¹ SAP Research, SAP Labs France SAS,
805 avenue du Dr. Maurice Donat, BP 1216, 06254 Mougins Cedex, France
`first.last@sap.com`

² EURECOM,
2229 route des cretes, 06560 Sophia Antipolis, France
`raphael.troncy@eurecom.fr`

Abstract. The Internet has created a paradigm shift in how we consume and disseminate information. Data nowadays is spread over heterogeneous silos of archived and live data. People willingly share data on social media by posting news, views, presentations, pictures and videos. SNARC is a service that uses semantic web technology and combines services available on the web to aggregate social news. SNARC brings live and archived information to the user that is directly related to his active page. The key advantage is an instantaneous access to complementary information without the need to dig for it. Information appears when it is relevant enabling the user to focus on what is really important.

Keywords: Social news, social networks, aggregator, data mining, mashup.

1 Introduction

With the rapid advances of the Internet, social media become more and more intertwined with our daily lives. The ubiquitous nature of Web-enabled devices, especially mobile phones, enables users to participate and interact in many different forms like photo and video sharing platforms, forums, newsgroups, blogs, micro-blogs, bookmarking services, and location-based services. Social networks are not just gathering Internet users into groups of common interests, they are also helping people follow breaking news, contribute to online debates or learn from others. They are transforming Web usage in terms of users' initial entry point, search, browsing and purchasing behavior.[1] A common scenario that often happens while reading an interesting article, coming across a nice video or participating in a discussion in a forum is the growing interest to check related material around the information read. To do so, users might go to Twitter¹,

¹ <http://www.twitter.com>

Google+² or YouTube³. They can try several times with several keywords to obtain the desired results. In the end, they might end up with several browser tabs opened and get distracted by the information overload from all these resources. The same happens in companies when business users are interested in information provided by corporate web applications like enterprise communities. SNARC is a semantic social news aggregator that leverages live rich data that social networks provide to build an interactive rich experience on the Internet. The service retrieves news related to the current page from popular platforms like Twitter, Google+, YouTube, Vimeo⁴, Slideshare⁵, Stackoverflow⁶ and the Web. As a possible front-end implementation, we have created a Google Chrome extension (visit <http://ahmadassaf.com/SNARC>) which enriches the user experience by augmenting related contextual information to entities on the page itself, as well as displaying related social news on a floating sidebar.

The remainder of this paper is split into three main sections. The first talks about the underlying mechanism of the service, splitting the functionalities into three main subsections. The second describes the front-end implementation, and the last talks about our conclusion and future work.

2 Underlying Mechanism

The back-end of SNARC consists of three major components: a document handler that creates a "Semantic Model" that represents any web resource, a query layer that is responsible for disseminating queries to the supported social services and a data parser which processes the search results, wraps them in a common social model and generates the desired output.

2.1 Document Handler

The main idea behind SNARC is to provide a uniform model for web entities, whether they are blog entries, multimedia objects or micro-posts. To do so, SNARC creates a "Semantic Model" containing all the annotations and meta-data needed to query and reconcile social results.

The Semantic Model is created by the Document Handler (see Figure 1) which receives a web page URL and performs these three main steps:

1. **Text Extraction:** Fetch the webpage that corresponds to the received URL and extract the textual content using a set of heuristics. These latter identify the main content of the page by stripping unwanted HTML tags and rank the different sections based on their semantics, class names and order. In the beginning we have used Alchemy API⁷ to perform text extraction; but

² <http://plus.google.com>

³ <http://www.youtube.com>

⁴ <http://www.vimeo.com>

⁵ <http://www.slideshare.com>

⁶ <http://www.stackoverflow.com>

⁷ <http://www.alchemyapi.com>

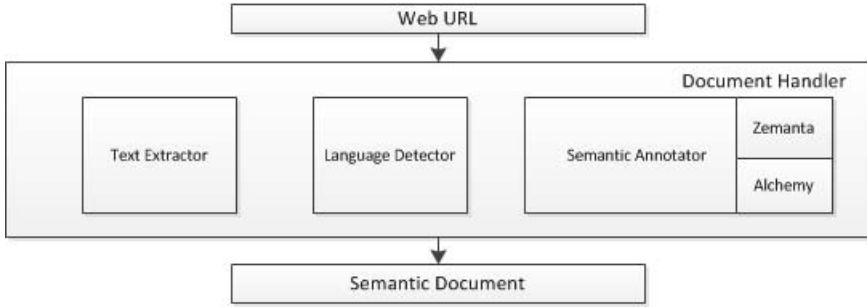


Fig. 1. SNARC’s Document Handler

we have chosen to implement a simpler method ourselves which saved us an extra API call.

2. **Language Detection:** Detect the web page language using the Language Detection service of Alchemy API. This is necessary to match the desired language with compatible services like Twitter, YouTube, etc.
3. **Semantic Annotation:** Annotating the extracted text is the most important step in this process. We use Zemanta Suggest⁸ and Alchemy API in order to extract:

- **Tags:** These are the finest-grained queryable ”keywords” that we use to retrieve the social results. From our experiments, combining tags results in better findings than using entities or concepts. However, we plan to evaluate the combination of keywords, entities and concepts in order to find the top-queryable terms that will retrieve the most relevant results on different abstraction levels.

Tags retrieved from these services are ranked by confidence values calculated by their internal algorithms, these values are normalized for each service. According to our experiments we have found that Alchemy’s Keywords Extraction API returns a large set of closely related keywords (i.e. Android, Android Phone, Android Tablet, ...). To construct a good query we therefore need to provide a certain level of abstraction. We perform a cleaning process on those keywords by applying the Levenshtein distance to rule out closely related keywords by disregarding those with lower confidences. We perform a similar process on the result of the union between the keywords returned by Alchemy and Zemanta to ensure a sparse keywords set.

- **Semantic Entities:** Entities provide a higher abstraction level of the document. They are used to reconcile the social results in order to maintain relevancy with the document. Similar to the keywords extraction services, the entities retrieved are ranked and contain outbound links to the matched entity on dbpedia, Wikipedia, Freebase, etc. A union is made between the results from Alchemy and Zemanta to ensure a wider

⁸ <http://developer.zemanta.com/docs/suggest/>

coverage of entities. When a match is found, we merge the links from the two sources to ensure that we include all the resources that can be used to augment extra information about that entity in the document.

- **Categories:** These are high-level taxonomies that can generally describe the document’s content. A taxonomy is used to narrow down our query scope when targeting services like YouTube. In our Semantic Document model we define two possible category sets, one retrieved from Alchemy’s Text Categorization API⁹ and the other retrieved from Zemanta Suggest API that follows the DMOZ categorization scheme¹⁰.

At the end of this process, we will have constructed the needed elements (keywords, entities and high level categories) wrapped in our Semantic Model to be passed to the query generator. For example, a summary of the Semantic Model for a web page titled ”Turkey protests: Erdogan in ’final’ warning¹¹” looks like:

1. **Categories:** Culture_Politics, Regional and Society
2. **Keywords:** Taksim Square, Protesters, Gezi Park, Mr Erdogan, Istanbul ...
3. **Entities:** Gezi Park, Recep Tayyip Erdogan, Taksim Square, Justice and Development Party (Turkey), Police of Turkey ...

2.2 Query Layer

In this component, the calls to the social services are made. SNARC uses the extracted keywords from the Semantic Document in order to construct the queries and disseminate them to the appropriate services. Figure 2 shows the different steps in order to retrieve a set of social results.

1. **Query Builder:** Responsible for identifying targeted services and building tailored queries for each service. For example, if the processed document is categorized as a computer or technology related one, Stackoverflow service will be targeted with the queries constructed. However, other categories will correspond to different services from the Stack Exchange websites¹².
2. **Query Federator:** Responsible for federating the queries identified in the previous step to the corresponding services. To enhance performance, we tried to reduce the number of external calls. Yahoo Query Language (YQL)¹³ helped us in minimizing the number of calls and batching them into a single one. It is an expressive SQL-like language that lets you query, filter, and join data across Web services. However, we have found that we cannot fully rely on YQL due to their API calls limit and the restriction on the query execution time that is set to 30 seconds. To overcome this, we have implemented a

⁹ <http://www.alchemyapi.com/api/categ/categs.html>

¹⁰ <http://www.dmoz.org/desc/Top>

¹¹ <http://www.bbc.co.uk/news/world-europe-22889060>

¹² <http://stackexchange.com/sites>

¹³ <http://developer.yahoo.com/yql/>

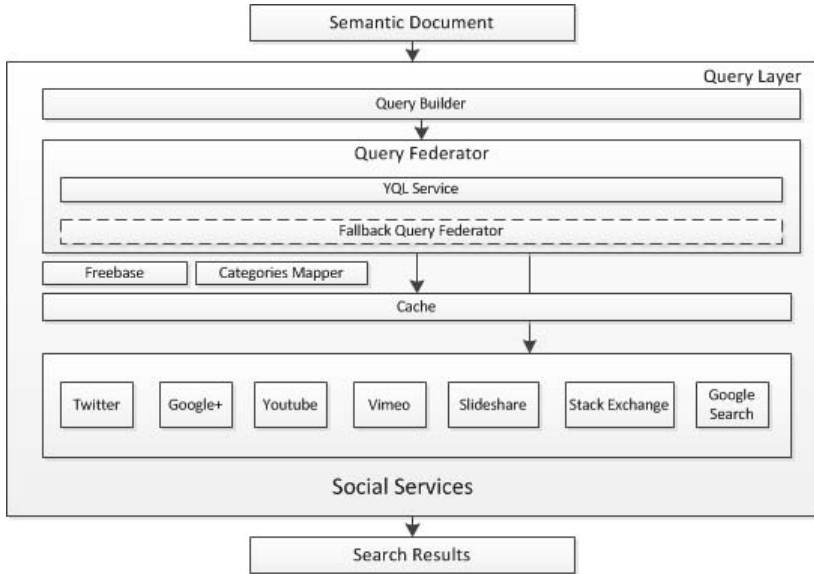


Fig. 2. SNARC’s Query Layer

fallback mechanism that federates the queries to the selected social services and groups the result to be passed afterwards to the parser.

To further optimize the number of calls, we have decided to take the top two ranked keywords. We do not apply logical operator (AND/OR) in our queries; instead, we perform one-to-one mapping between each keyword and query. Indeed, we have found that gathering keywords even if semantically related might bring up noise in the results. However, as mentioned earlier, a part of the future work will be investigating the best method to construct the most relevant queryable entity using different logical operators.

3. **Caching:** The main setback in the query layer was the variable limited number of calls we can make to external APIs. To overcome this, we have implemented a simple cache mechanism that saves the results on disk up to an hour. There are several cache levels; the first is a URL level one where the results of the parsed queries are cached. For example, if a user visited a certain article on the CNN webpage the results might take up to 15 seconds to appear, whereas a second user visiting the same article minutes afterwards will have the cached results in few seconds. The second level is keyword and service specific. This can be very helpful as users generally browse articles of related topics or interests (semantic concepts), so for each user we can end up with the same high level concepts being requested frequently. An important thing to note is that the caching is done on the server side and is disk-based.

The social services queried can be grouped as follows:

1. **Multimedia Services:** They include Slideshare, Vimeo and YouTube. Slideshare and YouTube allow the results to be fetched in a specific

language that was detected in the previous step. In addition to that, YouTube search services are called twice; the first call is done to the YouTube V2 API¹⁴ where we specify in addition to the keywords a high level category to be targeted. To do so, we have manually created a category mapping file that maps the high-levels categories of Alchemys API and DMOZ to those provided by YouTube. The second call is done to YouTube V3 API¹⁵. The new feature provided by Google in this version is the ability to search using a semantic concept that corresponds to a Freebase concept ID; it proves to retrieve better results than the normal search. Freebase concept calls are cached for longer periods as they are less prone to changes.

2. **Micro-posts Services:** They include Twitter, Google+ and Stackoverflow. Language filtering is done where applicable.
3. **General Search:** This includes similar results found via Google search or those retrieved from the Zemanta API call. They are general articles or blog posts related to the current active page.

2.3 Data Parser

This is the last step where the results are unified and wrapped in a single social model. Figure 3 shows the different steps needed to produce the final parsed results that will be pushed back to the front-end.

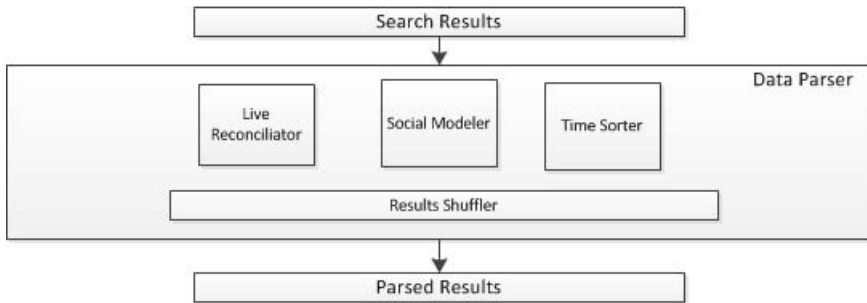


Fig. 3. SNARC's Data Parser

1. **Live Reconciliator:** Social (or folksonomic) tagging has become a trending method to describe, search and discover content on the web. Folksonomies empower users by giving them total freedom in choosing their categories and keywords that they think describe best the content. This contrasts with taxonomies that over-impose hierarchical categorization of content [6]. However, in services like Twitter and Google+, tagging has been abused in a way that increased noise in the stream of results. To overcome this problem, we align the incoming stream of posts with the set of semantic concepts

¹⁴ <https://developers.google.com/youtube/2.0/>

¹⁵ <https://developers.google.com/youtube/v3/>

or keywords that describe the document. There are several approaches and tools like [2, 3, 5, 6] that aim at solving this problem. In SNARC we rely on two levels of reconciliation: one uses the high-level taxonomy (categories); and the other uses the vector of entities defined in the Semantic Document. For example, if SNARC wants to reconcile a blog post result retrieved from a general search, it constructs a Semantic Document Model for that result and applies the Cosine Similarity on the vector of ranked entities for each Semantic Model. Currently, we only reconcile against blog posts as it is very straightforward to construct a Semantic Document Model for them. However, an integral part of the future work will be the integration of SNARC's model to micro-posts and video search services.

2. **Social Modeler:** Every social network has its own underlying data model. To overcome this problem, we need to present the social results in a common wrapper. To do so, we have created an optimized universal social model that contains all the necessary data to model social information and can be reused in other projects. The model contains service related attributes like the service name and type, general post information like the author's name, profile link, image and geo-location information and post-specific information like the title, thumbnail, embed code, main content and link.
3. **Time Sorter and Results Shuffler:** To better display the results on the front-end, we unify the time representation and sort the results based on it. Afterwards we pick the top N results and shuffle them to generate a random order.

3 Front-End

SNARC is a service that generates a JSON file containing the results wrapped in our universal social model. We have implemented a chrome extension that loads SNARC on any web page or application (see Figure 4). This UI implementation offers more flexibility to users by loading related social news anytime on any webpage or application. The results are visualized using jQuery templates as a sliding panel on one of the screen edges, extracted entities are highlighted in the page itself and a short excerpt is displayed when hovering over them.

4 Conclusions and Future Work

Aggregating relevant social news is not an easy task. SNARC performs the task in a nice and intuitive way that allows the user to discover what is happening instantly and without the need to navigate away from the current page. One of the important things to consider for the future is the integration of better reconciliation features and tools to ensure the display of relevant social posts. Moreover, real-time feature that can also push new related posts would be a great addition. We would also want to test SNARC on business web applications. It can be a good fit to perform brand monitoring especially after plugging a sentiment analysis component. We would also like to evaluate the necessity to use a content scrapping API like Alchemys as a fallback for our text extraction mechanism.

The image shows a screenshot of a web browser displaying a BBC News article titled "Turkey protests: Ruling AK party may hold vote on park". The SNARC Chrome extension is active, highlighting several entities in the text, such as "AK party" and "Istanbul project". A red box labeled "Highlighted Entities" points to these highlights. Another red box labeled "SNARC Information Box" points to a pop-up window for the "JUSTICE AND DEVELOPMENT PARTY (TURKEY)". This information box includes details about the party, such as its website, DBpedia, and Freebase links, and a brief description of its political stance. The background shows the BBC News website with various sections like "Top Stories", "Features & Analysis", and "Most Popular".

Fig. 4. SNARC's UI - The Google Chrome Extension

References

- [1] Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.: The role of social networks in information diffusion. In: Proceedings of the 21st International Conference on World Wide Web - WWW 2012, p. 519. ACM Press, New York (2012)
- [2] Cantador, I., Bellogín, A.: Semantic contextualisation of social tag-based profiles and item recommendations. *E-Commerce and Web ... (2)* (2011)
- [3] Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L., Nejdl, W.: Real-time top-n recommendation in social streams. In: Proceedings of the Sixth ACM Conference on Recommender Systems - RecSys 2012, p. 59. ACM Press, New York (2012)
- [4] Pennacchiotti, M., Gurumurthy, S.: Investigating topic models for social media user recommendation. In: WWW 2011, p. 101. ACM Press, New York (2011)
- [5] Preotiuc-Pietro, D., Samangoei, S.: Trendminer: An architecture for real time analysis of social media text. In: Sixth International AAAI Conference on Weblogs and Social Media, pp. 4-7. AAAI Publications (2012)
- [6] Zanardi, V., Capra, L.: Social ranking: uncovering relevant content using tag-based recommender systems. In: ACM Conference on Recommender Systems (2008)

Twindex Fuorisalone: Social Listening of Milano during Fuorisalone 2013

Marco Balduini¹, Emanuele Della Valle¹, Daniele Dell’Aglío¹,
Mikalai Tsytsarau², Themis Palpanas², and Cristian Confalonieri³

¹ DEIB – Politecnico di Milano, Italy

{marco.balduini, emanuele.dellavalle, daniele.dellaglio}@polimi.it

² DISI – Università degli Studi di Trento, Italy

themis@disi.unitn.eu, tsytsarau@disi.unitn.it

³ Studiolabo, Italy

cristian.confalonieri@studiolabo.com

Abstract. Fuorisalone during Milano Design Week, with almost three thousands events spread around more than six hundreds venues, attracts half a million visitors: what do they say and feel about those events? Twindex Fuorisalone is a mash-up that listens what all those visitors posted on Twitter and Instagram in that week. In this paper, we briefly report on how Twindex Fuorisalone works and on its ability to listen in real-time the pulse of Fuorisalone on social media.

1 Introduction

The Salone Internazionale del Mobile¹ is the largest furniture fair in the world. It is held in Milano every spring and it lasts a week. In the same week, thousands of satellite events are scheduled; they are grouped under the name of Fuorisalone². Those industrial design events have been gaining more and more attention in the last years. As a result Fuorisalone is one of the most important events for industrial design worldwide. Now, this week dedicated to design is called the Milano Design Week (MDW), and it attracts every year more than 500.000 visitors. This year (i.e., 2013), MDW was held in April 9-14.

While the Salone Internazionale del Mobile is held in one specific location (the Milano fair area), Fuorisalone is held in different venues (more than 600) around the city. Due to this decentralization of the event, the Fuorisalone organisers are interested in monitoring in real time the reaction of the visitors:

Q.1 What are the most attractive events?

Q.2 What do visitors say about the events they join?

Q.3 What is their mood before, during and after the events they join?

Manually collecting the information to answer those questions is complex and expensive. The state of the art consists in analysing mobile network data [1],

¹ Cf. http://www.cosmit.it/en/salone_internazionale_del_mobile

² Cf. <http://fuorisalone.it/2013/>

but only big event organisers (e.g., Olympic games' hosters) can afford its cost. Nowadays, the Web offers a cheaper way to collect the data to answer them: the rise of the social Web, e.g., Twitter³ and Instagram⁴, provides a continuous flow of information in the form of social text streams (shortly, social streams). Being able to process social streams in real-time allows to develop new services, useful not only for the aforementioned organisers (real time monitoring), but also for visitors (find the more popular events) and other subjects (e.g., Milano municipality, municipal police, etc.).

In this paper, we present Twindex Fuorisalone (TF), a mash-up that gathered, processed, and analysed social streams related to Milano Design Week using the fuorisalone.it repository of events as a source of information to make sense of the continuous flow. The techniques are similar to those previously used for disastrous events (e.g., earthquakes [2]), but are applied in the new setting of event management. In particular, this work builds on the results of the experiments we performed in the previous projects in Politecnico di Milano (i.e, BOTTARI [3], Social Listening of London Olympics 2012 [4]) and University of Trento (i.e., sentiment mining and contradiction detection on the Web [5]).

The remainder of this paper is organised as follows. Section 2 describes the data sources that TF integrates, the architecture of TF and how it processes the social streams from Twitter and Instagram against the information stored in fuorisalone.it repository. Section 3 describes the front end of TF as it was made available during MDW 2013. In Section 4, we report on the ability of TF to answer the three questions listed in this section. Finally, Section 5 concludes.

2 Data and System Architecture

In this section, we first analyse the data sets that TF consumes, then we explain its architecture and how social streams are processed.

2.1 Data Sources

As illustrated in Figure 1, TF gathers data from three sources: fuorisalone.it, Twitter and Instagram.

The *fuorisalone.it* repository is a proprietary data set that collects information about the schedule of the events. It contains about 2730 events, located in 676 venues. For each event, it describes its duration, its category and who sponsors it. The data are accessed through a Web service interface.

Twitter is the most famous microblogging service in the Web: it allows to post short text messages. The messages (tweets) have an identifier and a time stamp; additionally they can be annotated with topics (a word starting with #), user names (strings starting with @), and geo-references (WGS84 coordinates). TF accesses Twitter data through the streaming API⁵: it is a push service that

³ Cf. <http://www.twitter.com>

⁴ Cf. <http://www.instagram.com>

⁵ Cf. <https://dev.twitter.com/docs/streaming-apis>

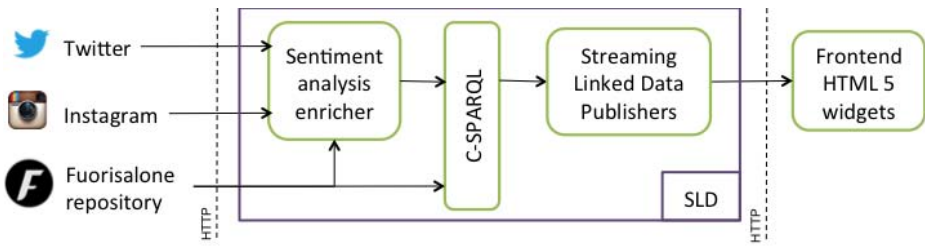


Fig. 1. Architecture of Twindex Fuorisalone

allows to obtain one or more data streams (sequences of tweets ordered by time stamp).

Instagram is another microblogging service; unlike Twitter, in this service users post images optionally associated to a short message that, in most of the cases, is a list of hashtags. Due to the fact that Fuorisalone is (mainly) composed of industrial design events, we are interested in monitoring this source as well: people taking pictures and sharing them on Instagram. The data is also accessed through a streaming API⁶.

TF subscribes three kinds of social streams on Twitter and Instagram:

- A *geo-bound stream* is associated to a rectangular area and it contains all the micro posts georeferenced with a coordinate inside the area up to a given maximum rate. The geo-bound stream used by TF contains all the micro posts with a coordinate in Milano. Note that both in Twitter and Instagram the georeference is optional, so this stream captures only a part of the micro posts related to the Fuorisalone events.
- A *topic-based stream* contains a subset of the micro posts about a given topic (a syntactic match is performed and a max rate filter is applied). Using the textual descriptions of Fuorisalone’s locations, events, and sponsors, which are stored in fuorisalone.it repository, we defined a list of 300 keywords to ask Twitter and Instagram about.
- A *user-centric stream* is associated to a social network accounts and it contains all the messages posted from those accounts. We manually inserted relevant Twitter users (e.g., @fuorisalone) whose micro posts are processed by TF.

2.2 Mash-Up Architecture and Processing

The processing of TF relies on four data-driven steps: data integration, enrichment, analysis, and visualisation. TF exploits Semantic technologies to achieve the first three steps (i.e., Semantic Web technologies to integrate and analyse the data, and opinion mining [6] to enrich it) and on HTML5 for the fourth step.

RDF is used as common data model. As vocabulary, to which the data from the different sources can be translated, TF uses the extension of Semantically-Interlinked Online Communities ontology [7] proposed in [3]. It is worth to note

⁶ Cf. <http://instagram.com/developer/realtime/>

that TF handles social streams, which are characterised by a (temporal) order and by high dynamics; to treat these data, TF uses RDF streams, an extension of the RDF model proposed in [8] and adopted by a growing community [9,10].

For instance, hereafter, we represent in RDF one of the tweets⁷ that was posted during Fuorisalone:

```
[ ] sioc:content "Fuori salone Milano 2013 #imanartist #milanodesignweek #mdw2013 pic.twitter.com/GqR7KZc2RP";
sioc:has_creator <https://twitter.com/WHOISBUMIN> ;
sioc:topic :imanartist, :milanodesignweek, :mdw2013 ;
sioc:links_to "http://pic.twitter.com/GqR7KZc2RP".
```

Data enrichment is performed through a dictionary-based sentiment classifier [5], which includes positive and negative emotion patterns. Given a micro post in English or in Italian, this technique allows to infer information about the opinion of the author (e.g., she likes or dislikes the main topic of the message). TF uses a dictionary-based sentiment classifier, because this type of classifier is known to be efficient for large-scale analysis of short texts concentrating on a single topic, such as micro posts. Moreover, since many sentiments are domain-specific, this kind of classifier is easy to adapt to the particular domain of analysis, i.e., industrial design and high tech products' launches. While this method is very suitable for large-scale analysis thanks to its minimal performance requirements, some sentiments (e.g., sarcasms, idioms) require more sophisticated methods.

The third step is the data analysis of the data collected and enriched in the previous steps. In TF, all analyses are encoded in C-SPARQL [11], an extension of SPARQL for continuous querying RDF streams as well as static RDF data sets⁸. For instance, the following C-SPARQL query counts for each hashtag the number of micro posts in a time window of 15 minutes that slides every minute.

```
1 REGISTER STREAM HashtagAnalysis AS
2 CONSTRUCT { [] sld:about ?tag ; sld:count ?n . }
3 FROM STREAM <http://.../fuorisalone2013> [RANGE 15m STEP 1m]
4 WHERE { { SELECT ?tag (COUNT(?tweet) AS ?n) WHERE { ?tweet sioc:topic ?tag . } GROUP BY ?tag } }
```

Complex analyses can be performed combining C-SPARQL queries in a network. The continuous results of the queries are published as Streaming Linked Data [12], and are used by HTML 5 visual widgets that compose the user interface. The component that supports networking of C-SPARQL queries and Streaming Linked Data publishing is named SLD [3]. Solutions alternative to SLD are presented in [13,14].

⁷ See <https://twitter.com/WHOISBUMIN/status/335490229429493760>

⁸ Similar results could have been obtained using continuous RDF stream processors like SPARQL_{stream} [9] and CQELS [10].

3 The Mash-Up

Figure 2 shows a screenshot of the Twindex Fuorisalone mash-up⁹. The top-most visual widget is a heatmap that illustrates in real time where the social streams originate from. The default view covers the entire Milano area. The users can zoom in each of the nine districts interested by Fuorisalone events by clicking on the names of the districts in the side bar on the heatmap. The users can also add to the map an overlay showing the position of the micro posts received by TF in the last 15 minutes. Clicking on the Twitter’s and Instagram’s icons, they can read the tweets and view the images.

Moving down, the second widget displays the tweets received on the user-centric stream. The third one is a bar chart that presents the volume of micro posts received on the geo-bound stream every 15 minutes in the last two hours. The fourth one uses an area chart to show the same information, but on a six hours time window that slides every 15 minutes. The yellow area highlights the number of tweets that refers to MDW 2013. The fifth one displays the top-10 hashtags used in the micro posts. The content of this graph depends on the zoom level of the heatmap. When zooming on a specific district, e.g., Brera Design District, it displays the top-10 hashtags used in that in the micro posts originating from that area. The last two graphs show the number of micro posts originating from the nine districts interested by Fuorisalone events using a bar char and a dot chart.

4 Evaluation

Twindex Fuorisalone, between April 8th and April 17th 2013, analysed 106,770 micro posts, and it was viewed by more than twelve thousands distinct users.

Before reporting on the ability of TF to answer the three questions proposed in Section 1, we point the reader to the area chart illustrated in Figure 2.(a). In this graph, it is straightforward to see that MDW 2013 is visible in the change of volume of micro posts. On April 8th, 2013 at 18.00 the number of tweets moves from 90/150 every 15 minutes to 180/210. For the entire duration of MDW 2013 the volume of tweets is larger than 100 tweets every 15 minutes, while normally is less than 100. During the week the number of tweets after midnight is much larger than in the normal days. The April 14th, 2013 at 20.00 MDW 2013 ends and the volume of tweets rapidly goes back under 100 tweets every 15 minutes. Moreover, the yellow area (the number of tweets that refers to MDW 2013) is more visible during the event than in the following days.

Our mashup is able to answer question Q.1: “What are the most attractive events?”. The heatmap allows to visually identify hotspots where the social stream activity concentrates. Figures 3.(b) and (c) show the *hot spots* in Brera design district during a night of MDW 2013 and in a night after MDW,

⁹ At the time we write this paper the mashup is still running at <http://twindex.fuorisalone.it>. Video recording of what was visible during MDW 2013 are available at <http://www.streamreasoning.org/demos/mdw2013>.



Fig. 2. The mashup published on the official Fuorisalone website by Studiolabo

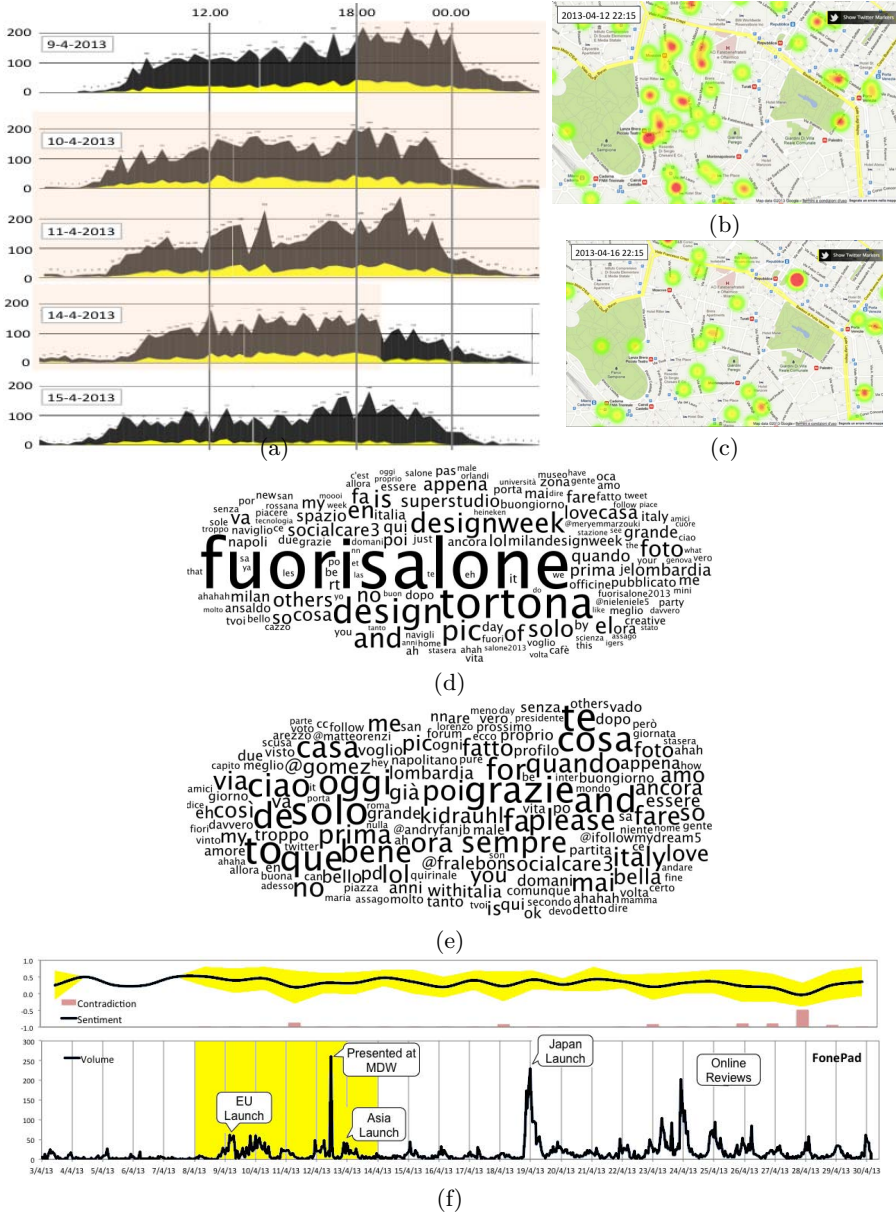


Fig. 3. A visual presentation of the results obtained with Twindex Fuorisalone: MDW 2013 is visible in the volume of micro posts (a); the hot spots Twindex Fuorisalone’s heatmap are all in proximity of Fuorisalone venues (b) and are different from those observable in a night after MDW 2013 (c); what the visitor say (d) around the venues, from which most of the geo-tagged micro posts originate, during Fuorisalone is different from what Milano’s citizens normally write in their micro posts (e); and changes were observed in public sentiment before, during and after the launch of an ASUS product during one of Fuorisalone events (f).

respectively. In a normal night, as the one in (c), few geo-tagged tweets are posted from Brera, whereas during MDW a number of hot points are visible on the map. The two most popular venues were Cesati antiques & works of art and Porta nuova 46/b; 16,653 and 13,416 tweets were, respectively, posted in their proximity. Thousands of tweets were posted in the proximity of a group of 6 other venues. Hundreds of tweets were posted around another group of 10 venues. Tens of tweets were posted on 62 venues. Around the remaining 81 only few tweets were posted.

The second question TF aims at answering is: “hat do visitors say about the events they join?”. TF displays in real time the top-10 hash tags appearing in the micro posts using a bar chart as the one illustrated in Figure 3. Normally the geo-tagged micro posts related to Milano cover a variety of topics (see Figure 3.(d)), whereas during MDW 2013 the most frequently used hashtags were all related to the ongoing event. Figure 3.(e) shows a tag cloud obtained using the words¹⁰ that appears in the 16,653 micro posts TF linked to the events hosted in Cesati antiques & works of art in Brera design district.

The number of geo-located micro posts per event was not enough to answer in real-time question Q.3: “What is the mood of the visitors before, during and after the event they join?”. However, as explained in Section 2.1, TF also listens to a topic-based stream related to MDW 2013. Between April 3rd and April 30th, TF analysed 107,044,487 micro posts and put us in the condition to answer a variant of Q.3: “What is the mood of the micro posts before, during and after an event where a new product is launched?”

Figure 3.(f) illustrates the results we obtained analysing the tweets related to *FonePad* – a product ASUS launched during MDW 2013 in one of the events in Brera Design District. Before MDW 2013, few micro posts every 15 minutes talk about FonePad; the peaks reach at most 25 micro posts every 15 minutes. During MDW 2013, ASUS announced the pre-sales in EU and presented the product. Those two facts are visible in the volume of micro posts: both the average number of micro posts and the height of the peaks are an order of magnitude larger than before MDW. In particular, the presentation¹¹ on April 12th is associable to the peak of 250 micro posts in 15 minutes. Similar peaks are visible also for the Japan launch¹² and when the first online reviews¹³ were published.

The sentiment expressed in the micro posts about FonePad was mostly positive. During this period some tweets about FonePad contained sentences like “*wanna buy it so bad!*”, which were classified as negative sentiments, but in reality were expressing positive sentiment. The contradiction level during such periods was also high due to a concern expressed by some users. For instance,

¹⁰ As often done in Natural Language Processing, we filtered out typical stop words in Italian and in English. Additionally, we also excluded Milano, which is the most commonly used word in micro posts originating from Milano.

¹¹ See <http://www.youtube.com/watch?v=vhyktTroDTw>

¹² See <http://www.asus.co.jp/News/JWtqmBbuQsxEkukt/>

¹³ See <http://www.expertreviews.co.uk/laptops/1299202/asus-FonePad> and http://reviews.cnet.com/tablets/asus-FonePad/4505-3126_7-35619221.html

reviews of FonePad, although very positive, caused a lot of discussions in the social media, where mixtures of positive and negative sentiments were mentioned, resulting in more contradicting distributions.

5 Conclusions

In this paper, we presented Twindex Fuorisalone, a mash-up that makes sense of social streams obtained from Twitter and Instagram using fuorisalone.it repository as a source of information about the events and the venues of Fuorisalone.

To cope with the streaming nature of micro posts, TF uses RDF streams and C-SPARQL within the Streaming Linked Data framework. The 106,770 tweets received during MDW 2013 as well as the 1,136,052 invocations to the Streaming Linked Data publishers were processed in main memory on a €25 per month cloud share using at most 2 CPU and 2 GB of RAM.

TF appears to be an effective solution to socially listen to Fuorisalone. TF found the events that attract the most of the visitors and observed what visitors say about them. The geo-bounded social stream lacked the volume to detect sentiment patterns, but the topic-based stream was rich enough to allow for observing changes in public sentiment before, during and after some events.

We are currently evolving TF in a generic solution to listen to city-scale events by blending social stream with mobile telecom data (e.g, number of phone calls, text messages and data connects originating from a given part of the city).

References

1. Calabrese, F., Colonna, M., Lovisolo, P., Parata, D., Ratti, C.: Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems* 12(1), 141–151 (2011)
2. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: *WWW*, pp. 851–860 (2010)
3. Balduini, et al.: BOTTARI: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *J. Web Sem.* 16, 33–41 (2012)
4. Balduini, M., Della Valle, E.: Tracking Movements and Attention of Crowds in Real Time Analysing Social Streams – The case of the Open Ceremony of London 2012. In: *Semantic Web Challenge at ISWC 2012* (2012)
5. Tsytsarau, M., Palpanas, T., Denecke, K.: Scalable Detection of Sentiment-Based Contradictions. In: *DiversiWeb workshop, WWW, Hyberabad, India* (2011)
6. Pang, B., Lee, L.: Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1-2), 1–135 (2007)
7. Breslin, J.G., Harth, A., Bojars, U., Decker, S.: Towards semantically-interlinked online communities. In: *Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 500–514. Springer, Heidelberg* (2005)
8. Della Valle, E., Ceri, S., Barbieri, D.F., Braga, D., Campi, A.: A first step towards stream reasoning. In: *Domingue, J., Fensel, D., Traverso, P. (eds.) FIS 2008. LNCS, vol. 5468, pp. 72–81. Springer, Heidelberg* (2009)

9. Calbimonte, J.-P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 96–111. Springer, Heidelberg (2010)
10. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
11. Barbieri et al.: C-SPARQL: A Continuous Query Language for RDF Data Streams. *Int. J. Semantic Computing* 4(1), 3–25 (2010)
12. Barbieri, D.F., Della Valle, E.: A proposal for publishing data streams as linked data - a position paper. In: LDOW (2010)
13. Phuoc, D.L., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. *J. Web Sem.* 16, 42–51 (2012)
14. Gray, A.J.G., et al.: A semantically enabled service architecture for mashups over streaming and stored data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 300–314. Springer, Heidelberg (2011)

DataConf: A Full Client-Side Web Mashup for Scientific Conferences

Lionel Médini^{1,2}, Florian Bacle², Benoît Durant de la Pastellière²,
Fiona Le Peutrec², and Nicolas Armando²

¹ LIRIS lab, University of Lyon, Lyon, France

² Université Claude Bernard Lyon, University of Lyon, Lyon, France

lionel.medini@liris.cnrs.fr,
{florian.bacle,benoit.durant-de-la-pastelliere,fiona.le-peutrec,
nicolas.armando}@etu.univ-lyon.fr

Abstract. This paper describes DataConf, a mobile Web mashup application that mixes Linked Data and Web APIs to provide access to different kinds of data. It relies on a widely used JavaScript framework and on a component-based approach to manage different datasources. It only requires static server-side contents and performs all processing on the client side.

DataConf aggregates conference metadata. It allows browsing conference publications, publication authors, authors' organizations, but also authors' other publications, publications related to the same keywords, conference schedule or resources related to the conference publications. For this, it queries the SPARQL endpoint that serves the conference dataset, as well as other open or custom endpoints and Web APIs that enrich these data.

Keywords: mobile Web, client-side mashup, Linked Data, Web API, SPARQL, SWC, SWDF, Backbone.js, publication browsing, conference events.

1 Introduction

The WWW'2012 conference that was held in Lyon was an occasion for the local Web community to initiate several innovating projects around conference material and Web technologies. We designed a mobile Web application to augment the poster track. It targeted conference attendees and allowed them to navigate among poster metadata (title, authors, abstract, keywords...), as well as authors and other publications metadata, using their smartphones and tablets. For this, it takes advantage of Linked Data [1] for enriching conference metadata from several sources. It dynamically constructs complex SPARQL queries, sends them to cross-domain endpoints and allows users to browse these metadata using either a textual or a graphical interface. It relies on a main conference publication dataset. Since 2006, such datasets are available on the SWDF¹ SPARQL

¹ <http://data.semanticweb.org/>

endpoint. These datasets mainly rely on the SWC vocabulary [2], [3], thus integrating FOAF, SWRC or iCal [4] elements.

After the conference, we refactored our application to be reusable and easily deployed for any conference that has its publication metadata available on the Web. This new version is named DataConf and is designed to be configurable and extendable. Several instances are already deployed for different conferences, as shown in Fig. 1.

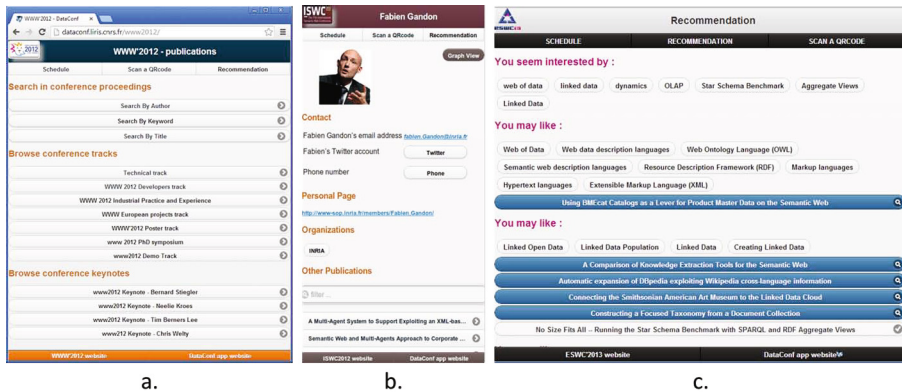


Fig. 1. Three different types of views in three different DataConf instances, respectively related to the WWC'2012, ISWC'2012 and ESWC'2013 conferences

We oriented the refactoring process around the two following tasks: conference chairs writing a configuration file for a new conference event and Web developers creating a new component that handles a particular datasource. We herein detail the architectural choices and structure, so that our application can be adopted by the community. The application homepage at <http://dataconf.liris.cnrs.fr/> provides access to different DataConf instances. More information on DataConf and its custom datasources are available on our wiki at <http://liris.cnrs.fr/dataconf/>. DataConf sources are downloadable at <https://github.com/ucbl/DataConf>.

This paper is organized as follows: we firstly present the architecture of our DataConf application. Then, we detail the different datasources that can be integrated with DataConf. We conclude and present evolution perspectives of this application.

2 DataConf Architecture

DataConf is an application that aggregates heterogeneous data about scientific conference materials, such as publications, authors and keywords. For this, it queries several datasources that can either be SPARQL endpoints (e.g. SWDF, DBLP²), or other Web services that provide information about these materials (e.g. conference schedule, external resources available on the Web, etc.).

² <http://dblp.l3s.de/d2r/>

In this section, after a short overview of mashup creation frameworks, we present the DataConf architecture. In order to facilitate its reuse for other conference, we then explain how to configure a DataConf instance and how to develop a component to connect it to a new datasource.

2.1 Mashup Framework Choice

We refactored our application to make it configurable, reusable and extendable. In order to facilitate its appropriation by the community, we chose to build it on top of an existing framework. For that matter, we chose an open source and widely used framework. This section describes the rationale of this choice.

Several frameworks exist in the field of enterprise mashup technologies, such as JackBe³ or Convertigo⁴. However, enterprise mashups [5] can be powerful applications that require users to log in and involve complex business workflows. On the contrary, we wanted our app to remain lightweight, since it only relies on components that query datasources, process their results and integrate them into the interface.

As our application originally targeted a WWW series conference, technical choices were oriented toward a full Web application applying recent/emerging Web technologies and standards. In particular, HTML5 APIs encouraged us designing the application as much client-sided as possible. We therefore chose to use a Web client-side (i.e. JavaScript) framework. Numerous JS frameworks now exist and several comparisons such as [6] are available, based on various criteria. As we aimed at favoring the adoption of our app by Web developers, we focused on community size, documentation and syntax simplicity. We thus chose to use Backbone.js⁵. On top of this framework, we developed the necessary objects to build our mashup application, among which a templating system and a workflow engine that can process queries to different datasources for each route (hash path) defined in a configuration file.

The Backbone.js framework provides a powerful routing system that permits the construction of new routes on the fly. The URL is the key of the routing system: by listening on the URL change event, it is able to match it to a route and to trigger the associated callback function. The router comes with a ready to go history handler which permits an automatic support of page navigation.

2.2 Architecture Overview

The core of the DataConf engine is an on-purpose built Backbone.js router. At runtime (when a new hash is selected), it plays the role of a controller and performs all the machinery needed to request the datasources, according to the commands specified in the configuration file (see next section). Fig. 2. shows the main objects involved in our architecture.

³ <http://www.jackbe.com/>

⁴ <http://www.convertigo.com/>

⁵ <http://backbonejs.org/>

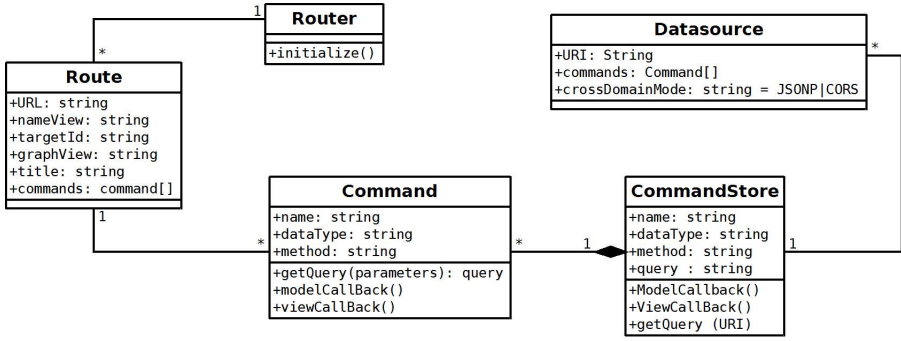


Fig. 2. Global view of the DataConf layer (on top of Backbone.js)

A command is a particular query type that can be sent to a datasource. For instance, in SWC, there will be a particular command to retrieve the publication corresponding to an URI. Commands contain the following fields: **name**, HTTP request **method** and returned **datatype**, as well as three methods:

- **getQuery**: returns the query object to send to the datasources
- **modelCallback**: handles the response from the datasource, performs calculations over the response data and transforms it in an internal common representation formatted
- **viewCallback**: is triggered by the router to integrate these formatted data into the views.

In order to cope with datasource heterogeneity problems, DataConf relies on an internal data representation that can differ from the data format contained in the response. For this, the model callback function that handles a response is limited to transforming the data into the appropriate common representation format (depending on its nature: title, author name, homepage...) and storing them using the LocalStorage API. Moreover, this architecture allows caching data in the LocalStorage object and more complex data composition into the views, since the view callback functions can wait for several types of data to be ready before starting the rendering process.

The datasource querying and integration process in DataConf is structured as follows. For each command associated to the current route, the router: i) gets the corresponding datasource and command objects, ii) verifies the data availability in the LocalStorage, iii) if not, constructs an asynchronous cross-domain request using the datasource and command configuration values, iv) sends the request, v) triggers the command model callback when the response is available, vi) triggers the command ViewCallback to render the view when the model callback has returned. This process is illustrated in Fig. 3.

2.3 Configuring a DataConf Instance

To be deployed for a particular conference, DataConf relies on a static configuration file. No sensitive information is present in the configuration file and this

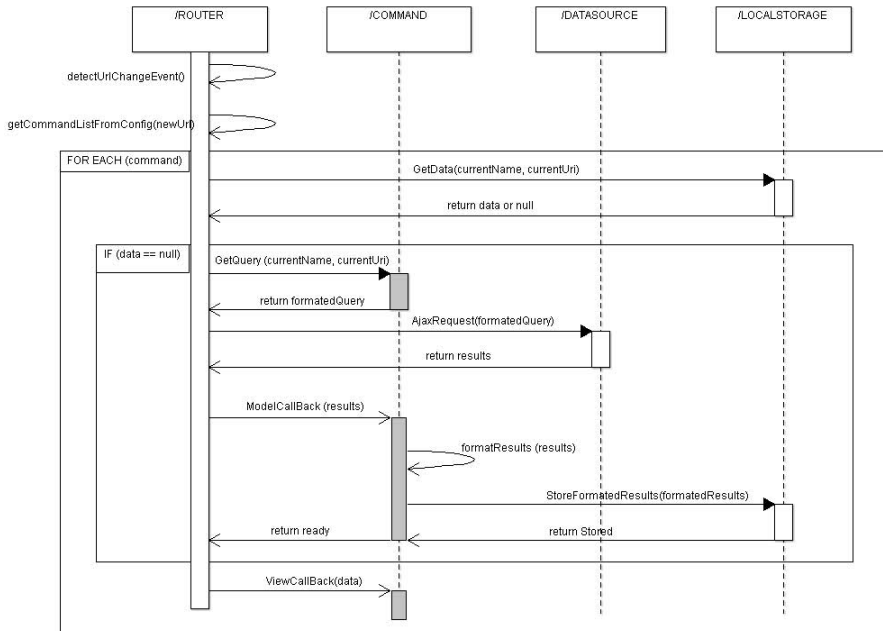


Fig. 3. Detailed view of the data retrieving process

file can be processed on the client side. This configuration file is written in JSON and contains the following elements:

- **Conference general information**, such as its name, base URI and logo URI.
- **View templates** that allow presenting different entities (publications, authors...) using different layouts. These templates are described by their locations and names. During initialization, all view templates are loaded and precompiled to create a single-page application that will be able to handle its presentation layer without calling the server again.
- The set of **datasources** that can be queried by the application. A data-source description contains a server URI, a cross-domain mode (JSONP / CORS)⁶ and references to the component that handles this datasource. This component is a JS file defining a set of commands and is called a *Command-Store*.
- **Routes** that link the URI hash paths to datasource commands.

⁶ JSONP (JSON with Padding, <http://bob.ippoli.to/archives/2005/12/05/remote-json-jsonp/>) and CORS (Cross-Origin Resource Sharing, <http://www.w3.org/TR/cors/>) are two techniques for sending cross-domain requests from a Web browser. The former exploits the same-origin policy exception for the HTML 'script' tag, whereas the latter uses a set of on-purpose HTTP headers issued in a recent W3C specification (currently: candidate recommendation).

2.4 Developing a Datasource Component

Developers who wish to add a new datasource to their DataConf instance are welcome to do so. For this, they need to gather in the same JS file (CommandStore), all the command objects associated to requests that can be sent to this datasource. Once this done, they need to declare this datasource and link it to their CommandStore in the configuration file and associate the commands with the appropriate routes.

3 DataConf Ecosystem

At the ESWC'2013 conference, we demonstrated the integration of our DataConf application with the following datasources:

- **Main datasource:** SPARQL endpoint serving the conference metadata in the Semantic Web Conference (SWC) ontology format (see section 1). It provides all the data about publications and authors, and refers to them using their URIs in the dataset.
- **DBLP:** SPARQL endpoint used to access authors' other publications in the DBLP database. By clicking on a DBLP publication, users can browse the DBLP datasource using DataConf as they browse the conference ones. In the ESWC'2013 instance, DataConf uses the RKBExplorer⁷ SPARQL endpoint to access the DBLP datasource. This endpoint is queried using the author's full name, which can cause homonymy issues. An example of DBLP integration in DataConf is shown in the "Other Publications" section of Fig. 1.b.
- **DuckDuckGo!**⁸: Search engine used to enrich organizations' information, such as homepage URL and logo. This engine is queried using the organization full name. We rely on its "I'm feeling ducky!" feature to retrieve the most probable query result. We noticed that it provides quite good results for organizations, but less good results for people. Indeed, we originally used DuckDuckGo! for retrieving authors' homepages, but switched to Google while finding too many inaccurate results on this particular query type.
- **Google Web search API**⁹: used to enrich data by finding authors' homepages, as explained previously. Homepages are displayed in the "Personal Page" section of the authors' views, see Fig. 1.b.
- **SimpleSchedule:** Custom datasource that allows retrieving the actual schedule of the conference events. Experience shows that the schedule is a dynamic artifact, whereas a conference dataset is not. In order to provide an accurate version of the time schedule, we developed a web application that gathers all the events from the conference dataset and proposes a back office graphical user interface for easily creating, rescheduling and deleting these events. DataConf users can thus view the time and location of a paper

⁷ <http://dblp.rkbexplorer.com/> -- <http://www.informatik.uni-trier.de/~ley/db/>

⁸ http://duckduckgo.com/1/c/RDF_data_access

⁹ <https://developers.google.com/web-search/>

presentation, even if it has been rescheduled by the conference organizers. SimpleSchedule can be part of the conference management process and exposes the conference schedule as a RESTful Web service. SimpleSchedule exports event data in ICS¹⁰. An example of SimpleSchedule integration in DataConf can be seen in the DataConf homepage (Fig. 1.a), where all the different types of events (tracks, keynotes...) are displayed.

- **DataPaper:** Custom datasource that allows conference actors (authors, chairs) to provide external resources about their own descriptions and publications. For this, DataPaper stores information about the URIs, description texts and format of these external resources in a NoSQL database (CouchDB) and can be queried by DataConf using the database Web API. These data are stored in key-value pairs, using for instance a publication URI in the conference dataset as a key and the URI of a web page that gives more details about this publication as a value. We also developed a back-office interface for enriching the DataPaper database, in which conference actors (identified by their foaf:profile and role in the dataset) can add external resources linked to their profile or publications. DataConf then queries this service using the URI of an entity present in the conference dataset, in order to retrieve all the external resources that DataPaper stores about this entity. Even if the backoffice interface is freely available as an unofficial WordPress plugin, we suggest to keep using our DataPaper instance for later conferences, in order to keep the benefits of the contents added by former conference actors about their profiles. Fig. 1.b. shows how DataPaper contents, such as photo and contact information, can be integrated to enrich an author’s view, using the data he/she already entered in DataPaper.
- **Reasoner:** Custom “local datasource” that performs client-side reasoning on the conference publication keywords. It is based on a keyword ontology that is loaded at initialization time. During user navigation, DataConf stores the keywords of the publications viewed by the user. The reasoner datasource allows retrieving super and sub-keywords (and thus the publications they refer to), and recommending to users publications they did not yet see and referring to keywords close to those they have already explored. This is performed using an adapted version of the OWLReasoner JS engine, that is embedded in a Web worker and queried in SPARQL (with limitations¹¹), la JSONP. Thus, its data can be processed and integrated in the views in the same manner as if it was a distant endpoint. Fig. 1.c shows an example of recommendation interface in the DataConf ESWC’2013 instance.

4 Conclusions

DataConf is a mobile Web mashup designed w.r.t. recent Web standards and technologies and so that all computation is done on client side. It proposes

¹⁰ <http://www.ietf.org/rfc/rfc2445.txt>

¹¹ Currently, our version of the OWLReasoner engine can process subClassOf queries in SPARQL, but with the constraint of having only one triple per query. In addition, OWLReasoner does not support literals and therefore, data properties.

various features, like querying and browsing conference publications and allowing users to access and enrich the metadata from two different SPARQL endpoints, four Web services and a local inference engine. Conference chairs can easily configure a DataConf instance as soon as the conference dataset is available on a SPARQL endpoint. We also designed this application to be reusable and extendable by Web developers: it uses the Backbone.js framework and relies on view templates and simple components that encapsulate datasource processing logics. Several DataConf instances are currently available and query datasets of different conferences, including ESWC'2013.

In the future, we hope that this application will gain audience in the Web and Semantic Web communities, so that it can provide even more other useful services for scientific conference attendees. The next envisaged version should allow users to authenticate, in order to gain service personalization and social features. Moreover, as templates and datasources can be changed and specified in a configuration file, we plan to explore the reusability of our application infrastructure for another application field than a conference, in order to evaluate the interest of this architecture as a lightweight framework for mobile Web mashups.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)* 5(3), 1–22 (2009)
2. Müller, K., Heath, T., Handschuh, S., Domingue, J.: Recipes for SemanticWeb dog food - The ESWC and ISWC metadata projects. In: 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007+ASWC 2007), Busan, South Korea, November 11-15 (2007)
3. Müller, K., Bechhofer, S., Heath, T.: *Semantic Web Conference Ontology* (2009), <http://data.semanticweb.org/ns/swc/ontology>
4. *Internet Calendaring and Scheduling Core Object Specification (iCalendar) - RFC 2445 - IETF* (November 1998), <http://www.faqs.org/rfcs/rfc2445.html>
5. Hinchcliffe, D., Benson, J.: *EMML changes everything: Profitability, Predictability & Performance through Enterprise Mashups*. Hinchcliffe & Company 1940 Duke Street, Alexandria, Virginia 22314 1-877-HIN-CHCL (877.446.2425) (2009), http://openmashup.org/whitepaper/docs/oma_whitepaper_120309.pdf
6. Hempton, G.L.: *The Top 10 Javascript MVC Frameworks Reviewed* (2012), <http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/>

Author Index

- Abdelrahman, Ahmed 302
Alexopoulos, Panos 271
Ali, Muhammad Intizar 219
Allik, Alo 178, 285
Alonen, Miika 204
Amini, Massih-reza 310
Aranda, Carlos Buil 189
Armando, Nicolas 337
Aroyo, Lora 199
Ashkpour, Ashkan 306
Assaf, Ahmad 319
Aufaure, Marie-Aude 271
Augenstein, Isabelle 101
- Babbar, Rohit 310
Bacle, Florian 337
Baget, Jean-François 157
Balduini, Marco 327
Basca, Cosmin 282
Bayerl, Sebastian 268
Beck, Nicolas 225
Ben-Mustapha, Nesrine 271
Bernstein, Abraham 282
Bilidas, Dimitris 125
Bosca, Alessio 162
Bouchard, Hugues 271
Buitelaar, Paul 242
- Cabrio, Elena 194
Calvanese, D. 293
Carvalho, Danilo S. 236
Casanova, Marco A. 248
Chen, Xi 279
Chenu-Abente, Ronald 298
Ciancarini, Paolo 253
Ciravegna, Fabio 101
Coetzer, Willem 87
Cojan, Julien 194
Confalonieri, Cristian 327
Copeland, Maria 113
Corcho, Oscar 189
Coughlan, Barry 242
Croitoru, Madalina 157
Cross, Valerie 279
Curry, Edward 214
- Darari, Fariz 275
da Silva, Bruno Paiva Lima 157
da Silva, João C.P. 236
de Boer, Victor 199
Declerk, Thierry 242
de la Pastellière, Benoit Durant 337
Dell'Aglio, Daniele 327
Della Valle, Emanuele 327
Di Iorio, Angelo 66, 231
Dixon, Simon 178, 285
Dogani, Kallirroï 209
Dragoni, Mauro 162
Draicchio, Francesco 253, 263
- Fazekas, György 178, 285
Fernández, Miriam 141
Ferré, Sébastien 273
Freitas, André 214, 236
Frosterus, Matias 289, 296
- Gandon, Fabien 184, 194, 287
Gangemi, Aldo 253, 263
García, José Luis Redondo 258
Gaussier, Eric 310
Gentile, Anna Lisa 101
Gerber, Aurora 87
Ghidini, Chiara 162
Giese, Martin 125, 293
Gillani, Syed Zeeshan Haider 219
Giunchiglia, Fausto 291, 298
Gomez-Pérez, José Manuel 271
Gonçalves, Rafael S. 113
Gotttron, Thomas 142, 225
Granitzer, Michael 173, 268
Guéret, Christophe 306
Guerrini, Giovanna 34
- Haase, Peter 125, 293
Hallili, Amine 194
Helmich, Jiří 147
Henderson, James 271
Hoefer, Patrick 173
Hoekstra, Rinke 78, 306
Höffner, Konrad 167
Hogan, Aidan 302

- Horrocks, Ian 125, 293
 Hu, Beibei 271
 Hubauer, T. 293
 Hume, Alethia 291
 Hyvönen, Eero 204, 289, 296, 300

 Ioannidis, Y. 293
 Ivanova, Valentina 152

 Jiménez-Ruiz, Ernesto 125, 293

 Käfer, Tobias 302
 Kauppinen, Tomi 204
 Kharlamov, Evgeny 125, 293
 Klímek, Jakub 147
 Kllapi, Herald 125, 293
 Klüwer, J. 293
 Koho, Mikko 300
 Kosch, Harald 268
 Koubarakis, Manolis 125, 209, 293
 Krayner, Bastian 142
 Krieger, Uli 242
 Kyzirakos, Kostis 209

 Lambrix, Patrick 152
 Lamparter, S. 293
 Lang, Joel 271
 Laurene, Nina 300
 Legrand, Damien 184
 Lehmann, Jens 167
 Le Peutrec, Fiona 337
 Lindstaedt, Stefanie 173
 López, Vanessa 141
 Lorey, Johannes 46
 Lyko, Klaus 167

 Marie, Nicolas 184, 287
 Marie-Thomas, Susan 242
 Markou, George 304
 Marx, Maarten 5
 McGuinness, Deborah L. 308
 Médini, Lionel 337
 Meroño-Peñuela, Albert 306
 Metzigg, Cornelia 310
 Mika, Peter 271
 Mileo, Alessandra 219
 Milicic, Vuk 258
 Möller, R. 293
 Monteiro, José Maria 248

 Moodley, Deshendran 87
 Musetti, Alberto 253

 Naumann, Felix 46
 Nebhi, Kamel 312
 Nečaský, Martin 147
 Nerima, Luka 312
 Neto, Luís Eufrazio T. 248
 Neuenstadt, C. 293
 Ngomo, Axel-Cyrille Ngonga 167
 Nikolaou, Charalampos 209
 Nordtveit, T. 293
 Norton, Barry 101
 Nuzzolese, Andrea Giovanni 66, 231,
 253, 263

 O'Byrne, Patrick 302
 Ockeloen, Niels 199
 O'Riain, Seán 214, 242
 Özcep, Ö. 293
 Özçep, Özgür 125

 Palpanas, Themis 327
 Parsia, Bijan 113
 Partalas, Ioannis 310
 Paulheim, Heiko 22
 Peroni, Silvio 66, 231
 Pessala, Sini 296
 Peters, Arne 142
 Presutti, Valentina 253, 263
 Priyatna, Freddy 189

 Refanidis, Ioannis 304
 Ribière, Myriam 184, 287
 Rietveld, Laurens 78
 Ristoski, Petar 22
 Rizzo, Giuseppe 258
 Rodriguez-Muro, M. 293
 Rodríguez-Muro, Mariano 125
 Rosati, Riccardo 125
 Roshchin, M. 293
 Rowe, Matthew 277

 Sabol, Vedran 173
 Sandler, Mark 178, 285
 Sattler, Uli 113
 Savo, F. 293
 Scharnhorst, Andrea 306
 Scheglmann, Stefan 225
 Scherp, Ansgar 142

- Schlatte, Rudolf 125
Schlegel, Kai 268
Schlobach, Stefan 1
Schmidt, Michael 125, 293
Schulz, Axel 22
Seifert, Christin 268
Senart, Aline 319
Seppälä, Katri 296
Shangguan, Zhenning 308
Silwal, Pramit 279
Solimando, Alessandro 34
Soylu, Ahmet 125, 293
Speck, René 167
Stegmaier, Florian 268
Stevens, Robert 113
Suominen, Osma 204

Tarasova, Tatiana 5
Troncy, Raphaël 258

Troncy, Raphael 319
Tsytsarau, Mikalai 327
Tuominen, Jouni 289, 296, 300

Umbrich, Jürgen 302

Vanrompay, Yves 271
Vidal, Vânia Maria P. 248
Völker, Johanna 1

Waalder, Arild 125, 293
Wagner, Claudia 277
Wahlroos, Mika 289
Wehrli, Eric 312

Zaihrayeu, Ilya 298
Zhang, Ziqi 101
Zheleznyakov, Dmitriy 125, 293
Zollinger, Mengia 282
Zwicklbauer, Stefan 268