

Marcelo Arenas · Oscar Corcho
Elena Simperl · Markus Strohmaier
Mathieu d'Aquin · Kavitha Srinivas
Paul Groth · Michel Dumontier · Jeff Heflin
Krishnaprasad Thirunarayan · Steffen Staab (Eds.)

LNCSE 9367

The Semantic Web – ISWC 2015

14th International Semantic Web Conference
Bethlehem, PA, USA, October 11–15, 2015
Proceedings, Part II

2
Part II

ISWC2015
Bethlehem, Pennsylvania



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zürich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7409>

Marcelo Arenas · Oscar Corcho
Elena Simperl · Markus Strohmaier
Mathieu d'Aquin · Kavitha Srinivas
Paul Groth · Michel Dumontier
Jeff Heflin · Krishnaprasad Thirunarayan
Steffen Staab (Eds.)

The Semantic Web – ISWC 2015

14th International Semantic Web Conference
Bethlehem, PA, USA, October 11–15, 2015
Proceedings, Part II

Editors

Marcelo Arenas
Pontificia Universidad Católica de Chile
Santiago de Chile
Chile

Oscar Corcho
Universidad Politecnica de Madrid
Boadilla del Monte
Spain

Elena Simperl
University of Southampton
Southampton
UK

Markus Strohmaier
Department of Computational Social Science
Gesis Leibniz-Institut
Köln, Nordrhein-Westfalen
Germany

Mathieu d'Aquin
The Open University
Milton Keynes
UK

Kavitha Srinivas
IBM Research
Yorktown Heights, NY
USA

Paul Groth
Elsevier Labs.
Amsterdam
The Netherlands

Michel Dumontier
School of Medicine
Stanford University
Stanford, CA
USA

Jeff Hefflin
Lehigh University
Bethlehem, PA
USA

Krishnaprasad Thirunarayan
Wright State University
Dayton, OH
USA

Steffen Staab
University of Koblenz-Landau
Koblenz, Rheinland-Pfalz
Germany

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-319-25009-0

ISBN 978-3-319-25010-6 (eBook)

DOI 10.1007/978-3-319-25010-6

Library of Congress Control Number: 2015950869

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

Springer Cham Heidelberg New York Dordrecht London

© Springer International Publishing Switzerland 2015

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.

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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

Since its inception in the last decade, the Semantic Web has experienced a steady and continuous development toward its original vision, both in terms of research and of technology development and applications. Many of the research results presented in the initial conferences have now matured and been taken up in commercial settings, giving rise to new research problems that are now being explored. Large-scale initiatives, such as some of the most popular datasets in the linked open data cloud, are now considered as commodities and are part of many services that are being used on a daily basis, not only inside our research community but also in other research areas.

The International Semantic Web Conference (ISWC) has continued to be the premier venue for presenting innovative systems, applications, and research results related to the Semantic Web. In this edition, we aimed at making it even more clear that the Semantic Web is not only about using the well-known W3C recommendations RDF, RDF Schema, and/or OWL, and dealing with their associated challenges, but generally about the combination of semantics, data, and the Web.

This volume contains the proceedings of ISWC 2015, with papers accepted into the various tracks for which specific calls for papers had been issued. Besides the usual research track, this year we split the Replication, Benchmark, Data and Software track and the Semantic Web In-Use track from previous editions into three more specialized tracks, covering: Empirical Studies and Experiments, In-Use and Software, and Data Sets and Ontologies.

We received a very good response to all our calls from a truly international community of researchers and practitioners. The statistics on the submissions and accepted papers were:

- Research track: 172 papers submitted, with 38 of them accepted
- Empirical Studies and Experiments track: 23 papers submitted, with seven of them accepted
- In-Use and Software track: 33 papers submitted, with 14 of them accepted
- Data Sets and Ontologies track: 35 papers submitted, with eight of them accepted

All submitted papers were reviewed by at least three Program Committee (PC) members. In the case of the research track, the review process for each paper was also overviewed by a senior PC member, whose job was to drive discussions among reviewers when their points of view diverged, to make sure that clear questions were sent to the authors so as to give them the opportunity to reply to reviewers during the rebuttal period, and to provide a final meta-review with a summary of the strongest and weakest aspects of each of the papers. Finally, the acceptance and rejection of papers were decided via phone conferences between PC chairs and senior PC members that lasted two consecutive days.

This year's edition also had two additional innovations. On the one hand, we encouraged authors to include pointers to any additional material that supports the

scientific claims made in their papers (e.g., extended technical reports, source code, datasets, links to applications). This proposal was received well among authors, who made an extra effort to make such additional material available for reviewers first, and if their paper was accepted, to make it available together with their camera-ready version of the paper. Such additional material has been uploaded into a variety of systems, including figshare, zenodo and institutional repositories of universities and research centers.

The second request introduced by PC chairs was the suggestion to reviewers to sign their reviews if they wished, following recent trends on open reviewing, so as to pave the way for having a more transparent review process for our conference. The number of signed reviews was still very low, which suggests that there is a need to continue discussions on whether this open review model is applicable for a conference like ISWC or should be left to journals, which have a longer review process.

ISWC 2015 also included a Doctoral Consortium track for PhD students from the Semantic Web community, giving them the opportunity not only to present their work but also to discuss in detail their research topics and plans and to receive extensive feedback from leading scientists in the field. The Doctoral Consortium was very efficiently run by Fabio Ciravegna and María-Esther Vidal.

Another unique aspect of the International Semantic Web Conferences is the Semantic Web Challenge. In this competition, practitioners and scientists are encouraged to showcase useful and leading-edge applications of Semantic Web technology. This year the Semantic Web Challenge was organized by Sean Bechhofer and Kostis Kyzirakos. It consisted of two main tracks, the Open track, focused on end-user applications, and the Big Data track, which follows on the success of the Billion Triple Data track from previous editions.

The ISWC program was further enriched by keynote talks given by leading figures from both the academic and business world. Specifically, Michael Atkin, Andrew McCallum, and Ian Horrocks.

As in previous ISWC editions, the conference program also included an extensive Tutorial and Workshop Program, with eight tutorials and 24 workshops, which were co-ordinated by Miriam Fernández and Krzysztof Janowicz.

We would like to thank Jeff Z. Pan and Serena Villata for chairing an excellent Poster and Demo Session, and Vinay Chaudhri and Tony Shaw for co-ordinating the Industry Track, a forum for the latest discussions and demonstrations of semantic applications in the commercial world. The Industry Track serves as a complement to the In-Use and Software Track and shows just how far semantics are expanding through the enterprise.

The conference also included a Lightning Talk session, where ISWC attendees could at very short notice get five minutes of attention from the audience, to report on anything they have done, plan to do, like or dislike about the Semantic Web.

We are also much indebted to Krishnaprasad Thirunarayan, our proceedings chair, who provided invaluable support in compiling the printed proceedings and exhibited super-human patience in allowing the other chairs to stretch deadlines to the absolute limit. Many thanks also to Matthew Horridge and Nadeschda Nikitina, our student coordinators, and to Juan Sequeda, our publicity chair.

As has been the case for the past few years, ISWC 2015 also contributed to the linked data cloud, by providing semantically characterized data on aspects of the conference. This would not have been possible without the efforts of our metadata chair, Heiko Paulheim.

We would like to give a special thank you to the local organization chair, Jeff Heflin and his team, who did a brilliant job in taking care of the local arrangements and ensuring that anything the Organizing Committee needed was promptly made available. We would also like to thank the generous contribution from our sponsors and the fine work of the sponsorship chairs, Michelle Cheatham and Carlos Pedrinaci. Finally, we are indebted to Andrei Voronkov and his team for providing the sophisticated and convenient service of EasyChair and to Alfred Hofmann, Anna Kramer, and their team at Springer for being most helpful with publishing the proceedings.

October 2015

Marcelo Arenas
Oscar Corcho
Elena Simperl
Markus Strohmaier
Mathieu d'Aquin
Kavitha Srinivas
Michel Dumontier
Paul Groth
Steffen Staab

Conference Organization

General Chair

Steffen Staab Universität Koblenz-Landau, Germany

Local Chair

Jeff Heflin Lehigh University, USA

Research Track Chairs

Marcelo Arenas Pontificia Universidad Católica de Chile, Chile
Oscar Corcho Universidad Politécnica de Madrid, Spain

Empirical Studies and Experiments Track Chairs

Elena Simperl University of Southampton, UK
Markus Strohmaier Universität Koblenz-Landau, Germany

In-Use and Software Track Chairs

Mathieu d'Aquin The Open University, UK
Kavitha Srinivas IBM T.J. Watson Research Center, USA

Data Sets and Ontologies Chairs

Michel Dumontier Stanford University, USA
Paul Groth Elsevier Labs, The Netherlands

Industry Track Chairs

Vinay Chaudhri SRI International, USA
Tony Shaw Dataversity, USA

Workshop and Tutorial Chairs

Miriam Fernandez The Open University, UK
Krzysztof Janowicz University of California, Santa Barbara, USA

Demos and Posters Chairs

Jeff Z. Pan University of Aberdeen, UK
Serena Villata Inria, France

Doctoral Consortium Chairs

Fabio Ciravegna University of Sheffield, UK
María-Esther Vidal Universidad Simón Bolívar, Venezuela

Proceedings Chair

Krishnaprasad Thirunarayan Wright State University, USA

Semantic Web Challenge Chairs

Sean Bechhofer University of Manchester, UK
Kostis Kyzirakos Centrum Wiskunde and Informatica, The Netherlands

Sponsorship Chairs

Michelle Cheatham Wright State University, USA
Carlos Pedrinaci The Open University, UK

Metadata Chair

Heiko Paulheim University of Mannheim, Germany

Publicity Chair

Juan Sequeda Capsenta Labs, USA

Assistant to General Chair

Ulrich Wechselberger Universität Koblenz-Landau, Germany

Student Coordinators

Matthew Horridge Stanford University, USA
Nadeschda Nikitina University of Oxford, UK

Senior Program Committee: Research Track

Harith Alani KMI, The Open University, UK
Lora Aroyo VU University, The Netherlands

| | |
|------------------------|--|
| Sören Auer | University of Bonn, Germany |
| Philipp Cimiano | Bielefeld University, Germany |
| Philippe Cudré-Mauroux | University of Fribourg, Switzerland |
| Birte Glimm | University of Ulm, Germany |
| Lalana Kagal | MIT, USA |
| Spyros Kotoulas | IBM, Ireland |
| David Martin | Nuance Communications, USA |
| Axel-Cyrille Ngonga | Universität Leipzig, Germany |
| Natasha Noy | Google, USA |
| Jeff Z. Pan | The University of Aberdeen, UK |
| Axel Polleres | Vienna University of Economics and Business, Austria |
| Marta Sabou | KMI, The Open University, UK |
| Uli Sattler | University of Manchester, UK |
| Juan F. Sequeda | Capsenta Labs, USA |
| Tania Tudorache | Stanford University, USA |
| Antoine Zimmermann | Ecole des Mines de Saint Etienne, France |

Program Committee: Research Track

| | |
|--------------------------|--|
| Faisal Alkhateeb | Yarmouk University, Jordan |
| Pramod Anantharam | Kno.e.sis Center, Wright State University, USA |
| Kemafor Anyanwu | North Carolina State University, USA |
| Medha Atre | University of Pennsylvania, USA |
| Isabelle Augenstein | The University of Sheffield, UK |
| Nathalie Aussenac-Gilles | IRIT CNRS, France |
| Payam Barnaghi | University of Surrey, UK |
| Andrew Bate | University of Oxford, UK |
| Christian Bizer | University of Mannheim, Germany |
| Roi Blanco | Yahoo! Research, Spain |
| Eva Blomqvist | Linköping University, Sweden |
| Kalina Bontcheva | University of Sheffield, UK |
| Paolo Bouquet | University of Trento, Italy |
| Loris Bozzato | Fondazione Bruno Kessler, Italy |
| Adrian M.P. Brasoveanu | MODUL University Vienna, Austria |
| Carlos Buil Aranda | Pontificia Universidad Católica de Chile, Chile |
| Paul Buitelaar | Insight - National University of Ireland, Galway, Ireland |
| Gregoire Burel | The Open University, UK |
| Jean-Paul Calbimonte | EPFL, Switzerland |
| Diego Calvanese | KRDB Research Centre, Free University of Bozen-Bolzano, Italy |
| Amparo E. Cano | Knowledge Media Institute, The Open University, UK |
| Iván Cantador | Universidad Autónoma de Madrid, Spain |
| Irene Celino | CEFRIEL, Italy |
| Pierre-Antoine Champin | LIRIS, France |
| Gong Cheng | Nanjing University, China |

| | |
|-------------------------|--|
| Key-Sun Choi | KAIST, Korea |
| Sam Coppens | IBM Research - Smarter Cities Technology Center (SCTC), Ireland |
| Isabel Cruz | University of Illinois at Chicago, USA |
| Bernardo Cuenca Grau | University of Oxford, UK |
| Claudia D'Amato | University of Bari, Italy |
| Daniele Dell'Aglio | Politecnico di Milano, Italy |
| Emanuele Della Valle | DEI, Politecnico di Milano, Italy |
| Stefan Dietze | L3S Research Center, Germany |
| John Domingue | Knowledge Media Institute, The Open University, UK |
| Jérôme Euzenat | Inria and University of Grenoble, France |
| Nicola Fanizzi | Università di Bari, Italy |
| Anna Fensel | Semantic Technology Institute (STI), University of Innsbruck, Austria |
| Miriam Fernandez | Knowledge Media Institute, The Open University, UK |
| Lorenz Fischer | University of Zurich, Switzerland |
| Achille Fokoue | IBM Research, USA |
| Enrico Franconi | Free University of Bozen-Bolzano, Italy |
| Fabien Gandon | Inria |
| Aldo Gangemi | Université Paris 13 and CNR-ISTC, France |
| Raúl García-Castro | Universidad Politécnica de Madrid, Spain |
| Daniel Garijo | Universidad Politécnica de Madrid, Spain |
| Anna Lisa Gentile | University of Sheffield, UK |
| Jose Manuel Gomez-Perez | Intelligent Software Components (isoco) S.A. |
| Thomas Gottron | University of Koblenz-Landau, Germany |
| Tudor Groza | The Garvan Institute of Medical Research, Australia |
| Michael Gruninger | University of Toronto, Canada |
| Giancarlo Guizzardi | Ontology and Conceptual Modeling Research Group (NEMO)/Federal University of Espirito Santo (UFES), Brazil |
| Kalpa Gunaratna | Kno.e.sis Center, Wright State University, USA |
| Claudio Gutierrez | Universidad de Chile, Chile |
| Christophe Guéret | Data Archiving and Networked Services (DANS), The Netherlands |
| Olaf Görlitz | Recommind, Germany |
| Peter Haase | fluid Operations, Germany |
| Armin Haller | Australian National University, Australia |
| Tom Heath | Open Data Institute, UK |
| Cory Henson | Bosch Research and Technology Center, USA |
| Martin Hepp | Bundeswehr University Munich, Germany |
| Pascal Hitzler | Wright State University, USA |
| Rinke Hoekstra | VU University Amsterdam, The Netherlands |
| Aidan Hogan | DCC, Universidad de Chile, Chile |
| Matthew Horridge | Stanford University, USA |
| Ian Horrocks | University of Oxford, UK |
| Andreas Hotho | University of Würzburg, Germany |

| | |
|-------------------------------|---|
| Geert-Jan Houben | TU Delft, The Netherlands |
| Julia Hoxha | Institute of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Germany |
| Bo Hu | Fujitsu Laboratories of Europe, UK |
| Wei Hu | Nanjing University, China |
| Eero Hyvönen | Aalto University, Finland |
| Ali Intizar | National University of Ireland, Ireland |
| Krzysztof Janowicz | University of California, Santa Barbara, USA |
| Mustafa Jarrar | Birzeit University, Palestine |
| Ernesto Jimenez-Ruiz | University of Oxford, UK |
| Hanmin Jung | KISTI, South Korea |
| Jaap Kamps | University of Amsterdam, The Netherlands |
| C. Maria Keet | University of Cape Town, South Africa |
| Matthias Klusch | DFKI, Germany |
| Jacek Kopecky | University of Portsmouth, UK |
| Manolis Koubarakis | National and Kapodistrian University of Athens, Greece |
| Markus Krötzsch | Technische Universität Dresden, Germany |
| Werner Kuhn | UCSB, USA |
| Agnieszka Lawrynowicz | Poznan University of Technology, Poland |
| Freddy Lecue | IBM Research, USA |
| Chengkai Li | University of Texas at Arlington, USA |
| Lei Li | Florida International University, USA |
| Nuno Lopes | IBM Research, Smarter Cities Technology Centre, Dublin, Ireland |
| Vanessa Lopez | IBM Research, Ireland |
| David Martin | Nuance Communications, USA |
| Diana Maynard | University of Sheffield, UK |
| Robert Meusel | University of Mannheim, Germany |
| Nandana Mihindukulasooriya | Universidad Politécnica de Madrid, Spain |
| Alessandra Mileo | National University of Ireland, Galway, INSIGHT Centre for Data Analytics, Ireland |
| Iris Miliaraki | Yahoo Labs, Spain |
| Riichiro Mizoguchi | Japan Advanced Institute of Science and Technology, Japan |
| Dunja Mladenic | Jozef Stefan Institute, Slovenia |
| Boris Motik | University of Oxford, UK |
| Enrico Motta | Knowledge Media Institute, The Open University, UK |
| Ekawit Nantajeewarawat | Sirindhorn International Institute of Technology, Thammasat University, Thailand |
| Nadeschda Nikitina | Oxford University, UK |
| Andriy Nikolov | fluid Operations AG, Germany |
| Lyndon Nixon | MODUL University, Austria |
| Jeff Z. Pan | The University of Aberdeen, UK |

| | |
|----------------------------|--|
| Bijan Parsia | University of Manchester, UK |
| Terry Payne | University of Liverpool, UK |
| Carlos Pedrinaci | The Open University, UK |
| Sujan Perera | Kno.e.sis Center, Wright State University, USA |
| Silvio Peroni | University of Bologna and ISTC-CNR, Italy |
| Robert Piro | University of Oxford, UK |
| Dimitris Plexousakis | Institute of Computer Science, FORTH, Greece |
| Valentina Presutti | STLab (ISTC-CNR), Italy |
| Freddy Priyatna | Universidad Politécnica de Madrid, Spain |
| Jorge Pérez | Universidad de Chile, Chile |
| Guilin Qi | Southeast University, China |
| Yves Raimond | BBC, UK |
| Ganesh Ramakrishnan | IIT Bombay, India |
| Maya Ramanath | IIT Delhi, India |
| Chantal Reynaud | LRI, Université Paris-Sud, France |
| Mariano Rico | Universidad Politécnica de Madrid, Spain |
| Giuseppe Rizzo | EURECOM, France |
| Marco Rospocher | Fondazione Bruno Kessler, Italy |
| Matthew Rowe | Lancaster University, UK |
| Sebastian Rudolph | Technische Universität Dresden, Germany |
| Harald Sack | Hasso Plattner Institute for IT Systems Engineering, University of Potsdam, Germany |
| Hassan Saif | The Open University - Knowledge Media Institute, UK |
| Francois Scharffe | LIRMM, University of Montpellier, France |
| Ansgar Scherp | Kiel University and Leibniz Information Center for Economics, Kiel, Germany |
| Stefan Schlobach | Vrije Universiteit Amsterdam, The Netherlands |
| Daniel Schwabe | Pontifical Catholic University of Rio de Janeiro, Brazil |
| Monika Solanki | Aston Business School, Aston University, UK |
| Dezhao Song | Thomson Reuters, USA |
| Milan Stankovic | Sépage and STIH, Université Paris-Sorbonne, France |
| Markus Strohmaier | University of Koblenz-Landau, Germany |
| Rudi Studer | Karlsruher Institut für Technologie (KIT), Germany |
| Gerd Stumme | University of Kassel, Germany |
| Vojtěch Svátek | University of Economics, Prague, Czech Republic |
| Valentina Tamma | Science, University of Liverpool, UK |
| Kerry Taylor | CSIRO and Australian National University, Australia |
| Matthias Thimm | Universität Koblenz-Landau, Germany |
| Krishnaprasad Thirunarayan | Kno.e.sis Center, Wright State University, USA |
| Ioan Toma | STI Innsbruck, Austria |
| Raphaël Troncy | EURECOM, France |
| Anni-Yasmin Turhan | TU Dresden, Germany |
| Jacopo Urbani | Vrije Universiteit Amsterdam, The Netherlands |
| Jacco van Ossenburg | CWI and VU University Amsterdam, The Netherlands |
| María-Esther Vidal | Universidad Simon Bolivar, Venezuela |
| Daniel Vila Suero | Universidad Politécnica de Madrid, Spain |

| | |
|----------------|---|
| Haofen Wang | East China University of Science and Technology, China |
| Kewen Wang | Griffith University, USA |
| Zhichun Wang | Beijing Normal University, China |
| Fouad Zablith | American University of Beirut, Lebanon |
| Qingpeng Zhang | Rensselaer Polytechnic Institute, USA |

Additional Reviewers

| | | |
|-----------------------|------------------------|-----------------------|
| Aggarwal, Nitish | Joshi, Amit | Rei, Luis |
| Angles, Renzo | Kaminski, Mark | Rezk, Martin |
| Banda, Fredah | Kapahnke, Patrick | Ritze, Dominique |
| Bedathur, Srikanta | Keskisärkkä, Robin | Rizzo, Giuseppe |
| Beek, Wouter | Kim, Eunkyung | Sanchez Ayte, Adam |
| Bertossi, Leopoldo | Kirrane, Sabrina | Savenkov, Vadim |
| Bortoli, Stefano | Knuth, Magnus | Schalk, Andrea |
| Braghin, Stefano | Kondylakis, Haridimos | Schlobach, Stefan |
| Buil Aranda, Carlos | Kosmerlj, Aljaz | Schmidt, Andreas |
| Butt, Anila Sahar | Kämpgen, Benedikt | Sengupta, Kunal |
| Carral, David | Lalithsena, Sarasi | Shekarpour, Saeedeh |
| Catasta, Michele | Lefort, Laurent | Solimando, Alessandro |
| Chekol, | Loebe, Frank | Steyskal, Simon |
| Melisachew Wudage | Lu, Chun | Taheri, Aynaz |
| Chen, Lu | Martinez-Prieto, | Tamma, Valentina |
| Cheng, Long | Miguel A. | Tiddi, Ilaria |
| David, Jérôme | Masopust, Tomas | Todorov, Konstantin |
| Davis, Brian | Mazzola, Luca | Tommasi, Pierpaolo |
| Denaux, Ronald | Meroño-Peñuela, Albert | Tonon, Alberto |
| Doerfel, Stephan | Meusel, Robert | Tran, Trung-Kien |
| Dou, Dejing | Mirrezaei, Iman | Tzitzikas, Yannis |
| Dragisic, Zlatan | Mongiovi, Misael | Unbehauen, Joerg |
| Ecke, Andreas | Nenov, Yavor | Vrgoc, Domagoj |
| Ell, Basil | Niebler, Thomas | Wang, Cong |
| Flouris, Giorgos | Novalija, Inna | Wang, Wenbo |
| Fundulaki, Irini | Nuzzolese, | Wang, Zhe |
| Färber, Michael | Andrea Giovanni | Wu, Jiewen |
| Gillani, Syed | Palmonari, Matteo | Wu, Tianxing |
| Guéret, Christophe | Piro, Robert | Xiao, Guohui |
| Hammar, Karl | Porrini, Riccardo | Zappa, Achille |
| Hentschel, Christian | Potoniec, Jędrzej | Zhang, Gensheng |
| Huan, Gao | Ratcliffe, David | Zhang, Xiaowang |
| Jayaram, Nandish | Ravindra, Padmashree | Zheleznyakov, Dmitriy |
| Jimenez-Ruiz, Ernesto | Reddy, Dinesh | Zhuang, Zhiqiang |

Sponsors

Platinum Sponsors

Elsevier

Gold Sponsors

Fujitsu
Google
iMinds
Ontotext
Systap
Yahoo! Labs

Silver Sponsors

Franz Inc.



Contents – Part II

In-Use and Software Track

SPARQL and Querying Linked Data

- RDFox: A Highly-Scalable RDF Store. 3
*Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu,
and Jay Banerjee*
- TR Discover: A Natural Language Interface for Querying and Analyzing
Interlinked Datasets. 21
*Dezhao Song, Frank Schilder, Charese Smiley, Chris Brew,
Tom Zielund, Hiroko Bretz, Robert Martin, Chris Dale, John Duprey,
Tim Miller, and Johanna Harrison*

Linked Data

- Drug Encyclopedia – Linked Data Application for Physicians. 41
Jakub Kozák, Martin Nečaský, and Jaroslav Pokorný
- Collecting, Integrating, Enriching and Republishing Open City Data
as Linked Data 57
Stefan Bischof, Christoph Martin, Axel Polleres, and Patrik Schneider
- ASSESS — Automatic Self-Assessment Using Linked Data. 76
Lorenz Bühmann, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo

Ontology-Based Data Access

- Ontology Based Access to Exploration Data at Statoil 93
*Evgeny Kharlamov, Dag Hovland, Ernesto Jiménez-Ruiz, Davide Lanti,
Hallstein Lie, Christoph Pinkel, Martin Rezk, Martin G. Skjæveland,
Evgenij Thorstensen, Guohui Xiao, Dmitriy Zheleznyakov,
and Ian Horrocks*
- BooTOX: Practical Mapping of RDBs to OWL 2 113
*Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov,
Ian Horrocks, Christoph Pinkel, Martin G. Skjæveland,
Evgenij Thorstensen, and Jose Mora*

| | |
|--|-----|
| Assessing and Refining Mappings to RDF to Improve Dataset Quality | 133 |
| <i>Anastasia Dimou, Dimitris Kontokostas, Markus Freudenberg, Ruben Verborgh, Jens Lehmann, Erik Mannens, Sebastian Hellmann, and Rik Van de Walle</i> | |

| | |
|--|-----|
| A Generic RDF Transformation Software and Its Application to an Online Translation Service for Common Languages of Linked Data. | 150 |
| <i>Olivier Corby, Catherine Faron-Zucker, and Fabien Gandon</i> | |

Ontology and Instance Alignment

| | |
|--|-----|
| Multilingual Ontology Mapping in Practice: A Support System for Domain Experts. | 169 |
| <i>Mauro Dragoni</i> | |

| | |
|---|-----|
| Data Access Linking and Integration with DALI: Building a Safety Net for an Ocean of City Data | 186 |
| <i>Vanessa Lopez, Martin Stephenson, Spyros Kotoulas, and Pierpaolo Tommasi</i> | |

Knowledge Graphs

| | |
|--|-----|
| Building and Using a Knowledge Graph to Combat Human Trafficking. | 205 |
| <i>Pedro Szekely, Craig A. Knoblock, Jason Slepicka, Andrew Philpot, Amandeep Singh, Chengye Yin, Dipsy Kapoor, Prem Natarajan, Daniel Marcu, Kevin Knight, David Stallard, Subesswara S. Karunamoorthy, Rajagopal Bojanapalli, Steven Minton, Brian Amanatullah, Todd Hughes, Mike Tamayo, David Flynt, Rachel Artiss, Shih-Fu Chang, Tao Chen, Gerald Hiebel, and Lidia Ferreira</i> | |

Data Processing, IoT, Sensors

| | |
|--|-----|
| Semantic-Guided Feature Selection for Industrial Automation Systems. | 225 |
| <i>Martin Ringsquandl, Steffen Lamparter, Sebastian Brandt, Thomas Hubauer, and Raffaello Lepratti</i> | |

| | |
|--|-----|
| A Semantic Processing Framework for IoT-Enabled Communication Systems | 241 |
| <i>Muhammad Intizar Ali, Naomi Ono, Mahedi Kaysar, Keith Griffin, and Alessandra Mileo</i> | |

Data Sets and Ontologies Track

SPARQL and Querying Linked Data

- LSQ: The Linked SPARQL Queries Dataset. 261
*Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan,
 Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo*
- Automatic Curation of Clinical Trials Data in LinkedCT 270
Oktie Hassanzadeh and Renée J. Miller

Linked Data

- DBpedia Commons: Structured Multimedia Metadata from the Wikimedia
 Commons 281
*Gaurav Vaidya, Dimitris Kontokostas, Magnus Knuth, Jens Lehmann,
 and Sebastian Hellmann*

Archiving and Publishing Scientific Data

- Provenance-Centered Dataset of Drug-Drug Interactions. 293
Juan M. Banda, Tobias Kuhn, Nigam H. Shah, and Michel Dumontier
- The GeoLink Modular Oceanography Ontology 301
*Adila Krisnadhi, Yingjie Hu, Krzysztof Janowicz, Pascal Hitzler,
 Robert Arko, Suzanne Carbotte, Cynthia Chandler, Michelle Cheatham,
 Douglas Fils, Timothy Finin, Peng Ji, Matthew Jones, Nazifa Karima,
 Kerstin Lehnert, Audrey Mickle, Thomas Narock, Margaret O'Brien,
 Lisa Raymond, Adam Shepherd, Mark Schildhauer, and Peter Wiebe*
- Semantic Bridges for Biodiversity Sciences 310
*Natalia Villanueva-Rosales, Nicholas del Rio, Deana Pennington,
 and Luis Garnica Chavira*

IoT and Sensors

- FraPPE: A Vocabulary to Represent Heterogeneous Spatio-temporal Data
 to Support Visual Analytics 321
Marco Balduini and Emanuele Della Valle
- The Transport Disruption Ontology 329
David Corsar, Milan Markovic, Peter Edwards, and John D. Nelson

Empirical Studies and Experiments Track

Experiments

| | |
|--|-----|
| LOD Lab: Experiments at LOD Scale | 339 |
| <i>Laurens Rietveld, Wouter Beek, and Stefan Schlobach</i> | |
| Strategies for Efficiently Keeping Local Linked Open Data Caches Up-To-Date | 356 |
| <i>Renata Dividino, Thomas Gottron, and Ansgar Scherp</i> | |
| CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets | 374 |
| <i>Muhammad Intizar Ali, Feng Gao, and Alessandra Mileo</i> | |

Evaluation

| | |
|--|-----|
| A Multi-reasoner, Justification-Based Approach to Reasoner Correctness | 393 |
| <i>Michael Lee, Nico Matentzoglou, Bijan Parsia, and Uli Sattler</i> | |
| Introducing Defeasibility into OWL Ontologies. | 409 |
| <i>Giovanni Casini, Thomas Meyer, Kody Moodley, Uli Sattler, and Ivan Varzinczak</i> | |

Empirical Studies

| | |
|--|-----|
| Timely Semantics: A Study of a Stream-Based Ranking System for Entity Relationships | 429 |
| <i>Lorenz Fischer, Roi Blanco, Peter Mika, and Abraham Bernstein</i> | |
| Link Analysis of Life Science Linked Data | 446 |
| <i>Wei Hu, Honglei Qiu, and Michel Dumontier</i> | |
| Author Index | 463 |

Contents – Part I

Research Task

Querying with SPARQL

| | |
|--|----|
| SPARQL with Property Paths | 3 |
| <i>Egor V. Kostylev, Juan L. Reutter, Miguel Romero, and Domagoj Vrgoč</i> | |
| Recursion in SPARQL. | 19 |
| <i>Juan L. Reutter, Adrián Soto, and Domagoj Vrgoč</i> | |
| Federated SPARQL Queries Processing with Replicated Fragments. | 36 |
| <i>Gabriela Montoya, Hala Skaf-Molli, Pascal Molli, and Maria-Esther Vidal</i> | |
| FEASIBLE: A Feature-Based SPARQL Benchmark Generation Framework . . . | 52 |
| <i>Muhammad Saleem, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo</i> | |

Querying Linked Data

| | |
|--|-----|
| LDQL: A Query Language for the Web of Linked Data | 73 |
| <i>Olaf Hartig and Jorge Pérez</i> | |
| Opportunistic Linked Data Querying Through Approximate Membership Metadata | 92 |
| <i>Miel Vander Sande, Ruben Verborgh, Joachim Van Herwegen, Erik Mannens, and Rik Van de Walle</i> | |
| Networks of Linked Data Eddies: An Adaptive Web Query Processing Engine for RDF Data | 111 |
| <i>Maribel Acosta and Maria-Esther Vidal</i> | |
| Substring Filtering for Low-Cost Linked Data Interfaces | 128 |
| <i>Joachim Van Herwegen, Laurens De Vocht, Ruben Verborgh, Erik Mannens, and Rik Van de Walle</i> | |

Linked Data

| | |
|--|-----|
| LinkDaViz – Automatic Binding of Linked Data to Visualizations | 147 |
| <i>Klaudia Thellmann, Michael Galkin, Fabrizio Orlandi, and Sören Auer</i> | |

| | |
|---|-----|
| Facilitating Entity Navigation Through Top-K Link Patterns. | 163 |
| <i>Liang Zheng, Yuzhong Qu, Jidong Jiang, and Gong Cheng</i> | |

| | |
|--|-----|
| Serving DBpedia with DOLCE – More than Just Adding a Cherry on Top . . . | 180 |
| <i>Heiko Paulheim and Aldo Gangemi</i> | |

Ontology-Based Data Access

| | |
|--|-----|
| Ontology-Based Integration of Cross-Linked Datasets | 199 |
| <i>Diego Calvanese, Martin Giese, Dag Hovland, and Martin Rezk</i> | |

| | |
|---|-----|
| Mapping Analysis in Ontology-Based Data Access: Algorithms and Complexity | 217 |
| <i>Domenico Lembo, Jose Mora, Riccardo Rosati, Domenico Fabio Savo, and Evgenij Thorstensen</i> | |

Ontology Alignment

| | |
|--|-----|
| Towards Defeasible Mappings for Tractable Description Logics | 237 |
| <i>Kunal Sengupta and Pascal Hitzler</i> | |

| | |
|---|-----|
| An Algebra of Qualitative Taxonomical Relations for Ontology Alignments . . . | 253 |
| <i>Armen Inants and Jérôme Euzenat</i> | |

| | |
|--|-----|
| CogMap: A Cognitive Support Approach to Property and Instance Alignment | 269 |
| <i>Jan Nöbner, David Martin, Peter Z. Yeh, and Peter F. Patel-Schneider</i> | |

| | |
|---|-----|
| Effective Online Knowledge Graph Fusion. | 286 |
| <i>Haofen Wang, Zhijia Fang, Le Zhang, Jeff Z. Pan, and Tong Ruan</i> | |

Reasoning

| | |
|---|-----|
| Adding <i>DL-Lite</i> TBoxes to Proper Knowledge Bases. | 305 |
| <i>Giuseppe De Giacomo and Hector Levesque</i> | |

| | |
|---|-----|
| R_2O_2 : An Efficient Ranking-Based Reasoner for OWL Ontologies | 322 |
| <i>Yong-Bin Kang, Shonali Krishnaswamy, and Yuan-Fang Li</i> | |

| | |
|--|-----|
| Rewriting-Based Instance Retrieval for Negated Concepts in Description Logic Ontologies | 339 |
| <i>Jianfeng Du and Jeff Z. Pan</i> | |

| | |
|---|-----|
| Optimizing the Computation of Overriding. | 356 |
| <i>Piero A. Bonatti, Iliana M. Petrova, and Luigi Sauro</i> | |

Instance Matching, Entity Resolution and Topic Generation

| | |
|---|-----|
| LANCE: Piercing to the Heart of Instance Matching Tools. | 375 |
| <i>Tzanina Saveta, Evangelia Daskalaki, Giorgos Flouris, Irini Fundulaki, Melanie Herschel, and Axel-Cyrille Ngonga Ngomo</i> | |
| Decision-Making Bias in Instance Matching Model Selection | 392 |
| <i>Mayank Kejriwal and Daniel P. Miranker</i> | |
| Klink-2: Integrating Multiple Web Sources to Generate Semantic Topic Networks | 408 |
| <i>Francesco Osborne and Enrico Motta</i> | |
| TabEL: Entity Linking in Web Tables | 425 |
| <i>Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey</i> | |
| Path-Based Semantic Relatedness on Linked Data and Its Use to Word and Entity Disambiguation | 442 |
| <i>Ioana Hulpus, Narumol Prangnawarat, and Conor Hayes</i> | |
| SANAPHOR: Ontology-Based Coreference Resolution | 458 |
| <i>Roman Prokofyev, Alberto Tonon, Michael Luggen, Loic Vouilloz, Djellel Eddine Difallah, and Philippe Cudré-Mauroux</i> | |
| Improving Entity Retrieval on Structured Data | 474 |
| <i>Besnik Fetahu, Ujwal Gadiraju, and Stefan Dietze</i> | |

RDF Data Dynamics

| | |
|---|-----|
| A Flexible Framework for Understanding the Dynamics of Evolving RDF Datasets | 495 |
| <i>Yannis Roussakis, Ioannis Chrysakis, Kostas Stefanidis, Giorgos Flouris, and Yannis Stavarakas</i> | |
| Interest-Based RDF Update Propagation. | 513 |
| <i>Kemele M. Endris, Sidra Faisal, Fabrizio Orlandi, Sören Auer, and Simon Scerri</i> | |

Ontology Extraction and Generation

| | |
|---|-----|
| General Terminology Induction in OWL | 533 |
| <i>Viachaslau Sazonau, Uli Sattler, and Gavin Brown</i> | |
| Understanding How Users Edit Ontologies: Comparing Hypotheses About Four Real-World Projects | 551 |
| <i>Simon Walk, Philipp Singer, Lisette Espin Noboa, Tania Tudorache, Mark A. Musen, and Markus Strohmaier</i> | |

| | |
|--|-----|
| Next Step for NoHR: OWL 2 QL | 569 |
| <i>Nuno Costa, Matthias Knorr, and João Leite</i> | |
| Concept Forgetting in <i>ALCOI</i> -Ontologies Using an Ackermann Approach. . . | 587 |
| <i>Yizheng Zhao and Renate A. Schmidt</i> | |
| Knowledge Graphs and Scientific Data Publication | |
| Content-Based Recommendations via DBpedia and Freebase: A Case Study in the Music Domain. | 605 |
| <i>Phuong T. Nguyen, Paolo Tomeo, Tommaso Di Noia, and Eugenio Di Sciascio</i> | |
| Explaining and Suggesting Relatedness in Knowledge Graphs | 622 |
| <i>Giuseppe Pirrò</i> | |
| Type-Constrained Representation Learning in Knowledge Graphs | 640 |
| <i>Denis Krompaß, Stephan Baier, and Volker Tresp</i> | |
| Publishing Without Publishers: A Decentralized Approach to Dissemination, Retrieval, and Archiving of Data. | 656 |
| <i>Tobias Kuhn, Christine Chichester, Michael Krauthammer, and Michel Dumontier</i> | |
| Author Index | 673 |

SPARQL and Querying Linked Data

RDFox: A Highly-Scalable RDF Store

Yavor Nenov¹(✉), Robert Piro¹, Boris Motik¹, Ian Horrocks¹, Zhe Wu²,
and Jay Banerjee²

¹ University of Oxford, Oxford, UK

{yavor.nenov,robert.piro,boris.motik,ian.horrocks}@cs.ox.ac.uk

² Oracle Corporation, Redwood Shores, CA, USA

{alan.wu,jayanta.banerjee}@oracle.com

Abstract. We present RDFox—a main-memory, scalable, centralised RDF store that supports materialisation-based parallel datalog reasoning and SPARQL query answering. RDFox uses novel and highly-efficient parallel reasoning algorithms for the computation and incremental update of datalog materialisations with efficient handling of `owl:sameAs`. In this system description paper, we present an overview of the system architecture and highlight the main ideas behind our indexing data structures and our novel reasoning algorithms. In addition, we evaluate RDFox on a high-end SPARC T5-8 server with 128 physical cores and 4TB of RAM. Our results show that RDFox can effectively exploit such a machine, achieving speedups of up to 87 times, storage of up to 9.2 billion triples, memory usage as low as 36.9 bytes per triple, importation rates of up to 1 million triples per second, and reasoning rates of up to 6.1 million triples per second.

1 Introduction

An increasing number of Semantic Web applications represent knowledge and data using the Resource Description Framework (RDF) [12]. Such applications use RDF stores to efficiently store large amounts of RDF data, manage background knowledge about a domain, and answer queries. The background knowledge is usually captured using an OWL 2 ontology [18], possibly extended with SWRL rules [10]. An ontology describes dependencies between entities, which allows an RDF store to enrich query answers with results not explicitly stated in the data. Queries are typically expressed in SPARQL [21], and the main computational service of RDF stores is to evaluate queries over both the explicit facts and the facts implied by the background knowledge. Answering queries with respect to arbitrary OWL 2 ontologies is often infeasible in practice due to the high computational complexity of the logical formalisms that underpin OWL 2 [9]. OWL 2 profiles [13] deal with intractability by restricting the expressivity of the ontology language in a way that enables efficient query answering over large datasets. OWL 2 RL is one such profile that is supported, at least to an extent, by many state of the art RDF stores. Consequences of OWL 2 RL ontologies can be captured using *datalog* [1]—a rule-based language developed by both the database and the knowledge representation communities. Queries over a datalog program and a dataset can be answered in several

different ways. In scenarios where the performance of query answering is critical, a common approach is to precompute and explicitly store all consequences of the program and the dataset so that subsequent queries can be evaluated without any further reference to the program. This approach is also known as *materialisation*, and it is used in state of the art systems such as GraphDB [3] and Oracle’s RDF store [22].

In this system description paper we present RDFox—a highly scalable, centralised, main-memory RDF store that supports materialisation-based parallel datalog reasoning and SPARQL query answering. It is developed and maintained at the University of Oxford and is available for download¹ under an academic licence. It is available on Linux, Mac OS X, Solaris, and Windows. It can be integrated as a library into C++, Java, and Python applications using an efficient native API; moreover, it can also be used as a standalone server accessible via a SPARQL endpoint. These versatile modes of use, combined with the very efficient storage and reasoning capabilities that we describe next, make RDFox suitable for a wide range of Semantic Web application scenarios.

RDFox supports datalog reasoning over RDF data using several novel datalog evaluation algorithms. To compute datalog materialisations, RDFox uses a shared-memory parallel algorithm that evenly distributes workload to threads by partitioning the reasoning task into many small, independent subtasks [15]. To support changes to the input data without recomputing materialisations from scratch, RDFox employs a novel incremental reasoning algorithm [14] that reduces the overall work by identifying early on whether a fact should be deleted or kept as a consequences of the update.

Many Semantic Web applications use the `owl:sameAs` property to state equalities between resources, which should be taken into account during materialisation. With many equality statements, however, this can significantly increase the memory consumption and degrade the overall reasoning performance [11]. *Rewriting* is a well-known technique for efficient equality reasoning [2, 20], where equal resources are substituted during reasoning by a common representative. The result of this technique is called an *r-materialisation*, and it consists of a mapping between resources and their representatives and a dataset over the representatives. The correct computation of r-materialisations is not straightforward even on a single thread, since equalities derived during reasoning may trigger changes of representatives, which may require the deletion of outdated facts as well as changes to the datalog rules. RDFox employs a novel algorithm that seamlessly incorporates the rewriting technique into the parallel materialisation processes without sacrificing the benefits of parallelisation [17]. Moreover, updating r-materialisations incrementally is highly nontrivial, and the main difficulties stem from the fact that retraction of equalities between resources requires the reevaluation of all facts containing the representative of these resources. RDFox provides support for the incremental update of *r-materialisations* using a novel algorithm that was proved very efficient for small to medium-sized updates [16].

¹ <http://www.rdflox.org/>

To the best of our knowledge, RDFox is the only system that supports incremental update of r-materialisations.

To ensure scalability of data storage and access, RDFox uses a novel, efficient RDF storage scheme. It stores RDF triples in RAM, which is much faster than disk-based schemes, particularly for random access. The storage scheme uses compact data structures that can store hundreds of millions of triples on commodity PCs, and tens of billions of triples on high-end servers. The storage scheme comes in two variants: an economical version that can store up to four billion triples, and a more scalable version that can store more triples at the expense of using more bytes per triple. The storage scheme has configurable indexes that support efficient data access. All indexes support highly scalable, ‘almost’ lock-free parallel updates, which is critical for the performance of parallel reasoning. A particular challenge is to ensure eager elimination of duplicate triples, which is important for both the performance and correctness of reasoning.

In our previous work, we have demonstrated the scalability of RDFox on mid-range servers. In particular, RDFox can store 1.5 G triples in 52 G of RAM [15]; on 16 physical cores our parallel materialisation algorithms achieve reasoning speedup over the single-threaded version of up to 13.9 [15, 17]; even in the single-threaded mode, RDFox often outperforms state of the art solutions based on relational and columnar databases [15]; and the (r-)materialisation can be efficiently updated for small to medium-sized updates [14, 16]. To test the limits of the storage and reasoning scalability of RDFox, in this paper we shift our focus to high-end servers and present the results of a performance evaluation on a SPARC T5 with TB of RAM and 128 physical cores powering 1024 virtual cores via hyperthreading. Our evaluation shows very promising results: RDFox achieved speedups of up to 87 times (with 1024 threads) over the single-threaded version, storage of up to 9.2 billion triples, memory usage as low as 36.9 bytes per triple, importation rates of up to 1 million triples per second, and reasoning rates of up to 6.1 million triples per second.

The rest of the paper is structured as follows. In Section 2 we discuss the features and the different ways of accessing RDFox. In Section 3 we discuss in detail the architecture of RDFox. In Section 4 we demonstrate by means of an example the key ideas behind the algorithms used in RDFox. Finally, in Section 5 we describe the results of our performance evaluation.

2 Features, APIs, and Use Cases of RDFox

We now discuss the features of RDFox, the possible ways in which the system can be integrated into applications, and practical scenarios in which it has been employed.

RDFox Features. In RDFox, a *data store* is the basic RDF data management unit. Each data store is associated with a type that determines the data storage and indexing strategies. In any application, one can instantiate an arbitrary number of data stores, each of which provides the following functionality.

- Triples can be added to a data store in one of three possible ways: they can be *imported*, after which they are available for querying and reasoning, or they can be scheduled for *incremental addition* or *incremental deletion*, which makes them available for incremental reasoning. In each case, triples can be added programmatically, read from an RDF 1.1 Turtle file, or extracted from an OWL 2 ontology.
- Analogously, a data store can import, or schedule for addition or deletion a set of datalog rules. Rules can be represented programmatically, read from a file in a custom *RDF datalog* format, or extracted from the OWL RL fragment of an ontology.
- A data store can answer SPARQL queries. Currently, RDFox supports most, but not all of SPARQL 1.1; in particular, we are still working on supporting aggregate queries and property paths.
- A data store can materialise the available triples with respect to the current set of rules. Reasoning can be carried out with or without optimised equality handling; the data store ensures that observable results in both cases are identical. The materialisation becomes available for querying immediately after reasoning has completed.
- One can incrementally update the materialisation according to the triples and rules scheduled for addition and/or deletion. RDFox does not support transactional updates; thus, the results of SPARQL queries are uniquely defined only after incremental update terminates.
- The triples in a store can be exported into a Turtle file, and the rules in a store can be exported into an RDF datalog file.
- The entire contents of a data store can be saved into a binary file, which can later be loaded to completely restore the state of the data store.

RDFox APIs. The core of RDFox is written in C++, but the system supports a number of APIs that enable integration with different kinds of applications.

- RDFox can be loaded as a C++ library, and all of its functionality can be easily accessed using a rich C++ API.
- RDFox can be accessed as a native library from Java and Python using suitable APIs. The Java and Python APIs act as a façade over the C++ API and provide access to the commonly used functionality.
- RDFox also supports a simple scripting language that supports command-line interaction with the system. This mode of interaction is particularly useful for ad hoc tests of the system’s functionality, as well as for exploring the data and the effects of various operations.
- RDFox can be started in a server mode, providing access via a SPARQL endpoint. The endpoint currently supports only query answering, but our future plans include support for SPARQL updates.

Use Cases. RDFox is used in various industrial prototype systems, some of which we describe next.

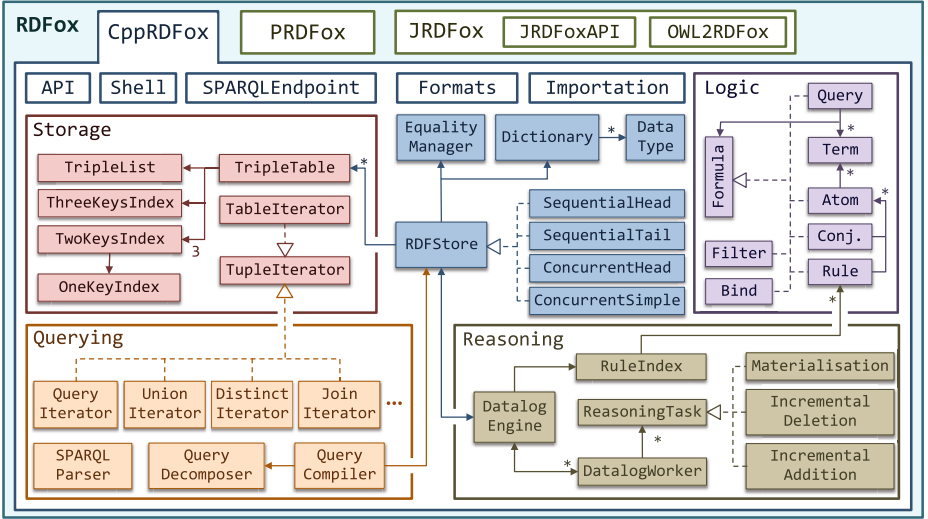


Fig. 1. RDFox Architecture

- **Statoil ASA** is a Norwegian multinational oil and gas company, and they are currently using RDFox as part of a large-scale Semantic Web application that facilitates the integration and analysis of oil production and geological survey data.
- **Électricité de France (EDF)** is a French electric utility company, and they are currently using RDFox to manage and analyse information about their electricity distribution network.
- **Kaiser Permanente** is a US health care consortium, and they are currently using RDFox to analyse patient data records.

3 System Architecture

The architecture of RDFox is summarised in Figure 1. It consists of a C++ module `CppRDFox`, a Java module `JRDFox`, and a Python module `PRDFox`. `CppRDFox` implements most of the functionality of RDFox and we discuss its structure in the rest of this section. `JRDFox` and `PRDFox` implement the Java and Python APIs, respectively, for the core functionality of `CppRDFox`. Moreover, `OWL2RDFox` uses the OWL API to load OWL 2 ontologies, extract their OWL 2 RL part, and translate the result into datalog; we discuss this translation in Section 4.2.

3.1 An Overview of CppRDFox

An `RDFStore` is the central concept of `CppRDFox` responsible for the storage, materialisation, and access of RDF data. There are several `RDFStore` variants,

each differing in its storage capacity, indexing scheme, support for parallel operation, and support for `owl:sameAs` reasoning; some of these variants are shown in Figure 1. Crucially, *sequential* store variants support only single-threaded access but without any synchronisation overhead, whereas *parallel* store variants support multi-threaded access and parallel materialisation. An `RDFStore` relies on the `TripleTable` from the `Storage` package for efficient storage of RDF triples and on the `DatalogEngine` from the `Reasoning` package for efficient datalog reasoning. Following common practice in RDF stores, RDF resources are encoded as integers to support efficient data storage and access; the `Dictionary` component manages this encoding. Finally, the `EqualityManager` component records the representatives for equal individuals.

The `Logic` package models well-known concepts from first-order logic, such as triple patterns, rules, and queries. The classes in this package provide the basis for the C++ API as they are used to represent the input to most components of the system. In this way, applications can interact with `RDFox` programmatically, and not just by representing triples and rules textually, which can eliminate a significant source of overhead.

The `Formats` package provides a pluggable architecture for various input/output formats supported by `RDFox`. At present, Turtle 1.1 and RDF datalog are fully supported, and support for RDF/XML is being finalised. The `Importation` package implements parallel importation of files. The `Querying` package handles the evaluation of SPARQL queries. The `Reasoning` package implements materialisation and incremental update algorithms. Finally, the `API` package provides a C façade over the native C++ API; the `Shell` package implements the aforementioned scripting language; and the `SPARQLEndPoint` package implements a SPARQL endpoint.

3.2 Components of `CppRDFox`

We next describe the main components of `CppRDFox` in more detail.

Dictionary. As is common in practice [4], the `Dictionary` component of `RDFox` encodes each RDF resource using a unique integer ID. These IDs are allocated sequentially starting from one and are thus ‘small’, which allows us to use the IDs as array indexes in various parts of the `RDFStore` component. A `Dictionary` uses various `DataType` implementations to handle different kinds of RDF literals. Each `DataType` instance is responsible for converting a lexical literal representation into a binary one, which involves literal normalisation. For example, integers 007 and 7 are both represented as the same binary value 7. While such an approach deviates slightly from the RDF specification, it allows us to store literals efficiently.

Storage. The `RDFStore` component stores RDF triples using a `TripleTable` component. Each `TripleTable` consists of a `TripleList` that stores the actual triples, as well as a number of indexes that support efficient iteration over subsets of the triples stored in the `TripleList`. A `TripleList` stores RDF triples as a two-dimensional array with six columns: the first three columns hold the IDs

of the subject, predicate, and object of a triple, while the latter three columns are used for indexing. In particular, the triples in the `TripleList` are organised in three linked lists, each of which is grouped by subject, predicate, and object, respectively; thus, the last three columns in the `TripleList` provide the next pointers in the respective lists. These linked lists are used to efficiently iterate over triples matching a combination of subject, predicate, and object. The grouping is fully configurable (at compile time), and it is chosen to support efficient answering of *triple patterns* of the form (t_s, t_p, t_o) , where each of t_s , t_p , and t_o is either a variable or a resource identifier; the iteration is achieved via the `TableIterator` component. The `ThreeKeysIndex` implements a hash table over all triples in the `TripleList`, thus allowing for efficient duplicate elimination. We next discuss how RDFox answers different kinds of triple patterns.

Triple patterns not containing individuals are answered by sequentially scanning the `TripleList` and skipping over triples that do not match the pattern; for example, (x, y, x) is answered by skipping over triples whose subject and object differ. Triple patterns containing only individuals (i.e., variable-free patterns) are answered by a lookup into the `ThreeKeysIndex`. For triple patterns with one or two variables, a `TripleTable` relies on three `TwoKeysIndex` components, one for each of the components subject, predicate, and object, each maintaining one of the three triple lists. Each `TwoKeysIndex` contains a `OneKeyIndex` that, given a resource ID, locates the first triple in the relevant list with the given ID. Triple patterns containing just one resource ID are thus answered by iterating over the relevant list and possibly skipping over triples not matching the pattern. Since IDs are ‘small’, `OneKeyIndex` is implemented as an array, which supports efficient update and access. Triple patterns containing two resources are answered in two possible ways, depending on the configuration of the relevant `TwoKeysIndex`.

In the *simple* configuration, the `TwoKeysIndex` works similarly to when just one component is specified: it iterates over all triples containing the first component, and skips over the triples not matching the rest of the pattern. For example, if the triple pattern is (s, p, z) , the `TwoKeysIndex` maintaining the triple list for s uses its `OneKeyIndex` to identify all triples containing s in the subject component, and skips over all triples not containing p in the predicate component. The benefit of this indexing scheme is simplicity of updates and low memory use, but the drawback is that answering certain triple patterns can be inefficient due to skipping.

In the *complex* configuration, the `TwoKeysIndex` maintains its relevant triple list grouped by an additional component, and it uses a hash table to efficiently locate the relevant sublist. For example, the `TwoKeysIndex` can keep the triples organised by first subject and then predicate; then, the triple pattern (s, p, z) is answered by querying the hash table by (s, p) to find the triples that contain s and p in their subject and predicate components, respectively, and by iterating over the triples in the list until the predicate component becomes different from p . Triple patterns of the form (s, y, o) are answered as in the case of a simple `TwoKeysIndex`. This indexing scheme offers more efficient triple retrieval with

no skipping, but comes at the expense of using more memory and more complex updates.

Currently, RDFox supports two kinds of indexing schemes. The *simple indexing scheme* employs a simple `TwoKeysIndex` for each of the three triple lists. In contrast, the *complex indexing scheme* employs a simple `TwoKeysIndex` for the predicate triple list, a complex `TwoKeysIndex` for the subject list grouped by predicate, and a complex `TwoKeysIndex` for the object list grouped by predicate. Hence, the complex indexing scheme can answer directly (i.e. without skipping) all triple patterns except for patterns of the form (s, y, o) , which are delegated to the `TwoKeysIndex` responsible for the subject triple list.

Alternative data indexing schemes, such as the one used in RDF-3X [19], maintain sorted indexes, which allows for high degrees of data compression as well as answering many queries using very efficient merge joins. However, the maintenance of such indexes can be very costly and difficult to parallelise, and so such indexing schemes can be inefficient in scenarios where data changes continuously and in parallel, as is the case of parallel datalog materialisation. In contrast, the data indexing scheme employed by RDFox supports efficient, low-contention parallel maintenance, and is thus highly suitable for parallel datalog materialisation (for full details see [15]).

Querying. The querying package is responsible for SPARQL query answering. To evaluate a query, one can construct a `Query` object either programmatically or by parsing a SPARQL 1.1 query using the `SPARQLParser` component. The `SPARQLCompiler` component converts a `Query` object into an `TupleIterator` component that provides iteration over the answers. The `SPARQLCompiler` can optionally be configured to use a `QueryDecomposer` to produce query evaluation plans based on the extensive theory of queries of bounded treewidth [5]. Such query plans have proved critical to answering certain hard queries, but further investigation is required to make them useful in general. RDFox contains many different `TupleIterator` variants, each implementing specific SPARQL constructs. For example, `TableIterator` supports iteration over SPARQL triple patterns, `DistinctIterator` implements the “DISTINCT” construct of SPARQL, `UnionIterator` implements the “UNION” construct, and `QueryIterator` represents entire queries.

Reasoning. The reasoning package implements datalog materialisation and incremental updates. The `DatalogEngine` component organises the reasoning process. To support parallel reasoning, `DatalogEngine` uses `DatalogWorker` components, each of which can execute on one thread one of the reasoning tasks: `Materialisation`, `IncrementalDeletion`, and `IncrementalAddition`. Note that incremental reasoning is split into two tasks since incremental deletion has to be performed before incremental addition and the latter task cannot start before the former task finishes. Each task can work with or without rewriting of `owl:sameAs`. The datalog program loaded into RDFox is stored in a `RuleIndex` object, which, given a triple, can efficiently identify the rules for which the specified triple matches a body triple pattern.

Importation. For parallel `RDFStore` variants, the `Importation` package can be used to import multiple files in parallel. Requests are handled by the `ImportEngine`, which initialises a configurable number of `ImportWorker` components, and a collection of `ImportTask` components, one for each file to import. Each `ImportWorker` then iteratively extracts and executes an `ImportTask` until all tasks have been processed and so all files have been imported.

4 Datalog Reasoning

In this section, we present an overview of the algorithms that RDFox uses to efficiently compute and update datalog materialisations of RDF data. These algorithms are also applicable to the less expressive but more widely used OWL 2 RL language. Towards this goal, we first discuss how datalog can be integrated with RDF, then we discuss two different ways of supporting OWL 2 RL reasoning using datalog, and finally we demonstrate the key ideas behind our reasoning algorithms by means of an example.

4.1 RDF Datalog

A *term* is a variable or an RDF resource. A *triple pattern* is a triple (t_s, t_p, t_o) , where t_s , t_p , and t_o are terms. An (*RDF*) *rule* r has the form (1)

$$H \leftarrow B_1 \wedge \dots \wedge B_k, \quad (1)$$

where H is the *head* triple pattern, and each B_i , $1 \leq i \leq k$, is a *body* triple pattern. A *program* is a finite set of rules. A rule r' is an *instance* of a rule r if r' can be obtained from r by uniformly replacing the variables in r by RDF resources.

4.2 Common Approaches to OWL 2 RL Reasoning via Datalog

There are two main approaches to OWL 2 RL reasoning in datalog. In the first approach, the data and the ontology axioms are encoded as triples, and they are interpreted using the fixed set of rules from the OWL 2 RL specification [13, Section 4.3]. For example, consider the data triple $(\text{peter}, \text{type}, \text{Teacher})$ (stating that `peter` is a `Teacher`) and the ontological triple $(\text{Teacher}, \text{subClassOf}, \text{Person})$ (stating that the `Teacher` class is a subclass of the `Person` class). Then triple $(\text{peter}, \text{type}, \text{Person})$ follows from these two triples, and it can be derived using the following rule from the OWL 2 RL specification:

$$(x, \text{type}, y_2) \leftarrow (x, \text{type}, y_1) \wedge (y_1, \text{subClassOf}, y_2) \quad (2)$$

Using a fixed rule set may seem appealing due to its simplicity, but it can be inefficient. First, the fixed rules must match both data and ontological triples, and so they often contain many joins. Second, the rule set promotes considerable redundancy. For example, if we add $(\text{Person}, \text{subClassOf}, \text{Mammal})$, due to the

transitivity of `subClassOf` we derive `(Teacher, subClassOf, Mammal)`; but then, rule (2) derives `(peter, type, Mammal)` twice. In practice, such redundant derivations can incur significant overhead.

In the second approach, an OWL 2 RL ontology is translated into datalog rules that derive the same data triples. Our example ontology thus produces the following rules:

$$(x, \text{type}, \text{Person}) \leftarrow (x, \text{type}, \text{Teacher}) \quad (3)$$

$$(x, \text{type}, \text{Teacher}) \leftarrow (x, \text{type}, \text{Mammal}) \quad (4)$$

These rules also derive the triples `(peter, type, Teacher)` and `(peter, type, Mammal)`; however, each rule contains only one body triple pattern and so it can be evaluated more efficiently. Furthermore, all data triples are derived only once.

RDFox can handle arbitrary datalog programs and so it can support both approaches to OWL 2 RL reasoning. For efficiency, we use the second approach in our evaluation.

4.3 Computing Datalog Materialisations

To compute a datalog materialisation, we must exhaustively apply all rules to the dataset until no new triples can be derived. We demonstrate this using an example dataset E and datalog program Σ . The dataset E consists of triples (E1)–(E3), which we call *explicit triples*, and the datalog program Σ consists of the rules (R1)–(R4), which correspond to typical OWL 2 RL axioms.

$$(\text{john}, \text{teach}, \text{math}) \quad (\text{E1})$$

$$(\text{john}, \text{teach}, \text{phys}) \quad (\text{E2})$$

$$(\text{peter}, \text{teach}, \text{math}) \quad (\text{E3})$$

$$(x, \text{type}, \text{Teacher}) \leftarrow (x, \text{type}, \text{Person}) \wedge (x, \text{teach}, y) \wedge (y, \text{type}, \text{Course}) \quad (\text{R1})$$

$$(x, \text{type}, \text{Person}) \leftarrow (x, \text{type}, \text{Teacher}) \quad (\text{R2})$$

$$(x, \text{type}, \text{Person}) \leftarrow (x, \text{teach}, y) \quad (\text{R3})$$

$$(y, \text{type}, \text{Course}) \leftarrow (x, \text{teach}, y) \quad (\text{R4})$$

Rules (R1) and (R2) capture the OWL 2 RL consequences of axiom

$$\text{EquivalentClasses}(\text{Teacher} \text{ ObjectIntersectionOf}(\text{Person} \text{ ObjectSomeValuesFrom}(\text{teach} \text{ Course})))$$

and rules (R3) and (R4) state that classes `Person` and `Course` are the domain and the range, respectively, of property `teach`. The materialisation I of E w.r.t. program Σ extends E with triples (I1)–(I6), which we call *implicit*.

$$(\text{john}, \text{type}, \text{Person}) \quad (\text{I1}) \quad (\text{peter}, \text{type}, \text{Person}) \quad (\text{I4})$$

$$(\text{math}, \text{type}, \text{Course}) \quad (\text{I2}) \quad (\text{john}, \text{type}, \text{Teacher}) \quad (\text{I5})$$

$$(\text{phys}, \text{type}, \text{Course}) \quad (\text{I3}) \quad (\text{peter}, \text{type}, \text{Teacher}) \quad (\text{I6})$$

In particular, applying rule (R3) to either (E1) or (E2) produces (I1); applying rule (R4) to either (E1) or (E3) produces (I2); applying rule (R4) to (E2) produces (I3); and applying rule (R3) to (E3) produces (I4). Moreover, applying rule (R1) to (I1), (I2), and (E1) produces (I5), and applying rule (R1) to (I2), (I4), and (E3) produces (I6). At this point, applying rules (R1)–(R4) to I derives no new triples, so materialisation finishes.

A naïve materialisation approach is to repeatedly apply the rules to the available triples as long as any fresh triples are derived. Using such an approach, we would compute the materialisation as follows: we first apply rules (R1)–(R4) to triples (E1)–(E3) to derive (I1)–(I4); then, we apply (R1)–(R4) again to (E1)–(E3) and (I1)–(I4) to derive (I5)–(I6). This, however, would be very inefficient as in the second application of the rules we would again derive (I1)–(I4) only to discover that these triples have already been derived in the first iteration. Such redundant derivations would pose a considerable source of inefficiency, so such naïve approaches are unsuitable for practical use.

To prevent redundant derivations from the previous paragraph, RDFox uses a novel materialisation algorithm [15] that captures the idea behind the well-known *seminaïve* materialisation approach [1]. The algorithm avoids redundant derivations by considering only rule instances with at least one freshly derived body triple. Roughly speaking, each thread in RDFox extracts an unprocessed triple from the current dataset, matches the triple in all possible ways to triple patterns in a rule body, extends each such match to a rule instance by querying the already processed triples, and adds the instantiated rule head to the dataset. Whenever a thread adds a new triple to the dataset, it notifies all threads that work is available. When there are no more triples to be extracted, the thread goes to sleep if there are still active threads; otherwise, it notifies all threads (all of which must be sleeping) that the materialisation has been completed.

This algorithm breaks down the reasoning process into as many subtasks as there are triples in the materialisation, and these subtasks are dynamically assigned to threads without any need for scheduling or any form of explicit load balancing. Consequently, in all but pathological cases, the algorithm distributes the work to threads evenly. This is in contrast to known approaches that parallelise materialisation by statically assigning either rules or rule instances to threads and are thus often susceptible to data skew.

We next show using our example how our algorithm avoids repeating derivations. A thread first extracts (E1) to derive (I1) and (I2); then, it extracts (E2) to derive (I3); and it extracts (E3) to derive (I4). When the thread extracts (I1), it matches the triple to the first body triple pattern of (R1), but this fails to produce an instance of (R1): although (I2) is available, it has not been processed yet. A thread then extracts (I2) and matches it to the third body triple pattern of (R1); the rule can now be instantiated using only processed triples to derive (I5); thus, the rule instance of (R1) that derives (I5) is considered only once.

4.4 Updating Datalog Materialisations

Instead of recomputing the materialisation from scratch when some of the explicit triples change, it is often desirable to update the materialisation *incrementally*—that is, with as little work as possible. Different such approaches have been considered: some require collecting information during the initial materialisation, whereas others require no extra information; please refer to [14] for an overview. We found the latter approaches more suitable for main-memory systems such as RDFox, where low memory consumption is critical. Moreover, adding explicit triples is generally easy because one can just restart the initial materialisation process, so in the rest of this section we focus on triple deletion.

Assume that we want to delete triple (E1) from our running example. Then, we must identify all triples that can be derived directly or indirectly using (E1), and then determine whether these triples have alternative derivations or need to be deleted as well. The *delete/rederive* (DRed) algorithm [8] is a well-known algorithm that follows this approach, and it proceeds as follows. First, in the *overdeletion* stage, the algorithm identifies all triples that have (directly or indirectly) been derived from (E1). Concretely, the algorithm applies rules (R1), (R3), and (R4) to I while matching at least one body triple pattern to (E1); consequently, triples (I5), (I1), and (I2) are deleted as well. By applying this process further, all implicit triples except (I3) are deleted. All of these triples, however, have an alternative derivation from (E2)–(E3); hence, in the *rederivation* stage, DRed reintroduces all such triples by applying the rules to the ‘surviving’ explicit triples. As this example demonstrates, the algorithm can potentially delete a large portion of the materialised dataset just to reintroduce it later, which can be inefficient. This problem is particularly acute when triples have many alternative derivations, which is often the case in Semantic Web applications.

DRed propagates the deletion of a triple regardless of whether the triple has alternative derivations or not. As a remedy, RDFox uses the *backward/forward* (B/F) algorithm [14]: before deleting a triple, the algorithm checks using a combination of backward and forward chaining whether an alternative derivation exists, and it deletes a triple only if that is not the case. Consider again the deletion of (E1). The B/F algorithm first tries to identify an alternative derivation by matching (E1) to the head of a rule; as this cannot be done, the algorithm deletes (E1). It then examines the direct consequences of (E1) in exactly the same ways as in DRed, and thus identifies (I1), (I2), and (I5) as the direct consequences of (E1). For each of these triples, B/F next tries to identify an alternative proof. In particular, the algorithm determines that (I1) is derivable using rule (R3) the the explicit ‘surviving’ triple (E2), and so it will not delete (I1); this will prevent the algorithm from further considering the consequences of (I1), which improves the overall performance of the algorithm. The checking of alternative proofs is more involved due to the need to ensure termination of backward chaining in the presence of recursive rules; please refer to [14] for details.

4.5 Handling owl:sameAs Using Rewriting

The `owl:sameAs` property states that two resources are equal: if $(a, \text{owl:sameAs}, b)$ holds, then a and b can be used interchangeably. The semantics of `owl:sameAs` can be captured using the following rules:

$$(x_i, \text{owl:sameAs}, x_i) \leftarrow (x_1, x_2, x_3) \quad \text{for } 1 \leq i \leq 3 \quad (\text{EQ1})$$

$$(x'_1, x_2, x_3) \leftarrow (x_1, x_2, x_3) \wedge (x_1, \text{owl:sameAs}, x'_1) \quad (\text{EQ2})$$

$$(x_1, x'_2, x_3) \leftarrow (x_1, x_2, x_3) \wedge (x_2, \text{owl:sameAs}, x'_2) \quad (\text{EQ3})$$

$$(x_1, x_2, x'_3) \leftarrow (x_1, x_2, x_3) \wedge (x_3, \text{owl:sameAs}, x'_3) \quad (\text{EQ4})$$

Rules (EQ2)–(EQ4) ‘copy’ triples between equal resources, which can adversely impact memory consumption [11] and reasoning performance [17].

Rewriting is an optimisation widely used by datalog materialisation algorithms to efficiently handle `owl:sameAs` reasoning. The idea is to replace all equal individuals by a common representative. The result of this technique is called an r-materialisation and it consists of a mapping between resources and their representatives and a dataset over the representatives. Consider, for example, the dataset E and the program Σ_{eq} obtained by extending Σ with rule (R5) that makes property `teach` inverse-functional.

$$(x, \text{owl:sameAs}, y) \leftarrow (x, \text{teach}, z) \wedge (y, \text{teach}, z) \quad (\text{R5})$$

By applying (R5) to (E1) and (E3), we determine that `john` and `peter` are equal. To apply rewriting, we choose one of the two resources as the representative of the other; for example, let us choose `john` as the representative of `peter`. The r-materialisation of E w.r.t. Σ_{eq} then contains triples (I1)–(I3) and (I5), which are obtained from (I1)–(I6) by replacing `peter` with `john`.

The parallel r-materialisation algorithm of RDFox [17] extends the algorithm from Section 4.3. In the extended algorithm, each thread can perform one of three possible actions. First, a thread can extract and process a triple in the dataset; if the triple is outdated (i.e., it contains a resource for which a different representative has been defined), then the triple is deleted and its updated version is added to the dataset; if the triple is of the form $(s, \text{owl:sameAs}, o)$ with $s \neq o$, then the thread identifies one resource as the representative of the other and adds the outdated resource to a special list; and in all other cases the thread applies rules to the triple as in the original materialisation algorithm. Second, a thread can extract an outdated resource c , delete each triple containing c and add its updated version to the dataset, and update all rules containing c . Third, a thread can evaluate a rule that was updated in the previous case.

Updating r-materialisations is nontrivial. First, deleting an equality may actually require *adding* triples. Consider again the dataset E , the program Σ_{eq} , and their r-materialisation computed as explained above, and assume again that we delete triple (E1). After the deletion, `john` is no longer equal to `peter`, and so (I4) and (I6) must be added to the r-materialisation; thus, the r-materialisation after deletion contains (I1)–(I6). Second, if we delete an equality containing a

resource c , we must reevaluate each triple that contains a resource that c represents. RDFox supports incremental r-materialisation updates using a novel algorithm that has been shown to be very efficient for small to medium-sized updates [16].

5 Evaluation

We tested RDFox on an Oracle SPARC T5-8 server. The system has 8 SPARC V9 processors with 16 physical cores per processor, each supporting 8 threads via hyperthreading; thus, the system supports 128 physical and 1024 virtual threads in total. The processors run at 3.6GHz, and each processor has 16KB of instruction cache, 16KB of data cache, 128KB of L2 cache, and 8MB of L3 cache. The system has 4TB of DDR3 memory and is running Solaris 11.1.

Test Data. We now describe the datasets that we used to evaluate RDFox; all datasets are available online.² The Lehigh University Benchmark (**LUBM**) [7] is a widely used synthetic benchmark for RDF systems. The LUBM ontology describes the university domain, and the data is generated by specifying a number of universities, with each university contributing about 100k triples. We used the LUBM-50K dataset with 50,000 universities, which in compressed form was 37 GB. For the rules, we extracted the *lower bound* from the LUBM ontology—that is, we identified the OWL 2 RL part of the ontology and converted it into an RDF datalog program using the transformation by [6]; we call the resulting program $LUBM_L$. **Claros** is a cultural database cataloguing archaeological artefacts. For the rules, we extracted the lower bound as above, but, to push the limits of RDFox, we extended the lower bound with some manually generated rules; we call the resulting datalog program $Claros_{LE}$. **DBpedia** represents structured data extracted from Wikipedia. As in the case of Claros, we extracted the lower bound and extended it with several challenging rules; we call the resulting program $DBpedia_{LE}$.

Materialisation Tests. Table 1 summarises the results of our materialisation tests. For each dataset, we measured the time needed to import the data (shown under ‘import’) without any materialisation, and the memory usage per triple (‘B/trp’) and the number of triples (‘Triples’) after import (‘aft imp’). The number of threads used during import was limited by the number of files storing the data; thus, we used just one thread for Claros and DBpedia, and 11 threads for LUBM-50K. We then computed the materialisation of the dataset while varying the number of threads. For each test, we show the overall time in seconds, as well as the speedup over using just one thread. For each dataset, we show the memory usage per triple (‘B/trp’) and the number of triples (‘Triples’) after materialisation (‘aft mat’). Finally, for each dataset we show the maximum rates in triples/second achieved during import (‘import rate’) and materialisation (‘mat. rate’); the former is the number of triples before materialisation divided by the import time, and the latter is the difference in the numbers of

² <https://krr-nas.cs.ox.ac.uk/2015/ISWC/index.html>

Table 1. Summarisation of the conducted tests

| Threads | LUBM-50K | | Claros | | DBpedia | |
|-------------|----------|---------|--------|---------|---------|---------|
| | sec | speedup | sec | speedup | sec | speedup |
| import | 6.8k | — | 168 | — | 952 | — |
| 1 | 27.0k | 1.0x | 10.0k | 1.0x | 31.2k | 1.0x |
| 16 | 1.7k | 15.7x | 906.0 | 11.0x | 3.0k | 10.4x |
| 32 | 1.1k | 24.0x | 583.3 | 17.1x | 1.8k | 17.5x |
| 48 | 920.7 | 29.3x | 450.8 | 22.2x | 2.0k | 16.0x |
| 64 | 721.2 | 37.4x | 374.9 | 26.7x | 1.2k | 25.8x |
| 80 | 523.6 | 51.5x | 384.1 | 26.0x | 1.2k | 26.7x |
| 96 | 442.4 | 60.9x | 364.3 | 27.4x | 825 | 37.8x |
| 112 | 400.6 | 67.3x | 331.4 | 30.2x | 1.3k | 24.3x |
| 128 | 387.4 | 69.6x | 225.7 | 44.3x | 697.9 | 44.7x |
| 256 | — | — | 226.1 | 44.2x | 684.0 | 45.7x |
| 384 | — | — | 189.1 | 52.9x | 546.2 | 57.2x |
| 512 | — | — | 153.5 | 65.1x | 431.8 | 72.3x |
| 640 | — | — | 140.5 | 71.2x | 393.4 | 79.4x |
| 768 | — | — | 130.4 | 76.7x | 366.2 | 85.3x |
| 896 | — | — | 127.0 | 78.8x | 364.9 | 86.6x |
| 1024 | — | — | 124.9 | 80.1x | 358.8 | 87.0x |
| size | B/trp | Triples | B/trp | Triples | B/trp | Triples |
| aft imp | 124.1 | 6.7G | 80.5 | 18.8M | 58.4 | 112.7M |
| aft mat | 101.0 | 9.2G | 36.9 | 539.2M | 39.0 | 1.5G |
| import rate | 1.0M | | 112k | | 120k | |
| mat. rate | 6.1M | | 4.2M | | 4.0M | |

triples after and before materialisation divided by materialisation time. Note that we could use just one thread while importing Claros and DBpedia, so the import rate is primarily limited by the speed of our Turtle parser. LUBM does not use `owl:sameAs`, so for we used the `RDFStore` without support for rewriting; in contrast, for Claros and DBpedia we used the `RDFStore` with rewriting support. In all cases we used the complex `RDFStore` variant, as it provides more efficient query evaluation.

We were unable to complete tests on LUBM-50K with more than 128 threads. As we discussed in [15], to reduce thread interference, each reasoning thread uses additional memory that depends on the number of triples; hence, RDFox exhausted the available memory with more than 128 threads. Optimising memory consumption with many threads is an important topic for our future work.

Memory Usage. For LUBM-50K, we used a version of `RDFStore` that uses 8-byte pointers and can thus store 2^{64} triples. We initialised the store to pre-allocate sufficient space for the target number of triples after materialisation. This eliminated the need for hash table resizing during import and materialisation, but due to a safety margin on the number of triples, RDFox used 101 bytes/triple, which is more than necessary: without preallocation, LUBM-50K could store the materialised dataset using 89.7 bytes/triple.

For Claros and DBpedia, we used a version of `RDFStore` that uses 4-byte pointers and can thus store 2^{32} triples. Claros was the smallest dataset; however, due to complex rules, materialisation increases the size of the data by a factor of 28. Due to this increase, the share of the size of the `Dictionary` drops from 30% before materialisation to 2%. The resulting dataset contains several large cliques of connected resources, so the variation in the number of different subject–property and object–property pair is low; this ensures that indexes are several

orders of magnitude smaller than the number of triples, so the entire dataset can be stored in only 36.9 bytes/triple. DBpedia is larger than Claros, but its rule set is similar in that it creates cliques of connected individuals. Hence, in the same way as in Claros, the dataset after materialisation can be stored very efficiently, using only 39 bytes/triple.

Reasoning Speedup. The target server supports only 128 physical cores, so 128 is the maximal possible speedup one can expect. As one can see from Table 1, RDFox achieved between 54% and 68% of the maximum, suggesting that our approach to parallelisation of reasoning is very effective. As one can see, parallelisation can be critical for dealing with large datasets and/or complex programs; for example, parallelisation reduces materialisation times on DBpedia from almost 9 hours to just under 6 minutes.

Materialisation in RDFox is a memory-bound task due to random index access, and so each core is susceptible to stalls due to CPU cache misses. However, as one can see from Table 1, hyperthreading seems to effectively compensate for this: on both Claros and DBpedia it roughly doubles the materialisation speed. Please refer to [15] for a more in-depth discussion about the problems related to CPU cache locality.

Incremental Maintenance. To tests our incremental update algorithms, we extracted five subsets of 5,000 triples from the LUBM-50K dataset; for each subset, we measured the time used to update the materialisation after deletion. On average, RDFox could update the materialisation in 0.49s while removing 8525.8 triples in total; the fastest update took 0.42s and required deleting 8,451 triples, while the longest one took 0.6s and required deleting 8,520 triples.

6 Conclusion

In this paper, we have presented RDFox, a main-memory RDF store that supports parallel datalog reasoning. We have described the system architecture of RDFox together with its numerous APIs, its highly-efficient and flexible storage scheme, and its state-of-the-art datalog reasoning algorithms. Its open-source cross-platform implementation also allows for easy integration in a wide range of Semantic Web application scenarios. With storage capabilities of up-to 9.2 billion triples, datalog reasoning speeds of up-to 6.1 million triples per second, and parallel reasoning speedups of up to 87 times, RDFox opens new possibilities for data intensive applications requiring expressive and highly-scalable reasoning. With memory consumption as low as 36.9 bytes per triple, RDFox is also suitable for smaller-scale applications managing up to hundreds of millions of triples on commodity hardware. RDFox thus provides a unique combination of versatility, rich functionality, high performance, and scalability.

In our future work, we plan to extend the functionality of RDFox and improve its performance in a number of ways. Firstly, we plan to add support to all of SPARQL 1.1, and we are already working on an improved query answering algorithm. Secondly, we plan to add support for named graphs, which are becoming

increasingly popular in Semantic Web applications, as well as support for reasoning with non-monotonic negation. Finally, we are in the process of building a shared-nothing, distributed version of the system, which will allow for the efficient storing, querying, and reasoning with larger datasets using less powerful hardware.

Acknowledgments. We thank Hassan Chafi and Brian Whitney for providing access to the T5 system and their support on hardware and OS questions. This work was funded by the EPSRC projects MaSI³, Score!, and DBOnto, and the FP7 project Optique.

References

1. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. AW (1995)
2. Baader, F., Nipkow, T.: *Term Rewriting and All That*. CUP (1998)
3. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLIM: A family of scalable semantic repositories. *Sem. Web* **2**(1), 33–42 (2011)
4. Chong, E.I., Das, S., Eadon, G., Srinivasan, J.: An efficient SQL-based RDF querying scheme. In: *Proc. VLDB*, pp. 1216–1227 (2005)
5. Flum, J., Frick, M., Grohe, M.: Query Evaluation via Tree-Decompositions. *Journal of the ACM* **49**(6), 716–752 (2002)
6. Grosz, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: *WWW*, pp. 48–57 (2003)
7. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. *JWS* **3**(2–3), 158–182 (2005)
8. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining views incrementally. In: *Proc. SIGMOD*, pp. 157–166 (1993)
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, pp. 57–67 (2006)
10. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member Submission* (2004)
11. Kolovski, V., Wu, Z., Eadon, G.: Optimizing enterprise-scale OWL 2 RL reasoning in a relational database system. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 436–452. Springer, Heidelberg (2010)
12. Manola, F., Miller, E., McBride, B.: *RDF Primer*. W3C Rec. **10**, 1–107 (2004)
13. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: *OWL 2 Web Ontology Language Profiles*, 2nd edn. W3C Rec. (2012)
14. Motik, B., Nenov, Y., Piro, R., Horrocks, I.: Incremental update of datalog materialisation: the backward/forward algorithm. In: *Proc. AAI*, pp. 1560–1568 (2015)
15. Motik, B., Nenov, Y., Piro, R., Horrocks, I., Olteanu, D.: Parallel materialisation of datalog programs in centralised, main-memory RDF systems. In: *Proc. AAI*, pp. 129–137 (2014)
16. Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Combining rewriting and incremental materialisation maintenance for datalog programs with equality. In: *Proc. (to appear 2015)*

17. Motik, B., Nenov, Y., Piro, R.E.F., Horrocks, I.: Handling owl:sameAs via rewriting. In: Proc. AAAI, pp. 231–237 (2015)
18. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al.: OWL 2 web ontology language: Structural specification and functional-style syntax. W3C Rec. **27**, 17 (2009)
19. Neumann, T., Weikum, G.: The RDF-3X Engine for Scalable Management of RDF Data. VLDB Journal **19**(1), 91–113 (2010)
20. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, chap. 7, pp. 371–443. Elsevier Science (2001)
21. SPARQL 1.1 Overview. W3C Recommendation (March 21, 2013)
22. Wu, Z., Eadon, G., Das, S., Chong, E.I., Kolovski, V., Annamalai, M., Srinivasan, J.: Implementing an inference engine for RDFS/OWL constructs and user-defined rules in oracle. In: ICDE, pp. 1239–1248 (2008)

TR Discover: A Natural Language Interface for Querying and Analyzing Interlinked Datasets

Dezhao Song¹(✉), Frank Schilder¹, Charese Smiley¹, Chris Brew², Tom Zielund¹, Hiroko Bretz¹, Robert Martin³, Chris Dale¹, John Duprey³, Tim Miller⁴, and Johanna Harrison⁵

¹ Research and Development, Thomson Reuters, Eagan, MN 55123, USA
{dezhao.song, frank.schilder, charese.smiley, chris.brew, thomas.zielund, hiroko.bretz, robertd.martin, chris.dale, john.duprey, tim.miller, johanna.harrison}@thomsonreuters.com

² Research and Development, Thomson Reuters, London, UK

³ Research and Development, Thomson Reuters, Rochester, NY 14694, USA

⁴ Intellectual Property and Science, Thomson Reuters, London, UK

⁵ Intellectual Property and Science, Thomson Reuters, Philadelphia, PA 19130, USA

Abstract. Currently, the dominant technology for providing non-technical users with access to Linked Data is keyword-based search. This is problematic because keywords are often inadequate as a means for expressing user intent. In addition, while a structured query language can provide convenient access to the information needed by advanced analytics, unstructured keyword-based search cannot meet this extremely common need. This makes it harder than necessary for non-technical users to generate analytics. We address these difficulties by developing a natural language-based system that allows non-technical users to create well-formed questions. Our system, called TR Discover, maps from a fragment of English into an intermediate First Order Logic representation, which is in turn mapped into SPARQL or SQL. The mapping from natural language to logic makes crucial use of a feature-based grammar with full formal semantics. The fragment of English covered by the natural language grammar is domain specific and tuned to the kinds of questions that the system can handle. Because users will not necessarily know what the coverage of the system is, TR Discover offers a novel auto-suggest mechanism that can help users to construct well-formed and useful natural language questions. TR Discover was developed for future use with Thomson Reuters Cortellis, which is an existing product built on top of a linked data system targeting the pharmaceutical domain. Currently, users access it via a keyword-based query interface. We report results and performance measures for TR Discover on Cortellis, and in addition, to demonstrate the portability of the system, on the QALD-4 dataset, which is associated with a public shared task. We show that the system is usable and portable, and report on the relative performance of queries using SQL and SPARQL back ends.

Keywords: Natural language interface · Question answering · Feature-based grammar · Auto-suggestion · Analytics

1 Introduction

Organizations adopt Linked Data because they want to provide information professionals with seamless access to all the relevant data that is present in the organization, irrespective of its arrangement into database tables and its actual physical location. Many organizations now have effective strategies for ensuring that there are well-designed links between related records in different tables. Technical professionals, such as database experts and data scientists, will use a mix of traditional and novel database query languages to access this information. But non-technical information professionals, such as journalists and patent lawyers, who cannot be expected to learn a database query language, still need a fast and effective means for accessing the data that is relevant to the task at hand.

Keyword-based search allows non-technical users to access large-scale linked data, and it can be applied in a uniform fashion to information sources that may have wildly divergent logical and physical structure. But it does not always allow precise specification of the user’s intent, so the result sets that are returned may be unmanageably large and of limited relevance. If non-technical users could write good SPARQL queries, they would get smaller and more relevant result sets. In addition, because database query languages impose structure on the result set, they can readily be used to provide dynamically generated analytics. This is not so easy to do when the results are less structured, as they will be when they come from keyword-based search.

Our system, *TR Discover*, is designed to bridge the gap between keyword-based search and structured query. In our system, the user creates natural language questions, which are mapped into a logic-based intermediate language. A grammar defines the options available to the user and implements the mapping from English into logic. An auto-suggest mechanism guides the user towards questions that are both logically well-formed and likely to elicit useful answers from the available databases. A second translation step then maps from the logic-based representation into a standard query language such as SPARQL or SQL, allowing the database query to rely on robust existing technology. Since all professionals can use natural language, we retain the accessibility advantages of keyword-based search, and since the mapping from the logical formalism to the query language is information-preserving, we retain the precision of query-based information access. We also retain the ability to generate useful structured analytics.

*TR Discover*¹ is composed of the following components: Web User Interface, Auto-suggest, Question Understanding, FOL Parsing and Translation, Query Execution, and Analytics Generation. We present the details for each of the components in the rest of the paper: Section 2 describes use cases for *TR Discover* with Cortellis. We then formally present the different components of *TR Discover* in Sections 3 to 5, and evaluate our system in Section 6. We discuss related work in Section 7 and conclude in Section 8.

¹ Beta version available at: <http://cortellislabs.com> (free sign-up)

2 Use Cases

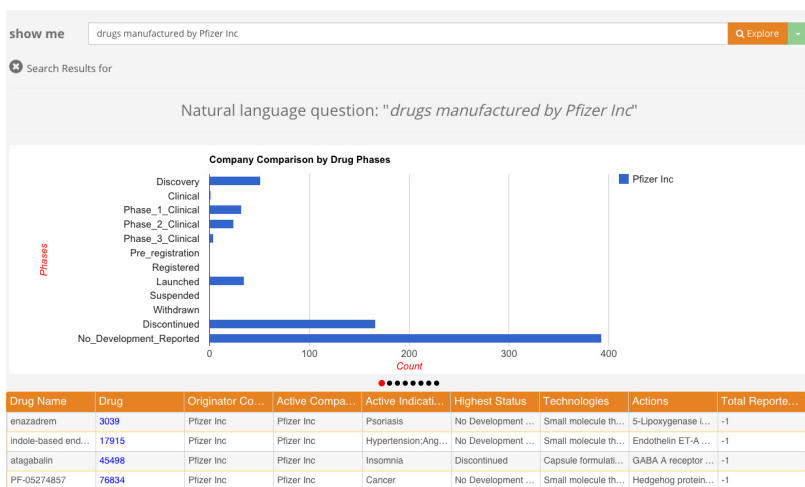
In this section, we present use cases of *TR Discover*, targeting different types of users. We describe the use cases in the context of Thomson Reuters Cortellis² (Cortellis). Cortellis is a data integration and search platform developed for professional users in the Pharmaceutical industry. It relies on a linked dataset that covers a variety of domains, including Life Sciences, Intellectual Property, Legal and Finance. A keyword-based query interface allows users to obtain information relevant to a wide range of tasks, such as drug repurposing, target finding, legal research about a specific drug, or search for patents owned by a particular company.

| | | |
|--|----------------------------------|-------------------------------|
| d | drugs | drugs manufactured by |
| NL | NL | NL |
| drugs | using | companies |
| drugs using | having a secondary indication of | company |
| drugs having a secondary indication of | having a primary indication of | Pfizer Inc |
| drugs having a primary indication of | developed by | National Institutes of Health |
| | manufactured by | GlaxoSmithKline plc |

(a) “d” is typed

(b) “drugs” is selected and suggestions are provided

(c) “manufactured by” is picked and “Pfizer Inc” can be chosen to complete a question



(d) Query Results and Analytics

Fig. 1. Use Case: First-time Users of TR Discover

² <http://thomsonreuters.com/en/products-services/pharma-life-sciences/pharma-business-development/cortellis-data-fusion.html>

The second use case targets expert professional users (e.g., medical professionals, financial analysts, or patent officers). This user, User B, understands the domain, and has specific questions in mind that may require material from multiple slices of data. She need not be concerned with how the data is partitioned across database tables because she is sheltered from this level of implementation detail. Suppose User B works for a pharmaceutical company and is interested in searching for patents relevant to a particular line of drug development. Guided by our structured auto-suggest, she could pose the detailed question, *patents filed by companies developing drugs targeting PDE 4 inhibitor using Small molecule therapeutic that have already been launched*. Our system returns 12 patents for this question and from the generated analytics (Figure 2), she can immediately see a general view of the competitive field. User B can then drill further into the patents, and begin to develop a strategy that navigates around potential infringements of her competitors’ protected rights, for example.

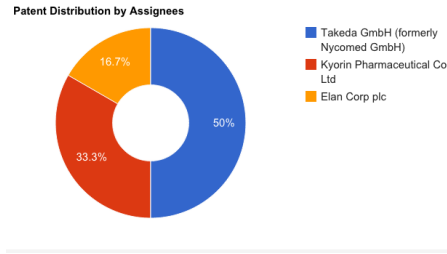


Fig. 2. Analytics for Complex Question from Professional Users

Our first use case targets first-time users of *TR Discover* or users with limited knowledge of the underlying data. This user, User A, may be interested in broad, exploratory questions; however, due to lack of familiarity with the data, guidance (from our auto-suggest module, Section 3.2) will be needed to help him build a valid question in order to explore the underlying data. Figures 1(a)-1(c) demonstrate this question building process. Assuming that User A starts by typing in *d*, *drugs* will then appear as a possible completion. He can either continue typing *drugs* or select it from the drop down list on the user interface. Upon selection, suggested continuations to the current question segment, such as *using* and *manufactured by*, are then provided to User A. Suppose our user is interested in exploring drug manufacturers and thus selects *manufactured by*. In this case, both the generic type, *companies*, along with specific company instances like *Pfizer Inc* and *Glaxo Smith Kline Plc* are offered as suggestions. User A can then select *Pfizer Inc* to build the valid question, *drugs manufactured by Pfizer Inc* thereby retrieving answers from our data stores along with the corresponding analytics (Figure 1(d)).

3 Question Understanding

3.1 Natural Language Question Parsing

In *TR Discover*, we use a feature-based context-free grammar (FCFG) for parsing natural language questions. Our FCFG consists of phrase structure rules on non-terminal nodes and lexical entries for leaf nodes. The large majority of the phrase structure rules are domain independent allowing the grammar to be portable to new domains. The following shows a few examples of our grammar rules: *G1 - G3*. Specifically, Rule *G3* indicates that a verb phrase (*VP*) contains a verb (*V*) and a noun phrase (*NP*).

G1: NP \rightarrow N
 G2: NP \rightarrow NP VP
 G3: VP \rightarrow V NP
 Lex1: N[TYPE=drug, NUM=pl, SEM=< $\lambda x.drug(x)$ >] \rightarrow ‘drugs’
 Lex2: V[TYPE=[drug,org,dev], SEM=< $\lambda X x.X(\lambda y.dev_org_drug(y,x))$ >, TNS=past, NUM=?n] \rightarrow ‘developed by’ /*In general, TYPE=[subject_constraint, object_constraint, predicate_name]*/
 Lex3: V[TYPE=[org,country,hq], NUM=?n] \rightarrow ‘headquartered in’

Each entry in the FCFG lexicon contains a variety of domain-specific features that are used to constrain the number of parses computed by the parser preferably to a single, unambiguous parse. *Lex1-Lex3* are examples of lexical entries. For instance, *Lex1* is the lexical entry for the word, *drugs*, indicating that it is of TYPE *drug*, is plural, and has the semantic representation $\lambda x.drug(x)$. Verbs (V) have an additional feature tense (TNS), as shown in *Lex2*. The TYPE of verbs specify both the potential subject-TYPE and object-TYPE, which can be used to filter out nonsensical questions like *drugs headquartered in the U.S.*

Disambiguation relies on the unification of features on non-terminal syntactic nodes. We mark prepositional phrases (PPs) with features that determine their attachment preference. For example, we specify that the prepositional phrase *for pain* must attach to an NP rather than a VP; thus, in the question *Which companies develop drugs for pain?*, *for pain* cannot attach to *develop* but must attach to *drugs*. Additional features constrain the TYPE of the nominal head of the PP and the semantic relationship that the PP must have with the phrase to which it attaches. This approach filters out many of the syntactically possible but undesirable PP-attachments in long queries with multiple modifiers, such as *companies headquartered in Germany developing drugs for pain or cancer*. In rare instances when a natural language question has multiple parses, we always choose the first parse. Future work may include developing ranking mechanisms in order to rank the parses of a question.

3.2 Enabling Question Completion with Auto-suggestion

Traditional question answering systems often require users to enter a complete question. However, often times, it may be difficult for novice users to do so, e.g., due to the lack of familiarity and an incomplete understanding of the underlying data. One unique feature of *TR Discover* is that it provides suggestions in order

to help users to complete their questions. The intuition here is that our auto-suggest module guides users in exploring the underlying data and completing a question that can be potentially answered with the data. Unlike Google’s query auto-completion that is based on query logs [3], our suggestions are computed based upon the relationships and entities in the dataset and by utilizing the linguistic constraints encoded in our grammar.

Our auto-suggest module is based on the idea of left-corner parsing. Given a query segment qs (e.g., *drugs, developed by*, etc.), we find all grammar rules whose left corner fe on the right side matches the left side of the lexical entry of qs . We then find all leaf nodes in the grammar that can be reached by using the adjacent element of fe . For all reachable leaf nodes (i.e., lexical entries in our grammar), if a lexical entry also satisfies all the linguistic constraints, we then treat it as a valid suggestion.

We rank the suggestions based upon statistics extracted from an RDF graph. Each node in the RDF graph represents a lexical entry (i.e., a potential suggestion), including entities (e.g., specific drugs, drug targets, diseases, companies, and patents), predicates (e.g., *developed by* and *filed by*), and generic types (e.g., Drug, Company, Technology, etc.). The ‘popularity’ (i.e., ranking score) of a node is defined as the number of relationships it is involved in. For example, if a company filed 10 patents and is also involved in 20 lawsuits, then its popularity will be 30. Our current ranking is computed based only upon the data; in future work, it may be possible to tune the system’s behavior to a particular individual user by mining our query logs for similar queries previously made by that user.

There are (at least) two ways of using the auto-suggest facility. Users can work in steps: they could type in an initial question segment, like *patents*, and wait for the system to provide suggestions. Then, users can select one of the suggestions to move forward. By repeating this process, users can build well-formed natural language questions (i.e., questions that are likely to be understood by our system) in a series of small steps guided by our auto-suggest. Alternatively, users can type in a longer string, without pausing, and our system will chunk the question and try to provide suggestions for users to further complete their question. For instance, given the following partial question *drugs developed by Merck using ...*, our system first tokenizes this question; then starting from the first token, it finds the shortest phrase (a series of continuous tokens) that matches a suggestion and treats this phrase as a question segment. In this example, *drugs* will be the first segment. As the query generation proceeds, our system finds suggestions based on the discovered query segments, and produces the following sequence of segments: *drugs, developed by, Merck, and using*. At the end, the system knows that *using* is likely to be followed by a phrase describing a drug technology, and is able to offer corresponding suggestions to the user. In general, an experienced user might simply type in *drugs developed by Merck using*; while first-time users who are less familiar with the data can begin with the stepwise approach, progressing to a more fluent user experience as they gain a deeper understanding of the underlying data.

4 Question Translation and Execution

In contrast to other Natural Language Interfaces (NLI) [11,18], *TR Discover* first parses a natural language question into a First Order Logic (FOL) representation. The FOL representation of a natural language question is further translated to other executable queries (e.g., SPARQL and SQL). This intermediate logical representation provides us the flexibility to develop different query translators for various types of data stores.

The process of translating FOL to SPARQL/SQL can be divided into two steps. In the first step, we parse the FOL representation into a parse tree according to an FOL parser. This FOL parser is implemented with ANTLR³ (a parser development tool). The FOL parser takes a grammar and an FOL representation as input, and generates a parse tree for the FOL representation. Figure 3 shows the parse tree of the FOL representation for the question “Drugs developed by Merck”.

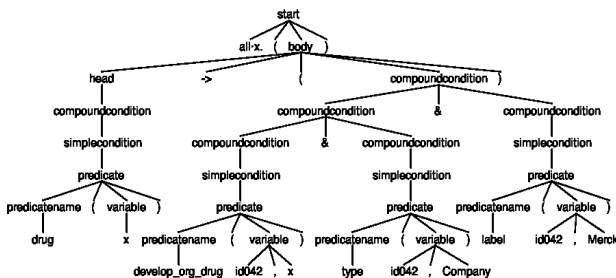


Fig. 3. The Parse Tree for the FOL of the Question “Drugs developed by Merck”

We then perform an in-order traversal (with ANTLR’s APIs) of the FOL parse tree and translate it to executable queries. While traversing the tree, we put all the atomic logical conditions and the logical connectors into a stack. When we finish traversing the entire tree, we pop the conditions out of the stack to build the correct query constraints; predicates in the FOL are also mapped to their corresponding attribute names (SQL) or ontology properties (SPARQL).

The following summarizes the translation from a natural language question to a SQL and SPARQL query via a FOL representation:

```
Natural Language Question: Drugs developed by Merck
FOL: all x.(drug(x) → (develop_org_drug(id0,x) & type(id0,Company) & label(id0,Merck)))
SQL Query: select drug.* from drug where drug.originator_company_name = 'Merck'
SPARQL Query (prefixes are omitted):
  select ?x
  where {
    ?id0 rdfs:label 'Merck'.
    ?id0 rdf:type example:Company .
```

³ <http://www.antlr.org/>

```

?x rdf:type example:Drug .
?id0 example:develops ?x .
}

```

We execute the translated SQL queries using SQLite, a light weight relational database management system that allows us to store the entire database as a single file on disk. We run the SPARQL queries in a Jena TDB triple store⁴. An additional query is issued for SPARQL queries to retrieve all attribute values of the entities in the result set.

5 Analytics Generation

This section details two different kinds of analytics that are generated for the result set for a given question. We developed these analytics for the Cortellis dataset (Section 2) and future work will include generating more analytics based on the content type and the questions derived from different use cases.

Descriptive and Comparative Analytics. Query results are analyzed according to the counts of the results, as shown in Figure 2. In addition to the counts of the result set, descriptive analytics are also shown across different dimensions (e.g., indication, technology, action, etc.) for drugs. Moreover, result sets can be compared across these dimensions via related entities, such as companies. Figure 4 demonstrates a concrete example of comparative analytics. Here, companies in the result set are compared against various phases of the drug development. This chart shows that *Signature Therapeutics* has the most drugs in the Discovery phase even though *Pfizer* has the most drugs across all phases.

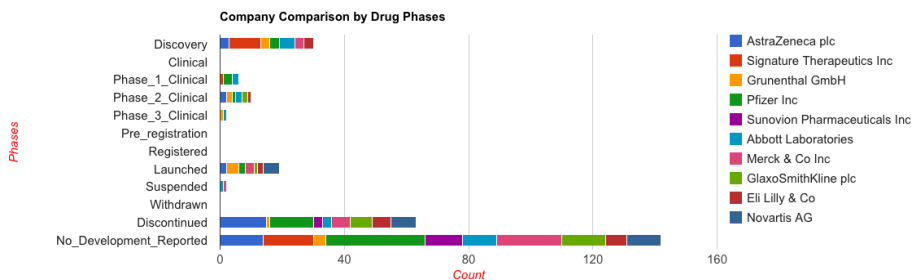


Fig. 4. Comparing companies on the dimension of drug development phase for “Drugs having a primary indication of pain”

Content Analysis. We also developed content-based analytics. To support this type of analytics, we applied named entity recognition (NER) and sentiment analysis. For NER, we used the Stanford CoreNLP toolkit [12] to recognize person, organization, and locations from the Reuters News Archive (RNA). There

⁴ <https://jena.apache.org>

are about 14 million documents and 147 million sentences in the entire RNA dataset and the named entity recognition is done in a distributed environment using Apache Spark. The entire process took roughly 48 hours and discovered about 280 million entities. As a second step, we ran an open-source sentiment

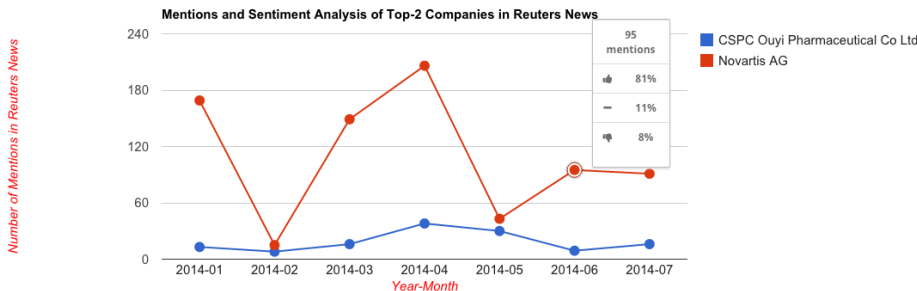


Fig. 5. NER and Sentiment Analysis Results for Top-2 Companies in Question “companies developing drugs having a primary indication of hypertension”

analysis tool over the entire corpus [14]. Given an entity from the outcome of the NER process, we retrieve documents from RNA that contain this entity. For each document, we then find all sentences that contain this entity and perform sentiment analysis on each of the sentences. The outcome for each sentence could be: Positive, Neutral, or Negative. Finally, we determine the overall sentiment of this document on the given entity by majority vote on the sentence-level results.

Figure 5 demonstrates our sentiment analysis results for the question *Companies developing drugs having an indication of Hypertension*. Here, we pick the top two companies that are most frequently mentioned in RNA in order to avoid overwhelming users with a dense chart. When the mouse hovers over a data point, the system displays the sentiment results. In this example, we can see that in June 2014, there are 95 news articles that mention “Novartis AG”, with 81% of such documents having a positive sentiment towards the company.

6 Evaluation

In this section, we present the evaluation results of *TR Discover*. First, we evaluate the response time of the different components of *TR Discover*. Then, we apply our proposed system to Task 2 (Biomedical question answering over inter-linked data) of QALD-4 [19] to compare its precision and recall to state-of-the-art question answering systems in the biomedical domain.

6.1 Runtime Evaluation

Dataset. We evaluate the runtime of the different components of *TR Discover* on the Cortellis dataset, which consists of about 1.2 million entities. Our dataset is actually integrated from three sources: Cortellis drug data, a Thomson Reuters patent dataset, and DrugBank. Using string matching, we linked Cortellis’ companies to patent assignees, and the drugs names between Cortellis and DrugBank. Thus, complementary information from different sources can be presented to users. The different entity types include *Drug*, *Drug_Target*, *Company*, *Technology*, *Patent*, etc. Various types of relationships exist between the entities, including *Using* (Drug uses Technology), *Developing* (Company develops Drug), *Headquartered in* (Company headquartered in Country), etc.

Since we can translate the logical representation of a natural language question to both SPARQL and SQL, we prepared two different data stores for our dataset. We store the Cortellis dataset into a relational database using SQLite; and, in order to be able to run SPARQL queries, we convert the relational data to triples and store them into a Jena TDB triple store. Take the above examples again: *Drug*, *Drug_Target*, *Company*, *Technology*, and *Patent* all become classes, while *Using*, *Developing*, and *Headquartered in* become predicates in our RDF data. This data transformation process produces about 12 million triples.

Random Question Generation. In order to evaluate the runtime of our proposed system, we randomly generated a total number of 5,000 natural language questions using our auto-suggest component (Section 3.2). Recall that our auto-suggest module provides suggestions as potential next steps in order to help users to complete their questions, thus making it also useful for generating random testing questions. We give the auto-suggest module a starting point, e.g. *drugs*, and then perform a depth-first search to uncover all possible questions. At each depth, for each question segment, we randomly select b suggestions; we then continue this search process with each of the b suggestions. By setting different depth limits, we generate questions with different levels of complexity (or different number of verbs). Using this random question generation process, we generated 1,000 natural language questions for each number of verbs from 1 to 5, thus 5,000 questions in total.

Runtime Results. We evaluate on a 16-core RedHat machine with 2.90GHz CPU and 264GB of memory. Figure 6(a) shows the parsing time of natural language questions. Although we adopt NLTK⁵ for parsing natural language questions in our system (by supplying NLTK with our own grammar and lexicon), this evaluation is to show the practicability of the overall approach in potential real-world scenarios. According to Figure 6(a), unless a question becomes truly complicated (with 4 or 5 verbs), the parsing time is generally under 1 second. One example question with 5 verbs could be *Patents granted to companies headquartered in Australia developing drugs targeting Lectin mannose binding protein modulator using Absorption enhancer transdermal*. Experts on the Cortellis

⁵ <http://www.nltk.org/>

Team assure us that questions with more than 5 verbs are rare, thus we did not evaluate questions beyond this level of complexity. Although the parsing time increases as a question becomes more complicated, we did not observe an exponential increase in our experiments.

Figure 6(b) shows the runtime for translating the FOL of a natural language question to SPARQL and SQL queries. In general, for both SPARQL and SQL, the translation time increases as the questions become more complicated, as the FOL translation module needs to traverse bigger FOL parse trees. However, in general, only a few milliseconds are needed for performing each translation, it should not add much burden to the overall runtime of our system.

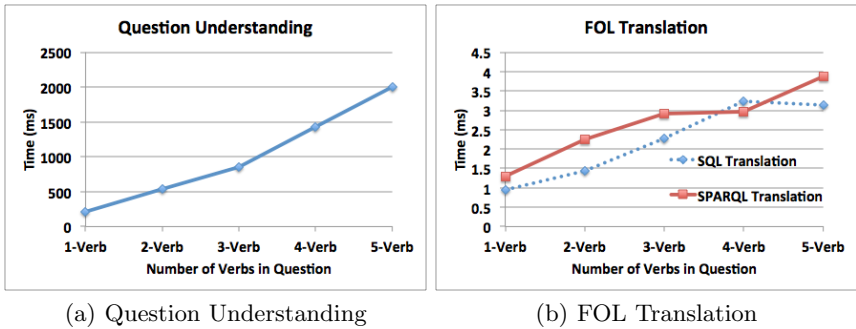


Fig. 6. Runtime Evaluation: Question Understanding and FOL Translation

Finally, we demonstrate the query execution time of both SPARQL and SQL in Figure 7. Generally speaking, SQL queries run faster than SPARQL queries

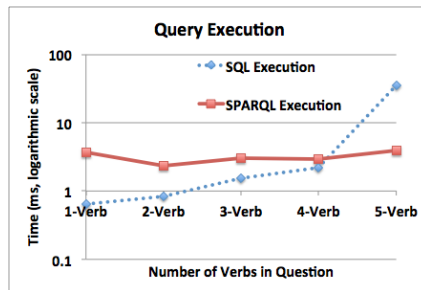


Fig. 7. Runtime Evaluation: SQL and SPARQL Query Execution

(for questions with up to 4 verbs). However, for really complicated questions (i.e., those with 5 verbs), SQL queries took much longer to finish than SPARQL queries did. One potential reason is that many joins are usually performed for

the SQL query of 5-verb questions, which could greatly slow down the query execution process. Consider the above 5-verb question again, its SQL Translation actually contains 6 joins between different tables.

6.2 Evaluation on the QALD-4 Benchmark

We also evaluate *TR Discover* on Task 2 (i.e. Biomedical question answering over interlinked data) of QALD-4 [19], using the FOL to SPARQL translation.

Dataset. The QALD-4 competition provides an RDF dataset, training and testing questions, and ground truth answers to the questions⁶. We loaded the data into a Jena TDB triple store. We also materialized the triples based upon *owl:sameAs* statements. For example, given explicit triples: *a owl:sameAs b. b hasPossibleDrug c.*, we then add the following triple *a hasPossibleDrug c* into our triple store. By adding such additional triples, we then do not have to explicitly add *owl:sameAs* triple patterns when performing SPARQL translation. We use the competition’s online evaluation tool to calculate the precision and recall. Our evaluation did not involve any human evaluators.

Evaluation Results. In Table 1, for the training questions, *TR Discover* was able to achieve a comparable recall to the state-of-the-art systems but with a much lower precision. The main reason for having low precision is that we added the materialized triples according to the *owl:sameAs* statements in the original dataset. Many of the returned results of *TR Discover* are actually correct, but they are not in the provided ground truth. They are linked (either explicitly or implicitly) to the instances in the ground truth via *owl:sameAs* statements. In order to report on a more realistic precision number, we augmented the ground truth by adding the equivalent instances to those already in the ground truth and achieved higher performance: 80% of precision and 92% of recall, i.e., *TR Discover+*. Similar to the training set, *TR Discover+* also achieves a much better precision by using the augmented ground truth on the testing set. For *TR Discover+*, we implemented an evaluation tool in Java; our evaluation tool produces the exact results for *TR Discover* as the online evaluation tool does.

Table 1. Evaluation Results on Task 2 of the QALD-4 Benchmark

| Dataset | System | Precision | Recall | F1 | Dataset | System | Precision | Recall | F1 |
|----------|---------------------|-------------|-------------|-------------|---------|---------------------|-------------|-------------|-------------|
| Training | <i>TR Discover</i> | 0.44 | 0.88 | 0.58 | Testing | <i>TR Discover</i> | 0.34 | 0.80 | 0.48 |
| | <i>TR Discover+</i> | 0.80 | 0.92 | 0.85 | | <i>TR Discover+</i> | 0.75 | 0.84 | 0.79 |
| | <i>GFMed</i> [13] | N/A | N/A | N/A | | <i>GFMed</i> [13] | 1.00 | 0.99 | 0.99 |
| | <i>POMELO</i> [8] | 0.83 | 0.87 | 0.85 | | <i>POMELO</i> [8] | 0.82 | 0.87 | 0.85 |
| | <i>RO-FII</i> [19] | N/A | N/A | N/A | | <i>RO-FII</i> [19] | 0.16 | 0.16 | 0.16 |

* We did not find the publication of *RO-FII*. Please refer to the QALD-4 paper for details.

In Table 1, we employ fuzzy string matching on literal values. Here, we also study the impact of adopting exact string matching on the performance of our system. In Table 2, by employing fuzzy matching, we achieve higher recall.

⁶ <http://greentackle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=task2&q=4>

Table 2. Fuzzy String Matching (Fuzzy) vs. Exact String Matching (Exact) on Literals

| Dataset | System | Precision | Recall | F1 | Time (s) | Dataset | System | Precision | Recall | F1 | Time (s) |
|----------|--------|-------------|-------------|-------------|----------|---------|--------|-------------|-------------|-------------|----------|
| Training | Exact | 0.87 | 0.88 | 0.87 | 7 | Testing | Exact | 0.55 | 0.67 | 0.60 | 3 |
| | Fuzzy | 0.80 | 0.92 | 0.85 | 50 | | Fuzzy | 0.75 | 0.84 | 0.79 | 20 |

Although fuzzy matching results in lower precision on the training questions, the overall F1-score was only slightly impacted. Different from the training questions, fuzzy matching leads to a much better precision and recall on the testing questions, because the identified entities and literal values in the testing questions often times do not match exactly with the underlying data. We also measure the runtime of the two matching approaches, and it is natural to observe that fuzzy matching requires a much longer time to perform the translated SPARQL queries. When running queries on even larger datasets, the trade-off between precision, recall, and runtime needs to be taken into account.

6.3 Discussion

We presented *TR Discover* that can be used by non-technical professionals to explore complex interlinked datasets. *TR Discover* relies on a feature-based context free grammar for question parsing. The grammar represents about 2 months of design and experimentation, with approximately 60 grammar rules and 1 million lexical entries. The lexical entries are automatically created using the attribute values in our database, and only the grammar rules are handcrafted for question types used in *TR Discover*. Our grammar covers conjunctions (*and/or*), noun phrases with optional quantifiers (*all* and *some*), nominal and adjectival modifiers, and verbal constructions. Given a new domain, the core grammar remains stable and new lexical entries can be automatically added. The adaptation to QALD-4 took roughly 30 person hours, including adding the types (e.g., enzymes, side-effects) and their relationships (e.g., associatedWith(gene, gene)), and grammar rules to cover syntactic differences in the QALD-4 questions.

Furthermore, we perform some error analyses to study the questions that *TR Discover+* failed to answer properly. Our system has very low precision for Training Question 5: “Which genes are associated with breast cancer?”. Our system finds two *Disease* instances with the exact *rdfs:label* “breast cancer” and returns their associated genes. However, the gold SPARQL query uses another *Disease* instance <http://www4.wiwiss.fu-berlin.de/diseasome/resource/diseases/1669> with *rdfs:label* “Breast cancer-1” and returns its associated genes. Consequently, our system gives many false positives to this question.

Our system also has difficulties understanding Training Question 14: “Give me drug references of drugs targeting Prothrombin”. Our system interprets two predicates from this question as follows: *drug_reference* between *drug* and *drug_reference*, and *target* between *drug* and *target*. However, the gold SPARQL query indicates that the *drug_reference* predicate is actually between *target* and *drug_reference*. For such questions, additional domain expertise may be required in order for us to build our question understanding module to achieve more

accurate parsing. A similar situation also applies to Testing Question 21: “Give me the drug categories of Desoxy”. Through our natural language question understanding process, we interpret one predicate from this question: *brandName* between *drug_category* and the literal value “Desoxy”; however, the gold SPARQL query indicates that *drug_category* should be an object property rather than a class. This misinterpretation results in an empty result set for this question.

We also notice a potential error in the ground truth for Training Question 19: “Which are the drugs whose side effects are associated with the gene TRPM6?”, which looks for drugs that satisfy certain constraints. However, in the ground truth, a *Disease* instance is given as the answer. The ground truth SPARQL query also looks for diseases rather than drugs, thus we think there might be an error in the ground truth for this question.

One limitation of *TR Discover* is that it lacks the ability to translate natural language questions with quantifiers, e.g., Testing Question 3: “which drug has the highest number of side effects?” Testing Question 14 and 23 are also of this category. Also, negation is only supported with our SPARQL translation.

7 Related Work

Keyword search [4, 6, 17] and faceted search [7, 23] have been frequently adopted for retrieving information from knowledge bases (KB). However, users may have to figure out the most effective queries in order to retrieve relevant information. Furthermore, without appropriate ranking methods, users may be overwhelmed by the information available in the search results. In contrast, our system allows users to ask questions in natural language format, which enables users to express their search requests in a more intuitive way.

Much of the prior work on question answering over linked data parses a natural language question with various NLP techniques, maps the identified entities, concepts and relationships to instances, classes and properties in an ontology to construct a SPARQL query, and retrieves answers from a triple store [9, 11, 15, 18, 21]. In addition to adopting fully automatic query parsing, CrowdQ also incorporates crowd sourcing techniques for understanding natural language questions [5]. Instead of only using structured data, HAWK [20] utilizes both structured and unstructured data for answering natural language questions.

Significant efforts have also been devoted to developing question answering systems in the biomedical research field [1]. Questions are first analyzed to identify entities (e.g., person, location, disease, gene, etc.) and relationships. The identified entities are then mapped to concepts in a taxonomy/ontology or linguistic resources for query expansion [10, 22]. An information retrieval query is finally issued to an index to find matching documents, which are then ranked and summarized to produce the final answer.

Compared to state-of-the-art systems, in this work, we maintain flexibility by first parsing a question into First Order Logic, which is further translated into both SPARQL and SQL. Using FOL allows us to be agnostic to which query language (e.g., SQL and SPARQL) will be used later. We do not incorporate

any query language statements directly into the grammar, keeping our grammar leaner and more flexible for adapting to other query languages. Furthermore, one distinct feature of our system is that it helps users to build a complete question by providing suggestions according to a partial question and a grammar. For example, when users type in *drugs*, our system will suggest: *developed by*, *having a primary indication of*, etc., since these are the options to form valid questions according to our grammar. Finally, to the best of our knowledge, none of existing NLI provide dynamic analytics for the results. Given different types of entities and their dimensions, our system performs descriptive analytics and comparisons on various dimensions of the data, and conducts sentiment analysis. Such analytics would support users to better conduct further analyses and derive insights from the data. Although ORAKEL [2] also maps a natural language question to a logical representation, there is no auto-suggest and analytics provided to the users. Compared to our prior work [16], in this paper, we provide a more fluent user experience of auto-suggest, and we perform a thorough evaluation of our system by examining the runtime of its various components and by comparing to state-of-the-art systems on the QALD-4 Benchmark.

8 Conclusion and Future Work

In this paper, we propose *TR Discover*, a natural language question answering system over interlinked datasets. *TR Discover* was designed with non-technical information professionals in mind in order to allow them fast and effective access to large-scale interlinked datasets. Going beyond keyword-based search, *TR Discover* produces precise result sets and generates analytics for natural language questions asked by information professionals, such as journalists or patent lawyers. Rather than asking users to provide an entire question on their own, *TR Discover* provides suggestions (i.e., auto-suggest) in order to facilitate this question building process. Given a completed natural language question, *TR Discover* first parses it into its First Order Logic (FOL) representation, by using a feature-based grammar with full formal semantics derived from interlinked datasets. By further translating the FOL representation of a natural language question into different executable queries (SPARQL and SQL), our system retrieves answers from the underlying data stores and generates corresponding analytics for the results. Through our runtime evaluation, we show that the overall response time of *TR Discover* is generally acceptable (< 2 seconds). *TR Discover* also achieves comparable precision and recall to that of state-of-the-art question answering systems on the QALD-4 benchmark. In future work, we plan to develop personalized auto-suggestion by using user query logs, and apply *TR Discover* on more and larger datasets to examine the response time of its various components. Furthermore, it would be interesting to seek feedback from real users on the performance and usability of our system. Finally, we plan to better handle synonyms, e.g., “medicines” for “drugs”.

References

1. Athenikos, S.J., Han, H.: Biomedical question answering: A survey. *Comput. Methods Prog. Biomed.* **99**(1), 1–24 (2010)
2. Cimiano, P., Haase, P., Heizmann, J., Mantel, M., Studer, R.: Towards portable natural language interfaces to knowledge bases - the case of the ORAKEL system. *Data Knowl. Eng.* **65**(2), 325–354 (2008)
3. Cornea, R.C., Weininger, N.B.: Providing autocomplete suggestions (February 4, 2014). US Patent 8,645,825
4. d’Aquin, M., Motta, E.: Watson, more than a semantic web search engine. *Semantic Web Journal* **2**(1), 55–63 (2011)
5. Demartini, G., Trushkowsky, B., Kraska, T., Franklin, M.J.: CrowdQ: crowd-sourced query understanding. In: *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research* (2013)
6. 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: *Proceedings of the 2004 ACM International Conference on Information and Knowledge Management*, pp. 652–659 (2004)
7. Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H., Scheel, U.: Faceted wikipedia search. In: Abramowicz, W., Tolksdorf, R. (eds.) *BIS 2010. LNBI*, vol. 47, pp. 1–11. Springer, Heidelberg (2010)
8. Hamon, T., Grabar, N., Mougin, F., Thiessard, F.: Description of the POMELO system for the task 2 of QALD-2014. In: *Working Notes for CLEF 2014 Conference*, pp. 1212–1223 (2014)
9. Lehmann, J., Furche, T., Grasso, G., Ngomo, A.N., Schallhart, C., Sellers, A.J., Unger, C., Bühmann, L., Gerber, D., Höffner, K., Liu, D., Auer, S.: DEQA: deep web extraction for question answering. In: *11th International Semantic Web Conference*, pp. 131–147 (2012)
10. Lin, R.T.K., Chiu, J.L., Dai, H., Tsai, R.T., Day, M., Hsu, W.: A supervised learning approach to biological question answering. *Integrated Computer-Aided Engineering* **16**(3), 271–281 (2009)
11. Lopez, V., Pasin, M., Motta, E.: AquaLog: an ontology-portable question answering system for the semantic web. In: Gómez-Pérez, A., Euzenat, J. (eds.) *ESWC 2005. LNCS*, vol. 3532, pp. 546–562. Springer, Heidelberg (2005)
12. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 55–60 (2014)
13. Marginean, A.: GFMed: Question answering over biomedical linked data with grammatical framework. In: *Working Notes for CLEF 2014 Conference*, pp. 1224–1235 (2014)
14. Narayanan, V., Arora, I., Bhatia, A.: Fast and accurate sentiment classification using an enhanced naive bayes model (2013). *CoRR* abs/1305.6143
15. Shekarpour, S., Ngomo, A.N., Auer, S.: Question answering on interlinked data. In: *22nd International World Wide Web Conference*, pp. 1145–1156 (2013)
16. Song, D., Schilder, F., Smiley, C., Brew, C.: Natural language question answering and analytics for diverse and interlinked datasets. In: *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 101–105 (2015)

17. Tummarello, G., Delbru, R., Oren, E.: *Sindice.com: weaving the open linked data*. In: Aberer, K., et al. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
18. Unger, C., Böhmann, L., Lehmann, J., Ngomo, A.N., Gerber, D., Cimiano, P.: *Template-based question answering over RDF data*. In: *Proceedings of the 21st World Wide Web Conference*, pp. 639–648 (2012)
19. Unger, C., Forascu, C., Lopez, V., Ngomo, A.N., Cabrio, E., Cimiano, P., Walter, S.: *Question answering over linked data (QALD-4)*. In: *Working Notes for CLEF 2014 Conference*, pp. 1172–1180 (2014)
20. Usbeck, R., Ngonga Ngomo, A.C., Böhmann, L., Unger, C.: *HAWK - hybrid question answering over linked data*. In: *12th Extended Semantic Web Conference* (2015)
21. Yahya, M., Berberich, K., Elbassuoni, S., Weikum, G.: *Robust question answering over the web of linked data*. In: *22nd ACM International Conference on Information and Knowledge Management*, pp. 1107–1116 (2013)
22. Yu, H., Cao, Y.G.: *Using the weighted keyword model to improve information retrieval for answering biomedical questions*. *Summit on translational bioinformatics*, p. 143 (2009)
23. Zhang, X., Song, D., Priya, S., Daniels, Z., Reynolds, K., Hefflin, J.: *Exploring linked data with contextual tag clouds*. *Journal of Web Semantics* **24**, 33–39 (2014)

Linked Data

Drug Encyclopedia – Linked Data Application for Physicians

Jakub Kozák^(✉), Martin Nečaský, and Jaroslav Pokorný

Faculty of Mathematics and Physics, Charles University in Prague,
Prague, Czech Republic
{kozak,necasky,pokorny}@ksi.mff.cuni.cz

Abstract. The information about drugs is scattered among various resources and accessing it is hard for end users. In this paper we present an application called Drug Encyclopedia which is built on the top of the data mart represented as Linked Data and enables physicians to search and browse clinically relevant information about medicinal products and drugs. The application has been running for more than a year and has attracted many users. We describe the development driven by requirements, data mart creation, application evaluation and discuss the lessons learned.

Keywords: Linked Data application · Drugs · Health care · RDF · SPARQL

1 Introduction

There is a lot of information about medicinal products and drugs which a physician should know. Although they can be found at many places, it is hard to navigate through the piles of data which might be in different languages, formats, quality, etc. And moreover, the information change quickly in time as new products are launched. Therefore we decided to go to the source and asked physicians about their needs in the area of drug related information and also conducted a little survey among them. The results were not surprising. The physicians would like to have a single source of information where clinically relevant information about drugs and medicinal products would be brought together and made available in a user friendly way.

In this paper, we present a web application for physicians, called *Drug Encyclopedia*¹, which is based on semantic web technologies and enables the end users to search and explore the clinically relevant information about medicinal products and drugs in general. The data architecture and application development were mainly driven by the user requirements. The application was designed to match the criteria of easy usage and also flexible development. This was achieved with the help of semantic web technologies as can be seen further in the paper.

¹ <http://www.lekovaencyklopedie.cz>

The application has been running since the end of 2013, attracted thousands of users and found regular users among physicians.

The paper is structured as follows. In Section 2, we describe the user requirements. Then we show the data architecture in Section 3. The application and its evaluation are presented in Section 4 and Section 5 respectively. Related work is described in Section 6 and finally, lessons learned are given in Section 7.

2 Use Case

Before we started to build the application, we had conducted a survey among physicians in the Czech Republic to gather their information needs in the drug domain. We collected 43 responses from physicians (diabetologists, obstetricians, ophthalmologists, etc.) using a web questionnaire complemented with screen shots demonstrating possible functionality. For more details please refer to [7]. The following functionality was considered by the physicians as highly desirable:

1. For a medicinal product and/or active ingredient show its indication, classification group, contraindication and interactions with other medicinal products and/or active ingredients. Moreover, information about risks of prescribing a medicinal product to a pregnant women is required by obstetricians.
2. For a list of medicinal products show whether they have the same active ingredients or belong to the same classification group or whether there are some interactions among them.
3. For a medicinal product show selected parts of textual documentation of the product, specifically extracted from Summary of Product Characteristics.
4. For an active ingredient show advanced clinical information i.e. pharmacological action, pharmacokinetics etc.
5. Find interactions in a set of medicinal products and/or active ingredients.

The use case is then quite simple – end users (i.e. physicians) want to be able to explore and search available information about drugs and medicinal products (coming from highly heterogeneous data sources) in a single application. The application should be easily extensible with further data sources in the future.

3 Data Architecture

Driven by the functional requirements described in the Section 2, we have selected the following data sources with relevant pieces of information for the application:

- **(CZ-DRUGS) Medicinal products registered in the Czech Republic** – data provided by the State Institute for Drug Control (SIDC) about medicinal products marketed in the Czech Republic – including prices.
- **SPC (Summary of Product Characteristics)** – documents attached to each marketed medicinal product and intended for professionals.

- **MeSH (Medical Subject Heading)** – a reference dictionary for linking other sources. It is partially translated to Czech and many other languages.
- **NDF-RT (National Drug File - Reference Terminology)** – data about indications, contraindications and data about pharmacological effects.
- **DrugBank** – data about interactions and descriptions of active ingredients.
- **ATC Hierarchy (Anatomical Therapeutic Chemical Classification System)** – classification of drugs maintained by WHO.
- **NCI Thesaurus** – direct links to FDA SPL.
- **FDA SPL (FDA Structured Product Labeling)** – pregnancy category.
- **MedDRA (Medical Dictionary for Regulatory Activities)** – adverse event classification dictionary.



Fig. 1. Data architecture - from data sources to data marts

Because of the heterogeneity of the data sources, we chose to work with Resource Description Format (RDF) [8] and Linked Data (LD) principles [1]. RDF is a format for representing data as triples where each real world entity is represented as a resource (URI) and LD principles give recommendation for publishing RDF. RDF data respecting LD principles are simply called LD.

The data architecture we designed to represent and integrate the selected data sources is depicted in Figure 1. It is logically structured into 4 levels described in the rest of this section.

Level 0 (L0) – Level 2 (L2). We have collected the data sources from L0 described above, transformed them into RDF and loaded them into a triple store. Each data set occupies one graph in the triple store at L1. We have used Virtuoso² as a triple store. Then we have applied several strategies for linking the data sets and created a data warehouse represented by data sets and link sets, i.e. L2. The data architecture is shown in the Figure 1 and more details can be found in our previous work [7] and [6]. According to the notation in the Figure 1, our previous work stopped at L2 which represents a LD warehouse of the integrated data sources. The LD warehouse contains about 120 M triples where about 80 M belong to the representation of SPCs.

Level 3 (L3) - Data mart. Because the interlinked data sets are usually large, contain also data irrelevant to a specific application, and it is hard to navigate in them, we introduce Level 3 called *data mart* which should enable developers to create applications more easily. We have borrowed this name from the data

² Virtuoso – <http://virtuoso.openlinksw.com>

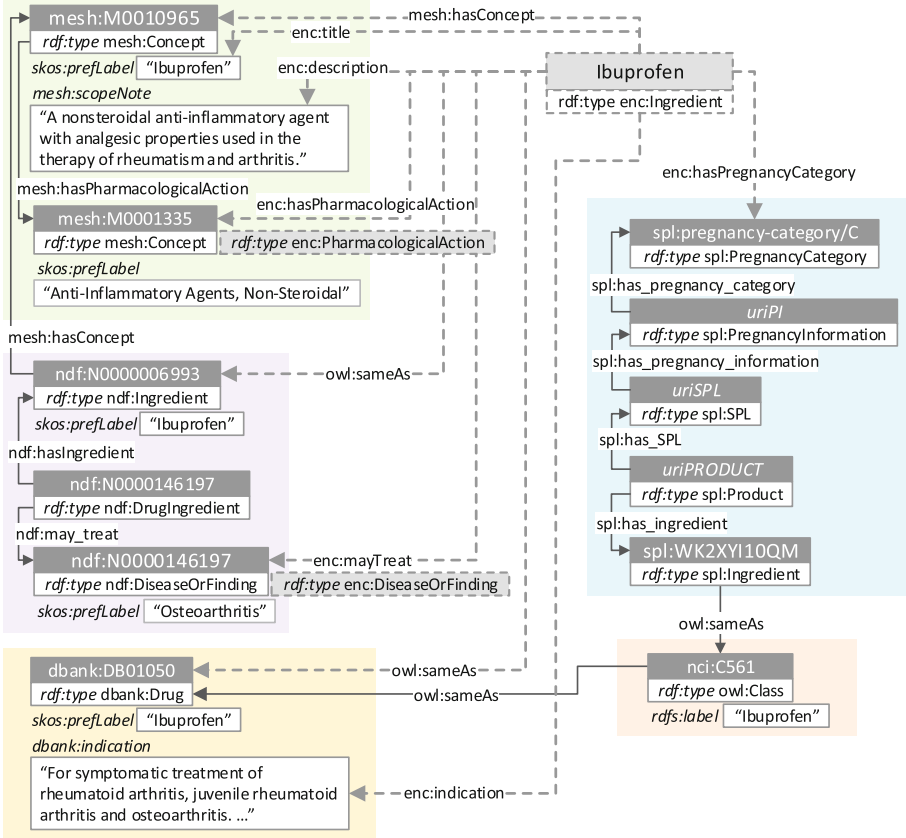


Fig. 2. Part of the triple store. The dashed lines represent the information in the data mart.

warehousing where it means a set of data optimized for specific business or application needs. Here, we have the same need – to make data available in such form that it would be easy and fast to access them. Therefore we make two kinds of optimization – static and dynamic. The *static optimization* simplifies the schema and makes the queries easier to be written. The *dynamic optimization* shortens the paths through the original data sets significantly by adding new triples connecting the resources directly.

To achieve the optimization, we construct the data mart as follows. We add new classes and properties, i.e. we create an application ontology. The new classes are then assigned to the existing resources from L2 in order to distinguish them for the application. The new properties then usually shorten the important paths in the data. Besides that, we also construct new resources which unify underlying resources from different data sets into one. This results into a simpler schema.

All the transformations are represented as SPARQL queries. The schema of the data mart is not derived automatically but rather manually by a domain expert.

We describe the procedure of creating the data mart on the example shown in the Figure 2 where a subset of the data is visualized. Each of the rectangles in the background represents a data set from L1. The dashed lines and rectangles with dashed borders represent new data (triples) from L3 – data mart. The example shows a part of the data for active ingredient *Ibuprofen*.

A new resource was created for the active ingredient *Ibuprofen* which is also linked to the original data sets, e.g. by `owl:sameAs` property. New types are assigned to the important resources from L2, i.e. pharmacological action and disease or finding. These entities are linked to the new active ingredient with properties from the application ontology, e.g., `enc:hasPharmacologicalAction`. This significantly simplifies the schema. When we take a look at the pregnancy category, it is originally available only by traversing several data sets with a long path. We optimize the schema (i.e. also the queries) by connecting the active ingredient directly to the pregnancy category by `enc:hasPregnancyCategory`.

The data mart simplifies the queries and the whole logic. On the other hand, it takes some storage space. Our data mart added 8 M triples to the triple store. The benefits of the data mart are also shown in the Section 4 and the Section 7.

Automation. It is necessary to update the data from the data sources regularly. Most of the data transformation is now done automatically with only minimal manual work needed. The data transformation has been done in *UnifiedViews*³, an extract-transform-load (ETL) tool for RDF [5]. If the data changes in the original data sources, no manual inference is necessary. If the schema changes, the ETL pipelines have to be adjusted.

4 Drug Encyclopedia – The Application

Drug Encyclopedia is a publicly available web application which enables users to search and explore data about medicinal products and drugs from different data sources. The user can search for a required medical resource (medicinal product, active ingredient, indications, pharmacological actions, etc.) and view its detail. The detail shows information about the resource available in the application data mart. The application is not bound to specific data structures – it is able to show any data associated with the resource in the data mart. (Technically, it is able to show any surrounding of the resource in the RDF representation). From the detail page, the user can browse to other related resources via the links stored in the data mart. We describe how those three features (search, showing detail and browsing) benefit from semantic technologies in the rest of this chapter. We also present two examples of advanced features built in the application.

³ <http://www.unifiedviews.eu>

4.1 Full Text Search of Resources in the Application Data Mart

The starting page of the application offers a full-text search field which allows a user to search resources in the data mart. When the user types at least 4 letters, the full-text search engine searches the application data mart for resources whose literal properties values contain the searched string and provides them to the user. The search is implemented as a SPARQL query which can be configured by the administrator to search only for resources of a given type and restrict the search only to given properties of those resources. All available language specific values of the configured properties are searched independently of the current language set by the user as the user interface language. For example, the user may specify the search string in Latin (physicians often use Latin) even if he has set Czech as his current language. The result is always shown in the language selected by the user, i.e. Czech in our case. In the data mart, we currently have textual properties of resources translated to Czech, English and Latin. Therefore, the user can type his search string in any of these three languages. The user can select Czech and English as his language for the application user interface. Latin is not supported for the application user interface.

Semantic technologies enable us to easily implement the above mentioned configuration of the search. We can implement advanced search strategies because of the flexibility of the RDF data model and SPARQL query language. A single SPARQL search query can contain different search strategies for RDF resources of different classes. E.g., we can say that all resources of sub-classes of `skos:Concept` (e.g., ATC group) are searched using the property `skos:prefLabel` while instances of a specific application class `enc:MedicinalProduct` are searched using specific application property `enc:title` which unifies various equivalent properties used for expressing a displayable title of a resource used by different data sets integrated in the application data mart. Moreover, if the schema of the data mart will be extended in the future (e.g., a new class and its instances will be added) it will be very easy to modify the search engine – it will be only necessary to extend the SPARQL search query.

Figure 3 shows the application user interface for searching. Every time the user writes at least 4 letters into the search field, a SPARQL query is executed and the search result is shown in the auto-complete panel (see the Figure 3 (A)). The SPARQL search query is also executed when the user clicks on the search button (see the Figure 3 (B)). Because of the simplicity, the auto-complete panel shows only a subset of the search result of all search strategies.

Formally, each class whose resources should be searched by the application is associated with a *search strategy*. A search strategy is a SPARQL graph pattern. All search strategies are concatenated using the SPARQL UNION construct. A search strategy must contain a variable `?resource` which matches the found resource, `?type` which matches the type(s) of `?resource` and `?text` which matches the found textual value of a property of `?resource`. The resulting SPARQL search query then constructs the search results from the union of the search strategies in the following form:

(A)

Search

Active ingredients

Ibuprofen

Medicinal products

IBUPROFEN AL 400

IBUPROFEN SANDOZ 20 MG/ML

IBUPROFEN PERRIGO 200 MG POTAHOVANÉ TABLETY

APO-IBUPROFEN 400 MG

IBUPROFEN 400 MG GALMED

APO-IBUPROFEN RAPID 400 MG SOFT CAPSULES

IBUPROFEN FARMALIDER 100 MG/5 ML PERORÁLNÍ SUSPENZE

IBUPROFEN FARMALIDER 200 MG/5 ML PERORÁLNÍ SUSPENZE

ACTION

potential interactions and products.

Search results

Active ingredients

Ibuprofen

ATC concepts

C01EB16 Ibuprofen

G02CC01 Ibuprofen

M01AE01 Ibuprofen

M01AE13 Ibuproxam

M01AE51 Ibuprofen, combinations

M02AA13 Ibuprofen

Medicinal products

APO-IBUPROFEN 400 MG

APO-IBUPROFEN RAPID 400 MG SOFT CAPSULES

IBUPROFEN 400 MG GALMED

IBUPROFEN AL 400

IBUPROFEN FARMALIDER 100 MG/5 ML PERORÁLNÍ SUSPENZE

IBUPROFEN FARMALIDER 200 MG/5 ML PERORÁLNÍ SUSPENZE

IBUPROFEN PERRIGO 200 MG POTAHOVANÉ TABLETY

IBUPROFEN SANDOZ 20 MG/ML

Drug Encyclopedia

to start? Insert at least first 4 letters of an active ingredient or medicinal product and the applicable... This means either class from the presented categories can be hit the Search button.

Fig. 3. Searching resources in the application data mart with Drug Encyclopedia

```
?resource a ?type ;
  enc:title ?text .
```

(The application understands the value of `enc:title` as a text to be displayed to the user.) Let us demonstrate the full-text search on a concrete example. When the user types “*ibup*” to the search field, the search engine uses all defined search strategies to explore the data mart. The following strategies are successful (i.e. return a non-empty result):

- The search strategy for class `enc:Ingredient` matches each active ingredient *I* with a value *t* of property `enc:title` such that *t* starts with “*ibup*”. For *I*, *t* is assigned to `?text`.
- The search strategy for class `enc:ATCConcept` matches each ATC group *A* with a value *l* of property `skos:prefLabel` such that *l* starts with “*ibup*”. For a found ATC group, it constructs the value of `?text` as a concatenation of the code of the ATC group (`skos:notation`) followed by the title of the ATC group (`skos:prefLabel`).
- The search strategy for class `enc:MedicinalProduct` matches all medicinal products in the same way as the search strategy for `enc:Ingredient`.

The search strategies for other classes, e.g., `enc:DiseaseOrFinding`, do not find any resource for the specified search string. If there is a requirement to change the searching logic (e.g., a new kind of resources is added), it is only necessary to add a new search strategy or to modify the existing one.

4.2 Viewing a Detail of a Resource in the Application Data Mart

When the user clicks on one of the resources found by the search engine, the application shows the detail of that resource. The detail page renders the displayable title of the resource (value of `enc:title`), its displayable description

(value of `enc:description`) and then other properties of the resource stored in the application data mart. Values of each property are displayed in a paragraph with the name (value of `rdfs:label`) of the property in its title.

Figure 4 contains 2 screen shots – a detail of the active ingredient *Ibuprofen* (on the left) and a detail of the medicinal product *Brufen 400*. Both details are displayed using the same page template – the title is shown at the top together with the class of the resource (value of `rdfs:label` of that class). Then there is a description of the resource below the title if it is available. The paragraphs for the values of properties of the resource are displayed below. For example, for *Ibuprofen* (on the left), there are paragraphs *Indication*, *Prevention*, *Contraindication*, *Pharmacological actions*, *Mechanisms of actions*, *Physiologic effects*, *Pharmacokinetics*, *Pregnancy categories*, etc.

Ibuprofen
Active ingredients

Detail Interactions

A nonsteroidal anti-inflammatory agent with analgesic properties used in the therapy of rheumatism and arthritis.

| Indication | Prevention | Contraindication |
|--|------------|--|
| Arthritis Arthritis, gouty Arthritis, juvenile rheumatoid Arthritis, psoriatic Arthritis, rheumatoid Back pain Bacterial infections Body weight | Pain | Asthma Bronchial hyperreactivity Drug hypersensitivity Pregnancy trimester, third Pregnancy, abdominal Rhinitis |

Pharmacological actions
Analgesics, non-narcotic | Anti-inflammatory agents, non-steroidal | Cyclooxygenase inhibitors

Mechanisms of action
Cyclooxygenase Inhibitors

Physiologic effects
Decreased Platelet Activating Factor Production | Decreased Prostaglandin Production | Decreased Thrombox

Pharmacokinetics
Hepatic Metabolism | Renal Excretion

Pregnancy categories
D
C
Contraindicated in third trimester (source: ndf-rt)
Contraindicated for abdominal pregnancy (source: ndf-rt)

Medicinal products

| Title | |
|----------------------|---------|
| ADVIL RAPID | M01AE01 |
| APO-IBUPROFEN 400 MG | M01AE01 |

BRUFEN 400
Medicinal products

Detail SPC Contraindication Pregnancy Adverse effects

Indication
Brufen se užívá k symptomatické léčbě reumatoidní artritidy, včetně Stillovy nemoci, juvenilní id spondylartritidy a osteoartrózy. Je rovněž indikován k léčbě bolesti svalů a kloubů, které provázejí například podvrtnutí kloubů a natažení svalů, k tlumení mírných nebo středně silných bolestí na zad, k symptomatickému mírnění bolesti hlavy (včetně migrény) a jako antipyretikum při horečk

Indication group
ANTIRHEUMATICA, ANTIPHLOGISTICA, ANTIURATICA

ATC concepts
M MUSCULO-SKELETAL SYSTEM | M01 ANTIINFLAMMATORY AND ANTIRHEUMATIC PI
M01A ANTIINFLAMMATORY AND ANTIRHEUMATIC PRODUCTS, NON-S | M01AE Propior

Active ingredients
Ibuprofen

Pharmacological actions
Analgesics, non-narcotic
Anti-inflammatory agents, non-steroidal
Cyclooxygenase inhibitors

Pregnancy category
D
C
Contraindicated for abdominal pregnancy (source: ndf-rt)
Contraindicated in third trimester (source: ndf-rt)

Medicinal product packaging

| Registration status | Medicinal product packaging | Strength | Packaging size | I adr |
|---------------------|------------------------------------|----------|----------------|--------------|
| R Yes | BRUFEN 400 POR TBL FLM 30X400MG | 400MG | 30 | Perc podí |
| R | BRUFEN 400 | 400MG | 100 | Perc |

Fig. 4. Showing a detail of a resource (left: active ingredient *Ibuprofen*, right: medicinal product *Brufen 400*) in Drug Encyclopedia

The different layouts for both detail pages in Figure 4 are not given by different page templates. There is only one page template for details of all types of medicinal resources in the application data mart. The different layouts are the result of a mechanism which assigns a specific visual configuration to each property. The configuration is applied anywhere the property appears. However, there is not a specific configuration for each specific property. Instead, the configuration is specified in a more semantic way.

There is a basic configuration saying how literal properties should be displayed and another basic configuration for object properties. If no more specific visual configuration is available, the basic one is applied for displaying the visual paragraph with the values of the property. For example, the basic configuration for object properties is used to display values of properties *Pharmacological actions* and *Mechanisms of action* in the detail of *Ibuprofen* on the left of Figure 4.

Moreover, there can be more specific visual configurations for specific kinds of properties with specific object values. For example, we have a specific visual configuration for the following kinds of object values:

- An object value which is an instance of `skos:Concept` and is related to the displayed resource with `skos:broader`. For this object value we show also the `skos:broaderTransitives`. The configuration is applied to render the *ATC Concepts* paragraph in the detail of *Brufen 400*. Here, we can see not only the ATC concept *M01AE01* but also all its broader transitive concepts – *M01AE*, *M01A*, *M01*, and *M*.
- Object values which are instances of `enc:Ingredient` - whenever an active ingredient is displayed, we do not display it as a generic object but we show a richer visualization which presents also other selected properties of the active ingredient (we show its pharmacological actions and pregnancy categories in the visualization). This configuration is applied to render the paragraph *Active ingredients* in the detail of *Brufen 400*.
- Object values which are related to an instance of class `enc:Interaction` with `enc:interactionMember` property such that the displayed resource is also related to this instance of `enc:Interaction` with the same property. The meaning of such relationship is that the displayed resource (e.g, a medicinal product, active ingredient or ATC group, etc.) interacts with the object value (again a medicinal product, active ingredient or ATC group, etc.). Moreover, the instance of `enc:Interaction` may be linked to an SPC with `frbr:textChunk` property. If the interaction is linked to an SPC it means that the interaction has been extracted from the SPC using NLP techniques. (We described the extraction mechanism in our previous paper [6]). The visualization shows these interactions as links to the corresponding place in the SPC where the interaction is described. The label of the link is the title of the other interacting object. The user can see this configuration in action when he chooses a detail of a medicinal product packaging (accessible from the detail of a medicinal product). It is applied for rendering the paragraph *Interactions (extracted from SPC)*.

For a detail page, a SPARQL query which extracts data about the displayed resource from the database is executed. There is a basic SPARQL query which extracts all triples with the displayed resource as a subject. The basic query can be extended by the administrator for selected application classes – any SPARQL query which returns the displayed resource and its surroundings can be provided instead of the basic query. The custom query can, e.g., ignore some properties or go beyond the distance 1 for some other resources. As a result, each application class is associated either with the basic query or a custom one.

Then, the application automatically analyzes the obtained data and assigns a visual configuration to each found property related to the displayed resource. This is possible because of the basic visual configurations which assign a visual configuration to each property and are described above. The analysis means that each template is checked whether it can visualize a given property which appears in the data. The template implements a method which is able to traverse the property and its values (and their properties and so on) and check whether the data contain everything what is necessary for the template. This can be an expensive process. However, because the assignment is done only on a very small portion of data, it is computed quickly.

This dynamic execution model is very flexible. We are able to display the detail of any kind of resource and use predefined visual configurations to render the detail page without the need of programming a specific code for each kind of displayed resources. This is useful mainly when new information to existing kinds of resources is added, when the representation of existing information is changed or when completely new kinds of objects are added. The application code has to be updated or extended only when a specific visualization of a property is required. When the data model is changed but there is no requirement to change the visual style, no intervention in the application code is required.

The mechanism built on top of semantic technologies described above also facilitates the maintenance of the consistency of the application logic and visual style within the whole application. A group of properties with the same or similar semantics is displayed and behaves in the same way across the whole application because their visual configuration is chosen on the base of their semantics instead of a programmatic selection directly in the application code. For example, anywhere in the application where a property with `skos:Concept` as its range is displayed, the visual style and behavior is the same if this rule is not explicitly overridden by another visual configuration.

4.3 Browsing Resources

When a detail page is displayed, values of object properties are displayed as links to detail pages of those object values. The basic visual configuration for object properties ensures this behavior. When a more specific configuration is defined it is required to display the objects as links as well. The link is an application URL which contains the URI of the object as a parameter. The mechanism for generating the detail page described above ensures that the detail of the linked object is compiled automatically. Therefore, when the user clicks on the link, the detail page of the object is displayed. By following the links, the user can browse the whole data mart because the data mart forms a connected graph.

The physician can use this browsing feature to explore the application data mart. He can, for example, discover medicinal products which are more suitable for his patient than the one he currently uses. For example, the patient uses the medicinal product *Brufen 400*. However, the patient is pregnant and the active ingredient of this medicinal product is contraindicated in pregnancy (which the physician can see on the detail of *Ibuprofen* in *Pregnancy categories*

paragraph). Therefore, he can browse the data mart to explore active ingredients and medicinal products containing those active ingredients using the links to pharmacological actions, mechanisms of action, etc. from the detail of *Ibuprofen*.

4.4 Advanced Features

The application contains several advanced features. Let us pick up two of them. First, there is an interaction finder. In the application data mart, there are integrated interactions between pairs of active ingredients from different data sources. The application provides a feature which allows a user to specify a set of active ingredients and/or medicinal products and then check the potential interactions among them. The interactions are discovered using a relatively complicated SPARQL query (its effectiveness is discussed in Section 5).

Second, there is a possibility to display SPC documents we have already discussed in the Section 4.2. An SPC document is a textual document with a prescribed structure of sections and subsections which provides a clinical information about a medicinal product to a physician. We have processed all SPC documents of medicinal products available on the Czech market – we converted them to RDF representation (we used *SALT* and *FRBR* ontologies⁴) and annotated medical resources in the text (medicinal products, active ingredients, etc.). The details of this process are described in our previous paper [6]. The application enables a user to view the SPC document from the detail page of each medicinal product. When the SPC document is displayed, the recognized medical resources are displayed as links to their details. Therefore, SPC documents become a part of the browsing feature provided by the application. A user can go to the SPC document from a detail page of a medicinal product and from here he can browse to the details of related medical resources.

5 Evaluation

The application was deployed at the end of 2013 and has been running since then. More than 3,300 unique users used our application at least once and about 57 % of them returned to the application during the time. In this section, we present the evaluation of performance of the application and statistics about usage of the application and user satisfaction.

5.1 Performance

The application is running on a virtual server with dedicated 10 GB of RAM and can use up to 16 cores of CPU (Intel Xeon CPU E5620 @ 2.40GHz). As mentioned before, we use Virtuoso as the underlying triple store. We have designed a test to evaluate the performance of the application queries. The test contains 11 different application queries which are run for 10 times sequentially. The run times of the

⁴ Can be found at Linked Open Vocabularies: <http://lov.okfn.org/>

queries and the number of resulting triples are shown in the Table 5.1. All queries run less than 1 second except the detail of an ATC concept and interaction finder. The reason for long run time of interaction finder is the complexity of the task and the query. It checks interaction among 4 entities – 2 medicinal products and 2 active ingredients – and searches for every pair that could possibly interact.

Table 1. Execution time of the application queries in milliseconds and triples in result.

| Action | MIN | MAX | AVG | # triples |
|---|-------|--------|-------|-----------|
| Searching for <i>ibup</i> | 363 | 496 | 391 | 53 |
| ATC Concept detail (M01AE01 Ibuprofen) | 1,373 | 1,509 | 1,440 | 226 |
| Disease or Finding detail (Osteoarthritis) | 571 | 742 | 619 | 960 |
| Ingredient detail (Ibuprofen) | 130 | 195 | 148 | 466 |
| Medicinal Product detail (BRUFEN 400) | 571 | 742 | 619 | 147 |
| Medicinal Product Packaging detail (BRUFEN 400, POR TBL FLM 10X400MG) | 648 | 891 | 756 | 763 |
| Pharmacological Action detail (Anti-inflammatory agents, non-steroidal) | 450 | 690 | 493 | 2,540 |
| Mechanism of Action detail (Cyclooxygenase Inhibitors) | 110 | 212 | 132 | 542 |
| Physiological Effect detail (Decreased Prostaglandin Production) | 184 | 264 | 218 | 907 |
| Pharmacokinetics detail (Renal Excretion) | 664 | 964 | 218 | 2,071 |
| Interactions Finder | 6,416 | 10,337 | 7,065 | 207 |

5.2 Application Usage

Here, we present the statistics collected via Google Analytics⁵ (GA) during the period between 2014/01/01 and 2015/02/28. The collected data does not contain any sessions triggered by developers of the application because they use a special attribute for accessing the application even for the purpose of presentations. Besides the statistics from GA, we also briefly present results from a questionnaire which was filled by 13 physicians who are users of the application.

GA Statistics. The application has attracted 3,324 unique users during the presented period which means 7,635 sessions and 35,133 pageviews in total. The monthly statistics can be seen in the Figure 5 (A). It results in almost 17 sessions per day. These numbers of sessions and pageviews (and corresponding statistics below) include 2,198 sessions when only homepage was visited and then the application was left immediately. We are not able to filter them out from all of the statistics because GA does not allow such fine grained export. We call these sessions “dead sessions” further on.

The users were coming to the application continuously except 2 peaks in 2014/01 and 2015/01 when little promotion was done. There were almost no users coming during weekends. Besides that, users are returning to the application. 57 % of all users used our application more than once. More detailed analysis shows that 37 % users came to the application more than 9 times. The

⁵ Google Analytics – tracks website traffic – <http://www.google.com/analytics>

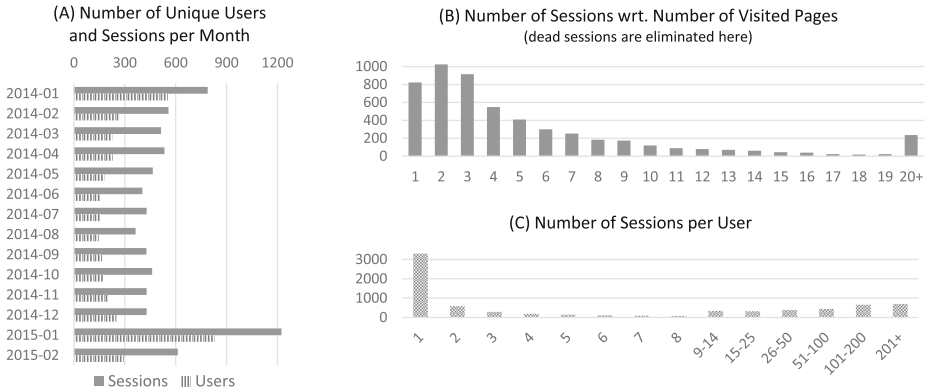


Fig. 5. Statistics about sessions in the application - numbers were exported from GA.

distribution of sessions per user can be found in Figure 5 (C). About 10 % of users were coming from mobile devices.

Interesting facts arise from the analysis of pages visited per session. There is 40 % of sessions with 5 or more visited pages. Thanks to the flow analysis available in GA we may say this represent the exploratory way of work rather than quick information lookup. The distribution of numbers of pages visited in one session is shown in the Figure 5 (B).

We have also set up GA to track types of pages visited. We exclude the static pages (home, about application etc.) from the following statistics. The most of the pageviews come from types medicinal product or its packaging - 11,542 pageviews. Then there were 3,557 pageviews of SPCs, 2,427 pageviews of ingredients, 2,270 pageview of ATC concept and 1,516 pageviews of interactions finder. The interaction finder logs only the first access to the mini application. The number of submitted interaction checks was not logged in the GA.

Survey. We conducted a small survey among the users of *Drug Encyclopedia* and collected 13 responses from the users, mainly physicians. The results of the 3 most important questions can be found in Figure 6. Questions were the following.

- Q1** Do you usually find information you are looking for?
- Q2** Does Drug Encyclopedia offer more information than comparable information sources in Czech?
- Q3** Is Drug Encyclopedia more user friendly than other comparable information sources or products?

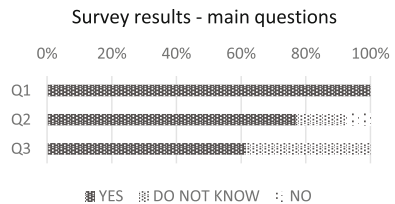


Fig. 6. Brief survey results

The results show the usefulness of the application from the users' perspective.

6 Related Work

LD was used in many domains of interest and many projects were also done in the area of biomedicine. One of the most important projects for sharing biomedical ontologies and terminologies is definitely BioPortal [11], [12]. Other projects try to give users interconnected databases of general biomedical data. These include, e.g., Bio2RDF [2], Linked Life Data⁶. There exist projects dealing specifically with drug related data. Linked Open Drug Data (LODD) was one of the first which published drug data as LD [13]. The major project for drug discovery is currently OpenPHACTS which integrates many databases about drugs [15].

All the projects above provide platforms for data discovery in the biomedical/pharmaceutical domains. They are not intended for end users, i.e. physicians or patients. When a person needs to get information relevant to clinical practice he needs to go somewhere else. Specialized applications have to be developed for this purpose. There is not so many applications build on top of LD.

LODMedics [14] does that for information about drugs. But it is a very simple application. There also exists an application called Pharmer [4] which tries to target semantic prescription of drugs. It uses data from LODD and is rather in the early stage of development. Neither of the mentioned applications allow browse the LD space e.g., show all drugs for a specific pharmacological action.

The complexity of underlying data (often integrated from many data sources) is a tough hurdle for creating such application and is necessary to make it easier e.g., by providing a reasonable API [3] with a simplified data model. The model can be represented by so called application ontology which is a subset of the full reference ontology [10]. This approach was used e.g., in [9].

7 Lessons Learned and Future Work

During our work, we learned several lessons about developing a web application using semantic web technologies. We learned that semantic technologies make the process of integrating various publicly available data sources much more flexible and easier. Moreover, the application code can be more flexible. The proposed mechanism of visual templates helps to reduce the amount of code. This significantly saves time of the database administrators and developers. We further list some important lessons learned.

Iteration of RDF Representation. Using RDF as a data model has several advantages. It is easy to convert the data sources to an RDF representation. At the beginning, it is sufficient to create some RDF representation which reflects the structure of the original data sources one-to-one. In the next iterations, the RDF representation of each data source can be improved by aligning the RDF structure with existing ontologies and vocabularies. Each particular data source and each link set between two data sources should be logically organized in a separate RDF graph. This allows the database administrator to see how each

⁶ Linked Life Data – <http://linkedlifedata.com>

data source and link set was changed and improved during the alignment and linking iterations. This makes the discovery of errors in the alignment and linking procedures easier.

Application Data Mart. The LD warehouse should be further optimized with respect to the typical application SPARQL queries. As we describe in Section 3, optimization means simplification (unification of properties and path shortening). The optimization is useful because the resulting SPARQL queries are more easily maintainable and can be evaluated more effectively by a triple store. Let us note that these simplification is materialized in the database as a set of new triples which should be stored in separate RDF graphs.

Visual Templates for Application Data. One of the most important lessons learned during our work was that the application code should be as generic as possible. This does not mean to create an application which serves as a generic LD browser. Rather it means that there should be a single template which shows all resources in the application data mart in a uniform way and which unifies the visual style of common properties across different kinds of resources. We implemented this practice by introducing generic visual configurations which are applicable to any RDF property and specific visual configurations which are applied to selected properties (details in Section 4). Therefore, it is not necessary to code a specific template for each type of resources. Instead, there is only one template and then visual configurations for particular kinds of properties which are dynamically applied when a resource is displayed. Such approach reduces the amount of code and makes the application code maintenance much easier.

Future Work. Because the LD warehouse and the application are ready for integration of other data sources, we are planning to extend the application in the near future. We would like to add links to current literature about drugs and also spread the application to other countries where local medicinal products are on the market. Moreover, we are also planning to build a mobile application with specific functions for patients.

Acknowledgments. This work was partially supported by the project GACR P103/13/08195S and the project SVV-2015-260222.

References

1. Berners-Lee, T.: Linked Data (2006). <http://www.w3.org/DesignIssues/LinkedData.html> (accessed: April 24, 2015)
2. Callahan, A., Cruz-Toledo, J., Ansell, P., Dumontier, M.: Bio2rdf release 2: Improved coverage, interoperability and provenance of life science linked data. In: *The Semantic Web: Semantics and Big Data*, pp. 200–212. Springer (2013)
3. Chichester, C., Harald, L., Harder, T.: Mobile applications driven by open phacts semantic web technology. *EMBnet. Journal* **19**(B), 21–23 (2013)
4. Khalili, A., Sedaghati, B.: Semantic medical prescriptions-towards intelligent and interoperable medical prescriptions. In: *2013 IEEE Seventh International Conference on Semantic Computing (ICSC)*, pp. 347–354. IEEE (2013)

5. Knap, T., Skoda, P., Klímek, J., Necaský, M.: Unifiedviews: towards ETL tool for simple yet powerful RDF data management. In: Proceedings of the Dateso 2015 Workshop, pp. 111–12 (2015)
6. Kozák, J., Necaský, M., Dedek, J., Klímek, J., Pokorný, J.: Using linked data for better navigation in summaries of product characteristics. In: Proceedings of the 6th International Workshop on Semantic Web Applications and Tools for Life Sciences, Edinburgh, UK, December 10, 2013
7. Kozák, J., Nečaský, M., Dědek, J., Klímek, J., Pokorný, J.: Linked open data for healthcare professionals. In: Proceedings of International Conference on Information Integration and Web-Based Applications & Services, pp. 400–409. IIWAS 2013. ACM, New York (2013)
8. Manola, F., Miller, E.: RDF Primer. W3C Recommendation (February 10, 2004)
9. Mejino, J.L.V., Rubin, D.L., Brinkley, J.F.: FMA-Radlex: an application ontology of radiological anatomy derived from the foundational model of anatomy reference ontology. In: AMIA Annual Symposium Proceedings, pp. 465–469 (2008)
10. Menzel, C.: Reference ontologies - application ontologies: either/or or both/and? In: Proceedings of the KI2003 Workshop on Reference Ontologies and Application Ontologies, Hamburg, Germany, September 16, 2003
11. Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A., Chute, C.G., Musen, M.A.: BioPortal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research* (2009)
12. Salvadores, M., Horridge, M., Alexander, P.R., Ferguson, R.W., Musen, M.A., Noy, N.F.: Using SPARQL to query bioportal ontologies and metadata. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 180–195. Springer, Heidelberg (2012)
13. Samwald, M., Jentzsch, A., Bouton, C., Kallesø, C., Willighagen, E., Hajagos, J., Marshall, M., Prud'hommeaux, E., Hassenzadeh, O., Pichler, E., Stephens, S.: Linked open drug data for pharmaceutical research and development. *Journal of Cheminformatics* **3**(1), 1–6 (2011)
14. Shruthi Chari, S.R., Mahesh, K.: LODMedics: Bringing Semantic Data to the Common Man (2014). http://challenge.semanticweb.org/2014/submissions/swc2014_submission_7.pdf
15. Williams, A.J., Harland, L., Groth, P., Pettifer, S., Chichester, C., Willighagen, E.L., Evelo, C.T., Blomberg, N., Ecker, G., Goble, C., et al.: Open PHACTS: semantic interoperability for drug discovery. *Drug discovery today* **17**(21), 1188–1198 (2012)

Collecting, Integrating, Enriching and Republishing Open City Data as Linked Data

Stefan Bischof^{1,2}(✉), Christoph Martin², Axel Polleres²(✉),
and Patrik Schneider^{2,3}(✉)

¹ Siemens AG Österreich, Vienna, Austria

² Vienna University of Economics and Business, Vienna, Austria
`bischof.stefan@siemens.com`, `axel.ploeres@wu.ac.at`

³ Vienna University of Technology, Vienna, Austria
`patrik@kr.tuwien.ac.at`

Abstract. Access to high quality and recent data is crucial both for decision makers in cities as well as for the public. Likewise, infrastructure providers could offer more tailored solutions to cities based on such data. However, even though there are many data sets containing relevant indicators about cities available as open data, it is cumbersome to integrate and analyze them, since the collection is still a manual process and the sources are not connected to each other upfront. Further, disjoint indicators and cities across the available data sources lead to a large proportion of missing values when integrating these sources. In this paper we present a platform for collecting, integrating, and enriching open data about cities in a reusable and comparable manner: we have integrated various open data sources and present approaches for predicting missing values, where we use standard regression methods in combination with principal component analysis (PCA) to improve quality and amount of predicted values. Since indicators and cities only have partial overlaps across data sets, we particularly focus on predicting indicator values across data sets, where we extend, adapt, and evaluate our prediction model for this particular purpose: as a “side product” we learn ontology mappings (simple equations and sub-properties) for pairs of indicators from different data sets. Finally, we republish the integrated and predicted values as linked open data.

1 Introduction

Nowadays governments have large collections of data available for decision support. Public administrations use these data collections for backing their decisions and policies, and to compare themselves to other cities, and likewise infrastructure providers like Siemens could offer more tailored solutions to cities based on this data. Having access to high quality and current data is crucial to advance

Compared to an informal, preliminary version of this paper presented at the Know@LOD 2015 workshop, Section 5, 6, and 8 are entirely new, plus more data sources have been integrated.

on these goals. Studies like the Green City Index [6] which assess and compare the performance of cities are helpful, in particular for public awareness. However, these documents are outdated soon after publication and reusing or analyzing the evolution of their underlying data is difficult. To improve this situation, we need regularly updated data stores which provide a consolidated, up-to-date view on relevant open data sources for such studies.

Even though there are many relevant data sources which contain quantitative indicators, e.g., population, about cities available as *open data*, it is still cumbersome to collect, clean, integrate, and analyze data from these sources: obstacles include different indicator specifications, different languages, formats, and units. Example sources of city data include DBpedia or the Urban Audit data set included in Eurostat; Urban Audit (<http://ec.europa.eu/eurostat/web/cities/>) for example, provides over 250 indicators on several domains for 258 European cities. Furthermore, several larger cities provide data on their own open data portals, e.g., London, Berlin, or Vienna.¹ Data is published in different formats such as RDF, XML, CSV, XLS, or just as HTML tables. The specifications of the individual data fields – (i) how indicators are defined and (ii) how they have been collected – are often implicit in textual descriptions only and have to be processed manually for understanding.

Moreover, data sources like Urban Audit cover many cities and indicators, but show a large ratio of *missing values* in their data sets. The impact of missing values is even aggravated when combining different data sets, since there is a fair amount of disjoint cities and indicators across those data sets, which makes them hard to integrate. Our assumption though – inspired also by works that suspect the existence of quantitative models behind the working, growth, and scaling of cities [1] – is that most indicators in such a scoped domain have their own structure and dependencies, from which we can build prediction models:² we evaluate different standard regression methods to choose the best fitting model to predict missing indicator values. We follow two approaches for computing such predictions. The first approach is based on a selection of “relevant” indicators as predictors for a target indicator. The second approach constructs the principal components (PCs) of the “completed” data sets (missing values are replaced with “neutral” values [21]), which are then used as predictors. We also compare both approaches according to their performance, prediction accuracy, and coverage (the number of possible predictions). We then extend the second approach for cross data set prediction, in case of a large disjointness of indicators and cities.

Contributions and Structure. Our concrete contributions are:

- We analyze and integrate several data sets (DS) including DBpedia, Urban Audit, USCCDB, and the UNSD Demographic and Social Statistics;

¹ <http://data.london.gov.uk/>, <http://daten.berlin.de/>, and <http://data.wien.gv.at/>

² We refer to “predicting” instead of “imputing” values when we mean finding suitable approximation models to predict indicators values for cities and temporal contexts where they are not (yet) available. These predictions may (not) be confirmed, if additional data becomes available.

- We provide a system architecture for an “City Data Pipeline” including a crawler, wrappers, ontology-based integration, and data access components;
- We evaluate two prediction approaches for filling-in missing values, combining different standard regression methods and PCs to maximize prediction accuracy;
- We develop an approach for cross DS prediction and discuss its performance;
- We present an approach for learning mappings of indicators between DS;
- We republish the integrated and predicated values as linked open data (LOD).

Section 2 describes the imported data sources and the challenges arising when processing/integrating their data. Section 3 presents an overview of the Open City Data Pipeline and a lightweight extensible ontology used therein. In Section 4 and 5 we explain the approaches developed for predicting missing values as well as the corresponding evaluation of their performance. Section 6 presents our ontology mapping learning approach. Our LOD interface to republish the integrated and predicted data is documented in Section 7. In Section 8 we discuss the use of Semantic Technologies and the lessons learnt from our application. Section 9 concludes with several possible future extensions.

2 Data Sources

The Open City Data Pipelines database contains data ranging from the years 1990 to 2014, but most of the data concerns years after 2000. Not every indicator is covered over all years, where the highest overlap of indicators is between 2004 and 2011 (see Tables 1 and 2). Most European cities are contained in the Urban Audit data set, but we also include the capital cities and cities with a population over 100 000 from the U.N. Demographic Yearbook (UNYB).

Before integration, locations have varying names in different data sets (e.g., Wien vs. Vienna), a Uniform Resource Identifier (URI) for every city is essential for the integration and enables to link the cities and indicators back to DBpedia and other LOD data sets. We choose to have a one-to-one (functional) mapping of every city from our namespace to the English DBpedia resource, which in our republished data is encoded by **sameAs** relations. We identify the matching DBpedia URIs for multilingual city names and apply basic *entity recognition*, similar to Paulheim et al. [17], with three steps using the city’s names from Urban Audit and UNYB:

- Accessing the DBpedia resource directly and following possible redirects;
- Using the Geonames API (<http://api.geonames.org/>) to identify the resource;
- For the remaining cities, we manually looked up the URL on DBpedia.

Table 1. Urban Audit Data Set

| Year(s) | Cities | Indicators | Available Values | Missing Values | Missing Ratio (%) |
|-------------------|--------|------------|------------------|----------------|-------------------|
| 1990 | 177 | 121 | 2 480 | 18 937 | 88.4 |
| 2000 | 477 | 156 | 10 347 | 64 065 | 85.0 |
| 2005 | 651 | 167 | 23 494 | 85 223 | 78.4 |
| 2010 | 905 | 202 | 90 490 | 92 320 | 50.5 |
| 2004 - 2012 | 943 | 215 | 531 146 | 1 293 559 | 70.9 |
| All (1990 - 2012) | 943 | 215 | 638 934 | 4 024 201 | 86.3 |

DBpedia. DBpedia, initially released in 2007, is an effort to extract structured data from Wikipedia and publish the data as Linked Data [4]. For cities, DBpedia provides various basic indicators such as demographic and geographic information (e.g., population, latitude/longitude, elevation). The Open City Data Pipeline extracts the URLs, weather data, and the population of a city. While we only integrated a limited subset of indicators from DBpedia for now, we plan to add other indicators like economic and spatial indicators in the future. Since temporal validity of indicators is rarely documented, we assume them to be current as accessed.

Urban Audit (UA). The Urban Audit collection started as an initiative to assess the quality of life in European cities. It is conducted by the national statistical institutes and Eurostat. Currently, data collection takes place every three years (last survey in November 2012) and is published via Eurostat (<http://ec.europa.eu/eurostat>). All data is provided on a voluntary basis which leads to varying data availability and missing values in the collected data sets. Urban Audit aims to provide an extensive look at the cities under investigation, since it is a policy tool to the European Commission: “The projects’ ultimate goal is to contribute towards the improvement of the quality of urban life” [15]. At the city level, Urban Audit contains over 250 indicators divided into the categories Demography, Social Aspects, Economic Aspects, and Civic Involvement. Currently, we extract the data sets including the topics population structure, fertility and mortality, living conditions and education, culture and tourism, labour market, transport, and environment.

United Nations Statistics Division (UNSD). The UNSD offers data on a wide range of topics such as education, environment, health, technology, and tourism. Our main source is the UNSD Demographic and Social Statistics, which is based on the data collected annually (since 1948) by questionnaires to national statistical offices (<http://unstats.un.org/unsd/demographic/>). The UNSD data marts consist of the following topics: population by age distribution, sex, and housing; occupants of housing units/dwellings by broad types (e.g., size, lighting); occupied housing units by different criteria (e.g., walls, waste). The collected data has over 650 indicators, wherein we kept a set of course-grained indicators and drop the most fine-grained indicator level, e.g., keeping *housing units total* but dropping *housing units 1 room*. We prefer more coarse-grained indicators to avoid large groups of similar indicators which are highly correlated. Fine-grained indicators would only be interesting for LOD publication.

Table 2. United Nations Data Set

| Year(s) | Cities | Indicators | Available Values | Missing Values | Missing Ratio (%) |
|-------------------|--------|------------|------------------|----------------|-------------------|
| 1990 | 7 | 3 | 10 | 11 | 52.4 |
| 2000 | 1 391 | 147 | 7 492 | 196 985 | 96.3 |
| 2005 | 1 048 | 142 | 3 654 | 145 162 | 97.5 |
| 2010 | 2 008 | 151 | 10 681 | 292 527 | 96.5 |
| 2004 - 2012 | 2 733 | 154 | 44 944 | 3 322 112 | 98.7 |
| All (1990 - 2012) | 4 319 | 154 | 69 772 | 14 563 000 | 99.5 |

U.S. Census. The *County and City Data Book 2007* (USCCDB) of U.S. Census Bureau [26] offers two data sets concerning U.S. statistics; *Table C-1 to C-6* of [26] cover the topics Area and Population, Crime, Civilian Labor Force for cities larger than 20 000 inhabitants; *Table D-1 to D-6* of [26] cover Population, Education, Income and Poverty for locations with 100 000 inhabitants and more. Initially, we have integrated the data sets from Table C-1 to C-3, which are the only sources including data points for several years, namely 1990, 2000, and 2005. Contrary to the UN and UA data sets, the USCCDB has a low ratio of missing values ranging from 0% to 5% for a total of 1267 cities. The data set contains 21 indicators, e.g., population, crime, and unemployment rate.

Future Data Sources. At the point of writing, the data sources are strongly focused on European cities and demographic data. Hence, we aim to integrate further national and international data sources. The Carbon Disclosure Project (CDP) is an organization based in the UK aiming at “[...] using the power of measurement and information disclosure to improve the management of environmental risk” (<https://www.cdp.net/en-US/Pages/About-Us.aspx>). The *CDP cities* project has data collected on more than 200 cities worldwide. CDP cities offers a reporting platform for city governments using an online questionnaire covering climate-related areas like Emissions, Governance, and Climate risks. Single city open data portals (e.g., New York, Vienna) could be added and integrated. This is surely a large effort by its own, since our crawling and mapping components would have to be extended to deal with heterogeneity of every cities’ portal.

3 System Architecture

The Open City Data Pipeline collects data, organizes it into indicators, and shows these indicators to the user. This section introduces the system which is organized in several layers (see Figure 1): *crawler*, *wrapper components*, *semantic integration*, *data storage*, *analytics*, and *external interfaces* (user interface, SPARQL endpoint, and LOD).

Crawler. The Open City Data Pipeline semi-automatically collects data from various registered open data sources periodically dependent on the specific source. The crawler currently collects data from 32 different sources. Due to a high heterogeneity in the source data, adding new data sources is still a manual process, where the source-specific mapping of the data to RDF has to be provided

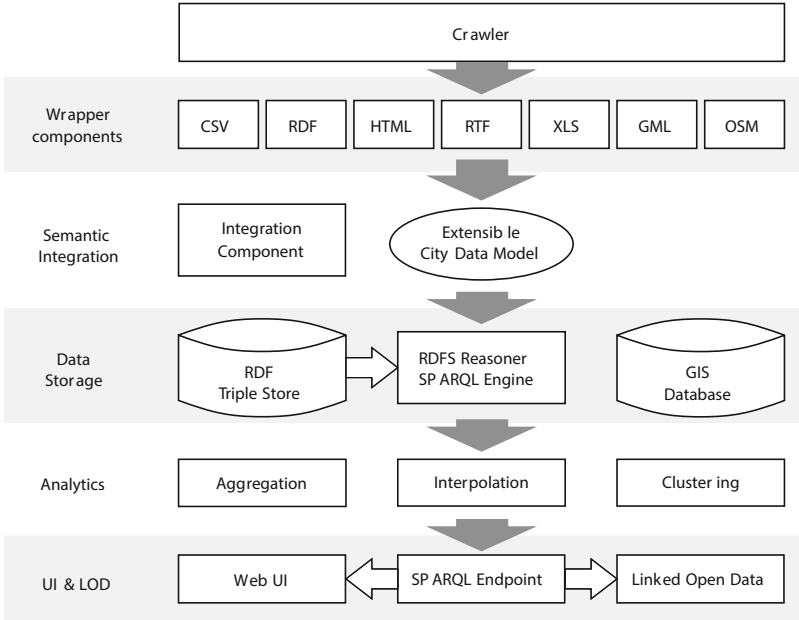


Fig. 1. City Data Pipeline architecture showing components for crawling wrapping, cleaning, integrating, and presenting information

by scripts. However, a more automated mapping process of new sources is an appealing extension for future work.

Wrapper Components. As a first step of data integration, a set of custom wrapper components parses the downloaded data and converts it to source-specific RDF. The set of wrapper components include a CSV wrapper for parsing and cleaning, a wrapper for extracting HTML tables, a wrapper for extracting tables of RTF documents, a wrapper for Excel sheets, and a wrapper for cleaning RDF data as well. All of these wrappers are customizable to cater for diverse source-specific issues. These wrappers convert the data to RDF and preprocess the data before integrating the data with the existing triple store. Preprocessing contains data cleansing tasks, i.e., unit conversions, number and data formatting, string encoding, and filtering invalid data (see [20]).

Semantic Integration (Ontology). To access a single indicator such as the population number, which is provided by several data sources, the semantic integration component *unifies the vocabulary* of the different data sources through an ontology (see Figure 2). The semantic integration component is partly implemented in the individual wrappers and partly by an RDFS [5] ontology (extended with capabilities for reasoning over numbers by using equations [2]) called *City Data Model* (see <http://citydata.wu.ac.at/ns#>). The ontology covers several aspects: spatial context (country, region, city, district), temporal context (valid-

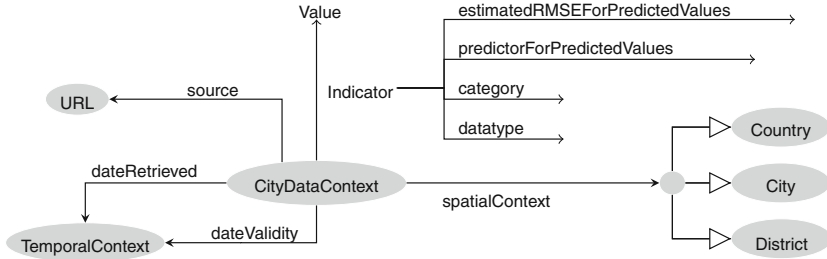


Fig. 2. Excerpt of the City Data Model ontology

ity, date retrieved), provenance (data source), terms of usage (license), and an extensible list of indicators.

Indicator is the super-property of all the indicator properties mapping **CityDataContexts** to actual values. Each **Indicator** of the ontology contains, a name, description, a unit of measurement, a data type, and is grouped into one of the following *categories*: (a) Demography, (b) Social Aspects, (c) Economic Aspects, (d) Training and Education, (e) Environment, (f) Travel and Transport, (g) Culture and Recreation, and (h) Geography. To integrate the source-specific indicators the ontology maps data-source-specific RDF properties to City Data Model properties, e.g., it maps `dbpedia:population` to `citydata:population` by an RDFS `subPropertyOf` property. A **CityDataContext** is an anchor connecting a set of data points to a spatial context, a temporal context, and a data source. When importing an input CSV file containing the indicators as columns and the cities as rows, each row corresponds to (at least) one **CityDataContext**. The **SpatialContext** class collects all resources with spatial dimension: country, province, region, city, and district. Furthermore entities of different granularity can be connected by the property `locatedIn`. The `dateValidity` property maps a **CityDataContext** to a point in time where the values are valid. Additionally the property `periodValidity` can indicate what the validity period is (possible values are biannual, annual, quarterly, monthly, weekly, daily, hourly or irregular). Whereas the `dateRetrieved` property records the date and time of the data set download. The `source` property links a **CityDataContext** to its data source.

Data Storage, Analytics, UI and LOD. To store the processed data we use Jena TDB as a *triple store* for RDF data. Subsequent subsystems can access the RDF data via a SPARQL interface (<http://citydata.wu.ac.at/>). The SPARQL engine provides RDFS reasoning support by query rewriting (including reasoning over numbers [2]).

The analytics layer includes tools to fill-in missing values by using statistical regression methods. Section 4 describes the missing value prediction in detail. The results are also stored in the RDF triple store and the SPARQL engine provides access to them. Section 7 explains the frontend, user interface, SPARQL endpoint, and publishing data as LOD. Bischof et al. [3] describe the system components in more detail.

4 Prediction of Missing Values

After integrating the different sources, we discovered a large number of missing values in our data sets. We identified two reasons for that:

- As shown in Table 1 and 2, we can observe a large ratio of missing values due to incomplete data published by the data providers;
- More severely, when we combine the different data sets even more missing values are introduced, since there is a fair amount of disjoint cities and indicators.

Base Methods. Our assumption is that every indicator has its own distribution (e.g., normal, Poisson) and relationship to other indicators. Hence, we aim to evaluate different regression methods and choose the best fitting model to predict the missing values. We measure the prediction accuracy by comparing the *normalized root mean squared error* in % (RMSE%) [29] of every regression method. In the field of Data Mining [29,10] (DM) various regression methods for prediction were developed. We chose the following three “standard” methods for our evaluation due to their robustness and general performance.

K-Nearest-Neighbour Regression (KNN), models denoted as M_{KNN} , is a wide-spread DM technique based on using a distance function to partition the instance space. As stated in [10], the algorithm is simple, easily understandable and reasonably scalable. KNN can be used in variants for clustering as well as regression.

Multiple Linear Regression (MLR), models denoted as M_{MLR} , has the goal to find a linear relationship between a target and several predictor variables. The linear relationship can be expressed as a regression line through the data points. The most common approach is *ordinary least squares* to measure and minimize the cumulated distances [10].

Random Forest Decision Trees (RFD), models denoted as M_{RFD} , involve the top-down segmentation of the data into multiple smaller regions represented by a tree with decision and leaf nodes. A random forest is generated by a large number of trees, which are built according to a random selection of attributes at each node. We use the algorithm introduced by Breiman [24].

Preprocessing. The preprocessing starts with the extraction of the base data set from our RDF triple store. We use SPARQL queries with the fixed period of 2004–2011 and produce an initial data set as a matrix with tuples of the form $\langle City, Indicator, Year, Value \rangle$. Based on the initial matrix, we perform the preprocessing as follows:

- Removing boolean and nominal columns, as well as all weather related data and sub-indicators in the U.N. data set, e.g., *housing units with 2 rooms*;
- Merging the dimensions year/city, resulting in $\langle City Year, Indicator, Value \rangle$;

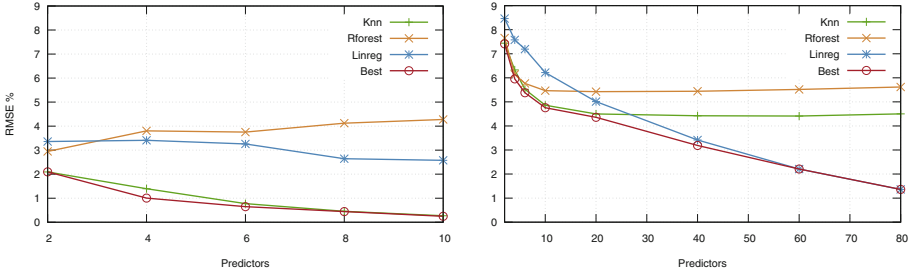
- Transposing the initial matrix by moving the indicators into the columns, resulting in tuples of the form $\langle City\ Year, Indicator_1\ Value, \dots, Indicator_n\ Value \rangle$;
- Deleting columns/rows which have a missing values ratio larger than 90%.

Our initial data set from UA, UN, and DBpedia contains 3 399 cities with 370 indicators. By merging city and year and transposing the matrix we create 13 482 city/year rows. And after deleting the cities/indicators with a missing values ratio larger than 90%, we have the final matrix of 4 438 rows (city/year) with 207 columns (indicators).

Approach 1 - Building Complete Subsets. In the first approach, we try to build models for a target indicator by directly using the available indicators as predictors. For this, we are using the correlation matrix of the data to find indicators which are suitable predictors. Subsequently, we build a complete subset from our data, i.e., we first perform a projection on our data table, keeping only the predictors and the specific target as columns. More detailed, our approach has the following steps on the initial data set, the matrix A_1 and a fixed number of predictors n (we test this approach on different n 's):

1. Select the target indicator I_T ;
2. Calculate the correlation matrix A_C of A_1 between I_T and the remaining indicators;
3. Create the submatrix A_2 of A_1 with I_T and the n "best" indicators (called the predictors). The predictors are selected according to the highest absolute correlation coefficients in A_C ;
4. Create the complete matrix A_3 by deleting all rows in A_2 with missing values;
5. Apply *stratified tenfold cross-validation* (see [29]) on A_3 to get ten training- and test sets. Then, train the models M_{KNN} , M_{MLR} , and M_{RFD} using the training sets. Finally, calculate the mean of the ten RMSE% based on the test set for each model and choose the best performing model M_{Best} ;
6. Use the method for M_{Best} to build a new model on A_2 for predicting the missing values of I_T .

The performance of the regression methods were evaluated for 2 to 10 predictors. Two regression methods have their best RMSE% with 10 indicators: 0.27% for KNN and 2.57% for MLR. Whereas RFD has the best RMSE% of 4.12% with 8 indicators. Figure 3a gives an overview of the results. By picking the best performing regression for every indicator (red line) the median RMSE% can be reduced only slightly. For 10 predictors the median RMSE% improves to 0.25% over KNN with 0.27%. Depending on n , we fill-in between 122 056 for 10 and 296 069 values for 2 predictors. For a single city and 10 predictors, the number of predicted values range from 7 to 1 770. The limited number of filled-in values is due to the restriction of using the complete matrix for the regression methods.



(a) Approach 1 (Building Complete Subsets)

(b) Approach 2 (PC Regression)

Fig. 3. Prediction results

Approach 2 - Principal Component Regression. In the second approach, we omit the direct use of indicators as predictors. Instead, we first perform a Principal Component Analysis (PCA) to reduce the number of dimensions of the data set and use the new compressed dimensions, called *principal components* (PCs) as predictors. As stated in [10], the PCA is a common technique for finding patterns in data of high dimensions. Parts of the evaluation is similar to Approach 1, but we have an additional step where we *impute* all the missing values with *neutral* values for the PCA. The neutral values are created according to the *regularized iterative PCA algorithm* described in [21]. This step is needed to perform the PCA on the entire data set. The following steps are evaluated having an initial data set A_1 as a matrix and a predefined number of predictors n (we test this approach also on different n 's):

1. Select the target indicator I_T ;
2. Impute the missing values in A_1 using the regularized iterative PCA algorithm resulting in matrix A_2 and remove the column with I_T ;
3. Perform the PCA on A_2 resulting in matrix A_3 of a maximum of 80 PCs;
4. Append the column of I_T to A_3 creating A_4 and calculate the correlation matrix A_C of A_4 between I_T and the PCs;
5. Create the submatrix A_5 of A_4 on the selection of the PCs with the highest absolute correlation coefficients and limit them by n ;
6. Create submatrix A_6 of A_5 for validation by deleting rows with missing values for I_T ;
7. Apply stratified tenfold cross-validation on A_6 with the Step 5 from Approach 1, which results in the best performing model M_{Best} ;
8. Use the method for M_{Best} to build a new model on A_5 (not A_6) for predicting the missing values of I_T .

Figure 3b shows the median RMSE% for KNN, RFD, MLR, and the best method with an increasing number of predictors. For 80 predictors MLR performs best with a median RMSE% of 1.36%, where KNN (resp. RFD) has a median RMSE% of 4.50% (resp. 5.62%). MLR improves steady up to 80 predictors. KNN provides good results for a lower number of predictors, but starts flattening with 20

predictors. An increasing number of k could improve the result though. The red line in Figure 3b shows the median RMSE% with the best regression method chosen. Up to 60 predictors, the overall results improves by selecting the best performing method (for each indicator). The best median RMSE% of 1.36% is reached with 80 predictors. For this, MLR is predominant but still 14 out of 207 indicators are predicted by KNN.

As mentioned, we have two quality measurements to evaluate our approaches. First, it is important to build models which are able to predict many (preferably all) missing values. Second, the prediction accuracy of the models is essential, so that the Open City Data Pipeline can fulfill its purpose of publishing high-quality, accurate data and predictions. Prediction accuracy is higher in Approach 1 than 2 (for 4 to 10 predictors), which we relate to the reduced size of the data set. However in Approach 1, we fill-in at the maximum 296 069 values with 2 predictors (median RMSE% of 2.09%), which is about 66% of Approach 2. Due to the reduced number of predictions, we will apply Approach 2 for publishing the filled-in missing values.

5 Cross Data Set Prediction

Our initial overall matrix has 13 482 city/year rows and 369 columns, which are reduced after deleting all with a missing values ratio of 90% to the matrix of 4 438 rows and 207 columns. Cross Data Set Predictions (CDP) aims to fill the gap of the 162 columns mainly caused by the disjointness of indicators/cities in the data sets (e.g., UN and UA). As seen in Figure 4, there are two areas which are not covered, the first is the UA cities for the UN indicators and the second is the UN cities for the UA indicators. The success of the CDP approach depends on one data set, which has a reasonable amount of

overlapping cities with the other data sets. At the time of writing the UN data set seems the most promising covering cities of the whole world.

For CDP, we always select one data set (e.g., UN), called the source data set S , and predict into another data set (e.g., UA), called the target data set T , denoted as $S \rightarrow T$. We evaluate again the different base regression methods and choose the best fitting model for prediction. The preprocessing is altered so we only delete columns and rows which are entirely empty. Since Approach 1 needs a complete matrix, we only consider Approach 2 and modify it accordingly. We

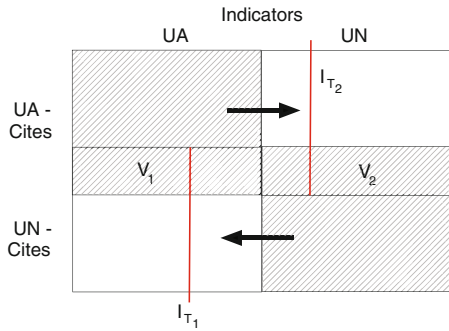


Fig. 4. Predicting I_{T_1} (resp. I_{T_2}) from the UN (resp. UA) data set

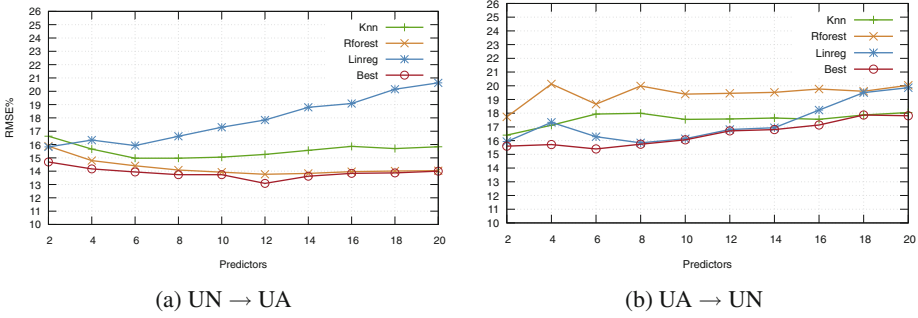


Fig. 5. Cross data set prediction results

start in the CDP approach with the initial source and target data sets A_S and A_T . The following steps are evaluated for a different number of predictors n :

1. Select the target indicator I_T from T ;
2. Impute the missing values in A_S using the regularized iterative PCA algorithm resulting in matrix A_{S_2} ;
3. Perform the PCA on A_{S_2} resulting in matrix A_{S_3} of a maximum of 40 PCs;
4. Append the column I_T to A_{S_3} creating A_{S_4} and calculate the correlation matrix A_{S_C} between I_T and the PCs;
5. Create the submatrix A_{S_5} of A_{S_4} on the selection of the PCs with the highest absolute correlation coefficients and limit them by n ;
6. Create validation submatrix A_{S_6} of A_{S_5} by deleting rows with missing values for I_T ;
7. Apply stratified *fivefold* cross-validation on A_{S_6} similar to Step 7 from Approach 2, which results in the best performing model M_{Best} ³;
8. Use the method for M_{Best} to build a model on A_{S_5} to predict missing values of I_T .

Note that the validation of Step 7 is performed on the set V_1 or V_2 of cities overlapping S and T . We ignore a target indicator if the set is empty, since we can not determine the quality of our prediction. The amount of overlapping cities with values ranging for T as UA (resp. T as UN) from 16 (resp. 11) to 1194 (resp. 1429) with an average of 445 (resp. 88) cities. We performed the CDP from UN → UA and the results are shown in Figure 5a. RFD performs best for with a median RMSE% of 13.76% for 12 predictors. The median RMSE% of M_{Best} is 13.08% with 12 predictors and always very close to the RFD results. With more than 12 predictors, the result does not improve anymore. The population related indicators are predicted best (e.g., *Population male* has a RMSE% of 4.86%), weather related indicators are worst (e.g., *Total hours of sunshine per day* has a RMSE% of 1176.36%). The reason lies within in the UN source data

³ Cross-validation is reduced from ten- to fivefold, so the test set is large enough.

set, where three indicators *population*, *population female*, and *population male* are predominant. For the UA \rightarrow UN predictions, shown in Figure 5b, the results are best with a median RMSE% of 15.40% with 6 predictors. More predictors do not improve the result, whereas MLR performs best overall. The most frequent indicators, again *population*, are reasonable predicted, whereas most of the other UN indicators can not be properly validated due to the low number of overlapping cities (avg. of 88). If we apply the threshold RMSE% of 7% for publishing the predicted values, we are able to predict for UN \rightarrow UA: 34 out of 210 indicators; and for UA \rightarrow UN: 2 out of 142 indicators. The results for UN \rightarrow UA are satisfying, since we are able to predict 34 UA indicators for UN cities, the picture is dimmer for UA \rightarrow UN, where only a few indicators are below our threshold.

6 Learning Ontology Mappings from Indicator Values

So far we used regression methods to predict missing values over the whole integrated data set as well as across different source data sets. But regression can also be a means to learn ontology axioms to express *dependencies between pairs of indicators* from different data sources. By exploiting these dependencies a reasoner can give more complete answers without materializing a potentially large number of new values beforehand.

The models expressing these dependencies should be intuitive, i.e., comprehensible by a domain expert, and should allow derivation of new values. We focus on pairs of indicators to cover several cases: (1) the same indicator, (2) the same indicator with different units (for example area in km² in mile²), (3) somewhat normalized indicators. Since we are interested in simple models and numerical data, we model the dependencies by *linear equations* containing two indicators from two different sources. Furthermore some data sources (UA) already publish the equations used to compute some of their indicators such as population density. Because of high dependencies of indicators within a data set we only consider pairs of indicators from different data sets.

As a special case we consider pairs of *equivalent indicators*, e.g., many data sets have an indicator for population. We could model this dependency as simple equation $p_1 = p_2$ but ontology languages already provide axioms to express the equivalence of two properties which in turn any standard Semantic Web reasoner can use to get more complete data. OWL 2 provides the `EquivalentDataProperties` axiom, while RDFS allows modeling equivalent properties by a pair of symmetric `subPropertyOf` axioms.

We use linear regression to compute the dependencies. In general linear regression estimates the *intercept* a and the *slope* b for a linear equation $y = a + bx$ where x is the *independent variable* (predictor indicator) and y the *dependent variable* (response indicator). Thus it tries to fit a line to the data with as little error as possible. A popular error measure is *least-squares* which is known to suffer heavily from outliers [29] which is also the case for our data set. Thus we perform a *robust regression*, which is computationally more expensive but handles both horizontal and vertical outliers better. We use the R function `rlm`

of the MASS library which implements robust regression by iterated re-weighted least squares and Huber weights [27]. Applying robust regression to all pairs of indicators from UN and Urban Audit results theoretically in 214×148 linear models. Many pairs of indicators have no *complete observations*, i.e., cities with values for both indicators, for which regression can not be applied.

For the ontology we want to keep only those linear models for which the pair of indicators has a strong dependency. We first filter out all dependencies which have less than 100 complete observations. Next we compute a correlation matrix of all indicator pairs to quantify the indicator dependency. Since the standard Pearson correlation assumes a normal distribution, which the indicators not necessarily follow, we use the non-parametric *Kendall rank correlation coefficient* τ implemented in the R function `cor` [19], instead. We filter out all models with a correlation less than 0.7.

For finding equivalent properties we perform a second linear regression without an intercept, i.e., forcing the linear model through the origin. As before we filter out linear models with low correlation or insufficient complete observations. If the slope of this second linear model is 1 ± 0.01 , then we consider the indicator pair as equivalent.

When performing this approach on the UN and UA data sets we get 98 linear equations, 4 of which indicate equivalent indicator pairs published in our ontology. Neither OWL nor RDFS provide a means to express linear equations except property equivalences (represented as sketched above). Thus, for the remaining linearly dependent indicator pairs we use the notation as in previous work [2] to express the respective mappings in our ontology. Further the ontology contains the number of complete observations and the correlation for each new axiom as annotations. Detecting more complex relationships between a *set* of indicators from one datasource and a single indicator from a second dataset (which would be expressible as equations using the notation of [2]) is on our agenda.

7 Publishing as Linked Data

Linked Open Data. The resources (cities) and properties in the City Data namespace (<http://citydata.wu.ac.at/>) are published according to the Linked Data principles. The ontology (as described in Section 3), contains all City Data property and class descriptions. Each city is assigned a dereferencable URI, e.g., <http://citydata.wu.ac.at/resource/Ljubljana> for the capital of Slovenia. Depending on the HTTP `Accept` header the server will return either an HTML, RDF/XML, or Turtle representation after a HTTP 303 redirect. The city resources are linked to the LOD cloud via `owl:sameAs` to the corresponding DBpedia resources.

Predictions. The prediction workflow is based on the current data in the triple store. The *preprocessing* is written in `Python` and *prediction* and *evaluation* is developed in `R` [19] using its “standard” packages. As mentioned before, we only publish the predicted values from Approach 2. After the best regression method

is selected for a particular indicator, we use this method to fill-in all the missing values and publish them as a *new* indicator with a *prefix* in the **CityDataContext**. We also add the the source and the year for the prediction. The threshold for publishing is a RMSE% of 7% with 80 predictors. This leads to 6 indicators (e.g. *price of a m³ of domestic water in EUR*) being dropped. We then introduce two new properties describing for each indicator the quality of the prediction by the median RMSE% and the regression method used. In future work, we aim to publish the data using the PROV Data Model [8].

Interface. A simple Java powered web interface allows users to select exactly which subset of the data should be shown. The interface provides programmatic access via HTTP GET to allow external tools such as data visualization frameworks, to query the database. The web application communicates with the Jena triple store via SPARQL 1.1. Users can select one or more of the 450 *indicators* sorted by categories like *Demography*, *Geography*, *Social Aspects*, or *Environment*. The list also shows how many data points are available per indicator and for how many cities data points are available for this indicator. Next the user can select one or several of more than 5 260 *cities* for which we collected data. For a few cities we even have information on the individual districts available. In these cases the user can select one or several of the districts. Optionally the user can specify a *temporal context*, for which year the database should be queried. This feature allows to compare several cities with each other at a certain point of time instead of listing data of all available times.

8 Lessons Learnt and Related Work

We emphasize that our work is not a “Semantics in-use” paper in the classical sense of applying Semantic Web technologies to solve a use case, but rather a demonstration that a portfolio of statistical methods *in combination* with semantic technologies for data integration helps to collect, enrich and serve domain-specific data in a reusable way for *further applications* of the LOD cloud to be developed on top. While there are practical use cases within Siemens, such as studies like the Green City Index [6] which can benefit from an up-to-date data repository for city data, we are looking forward to diverse other applications on top of our collection by others. Also, we have demonstrated that building a domain-specific Open Data pipeline is feasible and enabled by Semantic Web technologies. We envision that such an approach may be worthwhile for other domains as well as a multiplier to leverage usage of Open Data: for instance similar data pipelines could be built for business intelligence, investment use cases for company data, or finance data. For publishing the prediction as LOD, we set a threshold RMSE% of 7%, which could be adjusted according to the domain of use.

Lessons Learnt. In the wrapper component, integrating cities and indicators for a new data set (often CSV tables) is still a slow manual process and needs custom scripting. The entity recognition for cities and the ontology learning techniques from Section 6 provide a first automation step, where indicators of new data sets can be mapped to existing indicators. This approach is similar

to instance based mapping learning techniques also used in ontology matching (cf. [7]). In the analytics and query component, we have had to deal with sparse data sets with many missing values, which is a drawback for analyzing and reusing the data. By applying the PCA-based Approach 2, using a basket of *standard* DM techniques without customization, we reach a good quality for predictions (overall RMSE% of 1.36%) and are able to fill large gaps of the missing values. However, Approach 2 does not tackle the gap of disjoint cities/indicators, which is addressed by extending it to the CDP approach, where we predict from one single data set into another. We applied CDP for predicting $UA \rightarrow UN$ and $UN \rightarrow UA$ and discovered reasonable results for the first but unsatisfying for the second direction. The cause for the unsatisfying results can be found in the UN data set with sufficient values for only three population-related indicators. For the CDP approach to succeed, we need one *base* data set which covers a wider range of cities/indicators; this is not the case yet.

Related Work. *QuerioCity* [13] is a platform to integrate static and continuous data with Semantic Web tools. While it uses partly similar technologies, it works as a single city platform and not as a data collection of many cities and concentrates on data integration. We focus on predicting missing values, and publishing the outcomes as Linked Data. The EU project *CitySDK* (<http://www.citysdk.eu/>) provides unifying APIs, including a Linked Data API for mobility and geo data usable across cities. These reusable APIs enable developers to create portable applications and ease service provisioning for city administrators. If enough cities adopt CitySDK, its APIs can become a valuable data source for the Open City Data Pipeline as well. Regarding the methods, works of Paulheim et al. [16,17,18] are closely related, however they focus on unsupervised DM approaches of unspecified features from Linked Data instead of filling-in missing values for specific attributes. The work by Nickel et al. [14] focuses on relational learning, i.e., rather learning object relations than predicting numeric attribute values. The work in [11] also integrates statistical Linked Data, however it is mainly concerned with query rewriting and less with missing values. The Open City Data Pipeline uses techniques from ETL frameworks (cf. [25]) and DM tools (e.g., [9]) which are *general* technologies and build our base techniques. The main difference to a plain ETL and DM approach concerns (a) the ontology-based integration with query capabilities and continuous integration in the LOD cloud, (b) the ontology-learning capabilities, and (c) using the axioms of the ontology to validate the prediction results by the data type and ranges.

9 Conclusions and Future Work

In this paper we have presented the *Open City Data Pipeline*, an extensible platform for collecting, integrating, and predicting open city data from several data providers including DBpedia and Urban Audit. We have developed several components including a data crawler, wrappers, an ontology-based integration platform, and a missing value prediction module. Having sparse data sets, the prediction of missing values is a crucial component. For this, we have developed

two approaches, one based on predicting a target indicator directly from other indicators, and one based on predictors from components calculated by Principal Components Analysis (PCA). We applied for both approaches three basic regression methods and selected the best performing one. They were compared regarding the number of filled-in values and prediction accuracy, concluding that the PCA-based approach will be used for future work. Filled-in missing values are then published as LOD for further use. In case of a large disjointness regarding indicators/cities, we extended the second approach to Cross Data Set Predictions (CDP).

Our future work includes extensions of the presented data sets, methods, and the system itself. Regarding the data sets, we already mention several sources, e.g., the Carbon Disclosure Project, which are needed to cover a wider range of cities worldwide. As to the methods, CDP has to be evaluated with more data sets to further evaluate the performance of CDP and find the threshold of indicators/cities with sufficient *overlapping* values. We also aim to extend our basket of base methods with other well established regression methods. Promising candidates are Support Vector Machines [22], Neural Networks, and Bayesian Generalized Linear Model [28]. Moreover, we plan to publish more details on the best regression method per indicator as part of our ontology: so far, we only indicate the method and estimated RMSE%, whereas further details such as used parameters and regression models would be needed to reproduce and optimize our predictions. Ontologies such as [12] could serve as a starting point here. We also plan to connect our platform to the Linked Geo Data Knowledge Base [23] including OpenStreetMap (OSM) data: based on such data, new indicators could be directly calculated, e.g., the size of public green space by aggregating all the parks. Furthermore, we are in the process of improving the user interface to make the application easier to use. For this we investigate several libraries for more advanced information visualization.

Acknowledgments. This work was supported by the Vienna Science and Technology Fund (WWTF) project ICT12-15 and the EU project CityPulse FP7-609035.

References

1. Bettencourt, L.M.A., Lobo, J., Helbing, D., Kühnert, C., West, G.B.: Growth, innovation, scaling, and the pace of life in cities. *Proc. of the National Academy of Sciences of the United States of America* **104**(17), 7301–7306 (2007)
2. Bischof, S., Polleres, A.: RDFS with attribute equations via SPARQL rewriting. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) *ESWC 2013. LNCS*, vol. 7882, pp. 335–350. Springer, Heidelberg (2013)
3. Bischof, S., Polleres, A., Sperl, S.: City data pipeline. In: *Proc. of the I-SEMANTICS 2013 Posters & Demonstrations Track*, pp. 45–49 (2013)
4. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: *DBpedia - A crystallization point for the web of data*. *J. Web. Sem.* **7**(3), 154–165 (2009)
5. Brickley, D., Guha, R., (eds.): *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, W3C (2004)

6. Economist Intelligence Unit (ed.): The Green City Index. Siemens AG (2012)
7. Euzenat, J., Shvaiko, P.: Ontology matching, 2nd edn. Springer (2013)
8. Gil, Y., Miles, S.: PROV Model Primer. W3C Note, W3C (2013)
9. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. *SIGKDD Explor. Newsl.* **11**(1), 10–18 (2009)
10. Han, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann Publishers Inc. (2012)
11. Kämpgen, B., O’Riain, S., Harth, A.: Interacting with statistical linked data via OLAP operations. In: Simperl, E., Norton, B., Mladenic, D., Valle, E.D., Fundulaki, I., Passant, A., Troncy, R. (eds.) *ESWC 2012*. LNCS, vol. 7540, pp. 87–101. Springer, Heidelberg (2015)
12. Keet, C.M., Lawrynowicz, A., d’Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., Hilario, M.: The data mining OPTimization ontology. *Web Semantics: Science, Services and Agents on the World Wide Web* **32**, 43–53 (2015)
13. Lopez, V., Kotoulas, S., Sbodio, M.L., Stephenson, M., Gkoulalas-Divanis, A., Aonghusa, P.M.: QuerioCity: a linked data platform for urban information management. 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. 148–163. Springer, Heidelberg (2012)
14. Nickel, M., Tresp, V., Kriegel, H.: Factorizing YAGO: scalable machine learning for linked data. In: *Proc. of WWW 2012*, pp. 271–280 (2012)
15. Office for Official Publications of the European Communities: Urban Audit. *Methodological Handbook* (2004)
16. Paulheim, H.: Generating possible interpretations for statistics from linked open data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012*. LNCS, vol. 7295, pp. 560–574. Springer, Heidelberg (2012)
17. Paulheim, H., Fürnkranz, J.: Unsupervised generation of data mining features from linked open data. In: *Proc. of WIMS 2012*, p. 31. ACM (2012)
18. Paulheim, H., Ristoski, P., Mitichkin, E., Bizer, C.: Data mining with background knowledge from the web. In: *Proc. of the 5th RapidMiner World* (2014)
19. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2009)
20. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.* **23**(4), 3–13 (2000)
21. Roweis, S.T.: EM algorithms for PCA and SPCA. In: *Advances in Neural Information Processing Systems, (NIPS 1997)*, vol. 10, pp. 626–632 (1997)
22. Sanchez, V.: Advanced support vector machines and kernel methods. *Neurocomputing* **55**(1–2), 5–20 (2003)
23. Stadler, C., Lehmann, J., Höffner, K., Auer, S.: LinkedGeoData: A core for a web of spatial open data. *Semantic Web* **3**(4), 333–354 (2012)
24. Statistics, L.B., Breiman, L.: Random forests. In: *Machine Learning*, pp. 5–32 (2001)
25. Thomsen, C., Pedersen, T.B.: A survey of open source tools for business intelligence. In: Tjoa, A.M., Trujillo, J. (eds.) *DaWaK 2005*. LNCS, vol. 3589, pp. 74–84. Springer, Heidelberg (2005)

26. U.S. Census Bureau: County and City Data Book 2007 (2007). <https://www.census.gov/compendia/databooks/>
27. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S., 4th edn. Springer (2002)
28. West, M., Harrison, P.J., Migon, H.S.: Dynamic generalized linear models and bayesian forecasting. *Journal of the American Statistical Association* **80**(389), 73–83 (1985)
29. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd edn. Morgan Kaufmann Publishers Inc. (2011)

ASSESS — Automatic Self-Assessment Using Linked Data

Lorenz Bühmann, Ricardo Usbeck^(✉), and Axel-Cyrille Ngonga Ngomo

IFI/AKSW, Universität Leipzig, Leipzig, Germany
{buehmann,usbeck,ngonga}@informatik.uni-leipzig.de

Abstract. The Linked Open Data Cloud is a goldmine for creating open and low-cost educational applications: First, it contains open knowledge of encyclopedic nature on a large number of real-world entities. Moreover, the data being structured ensures that the data is both human- and machine-readable. Finally, the openness of the data and the use of RDF as standard format facilitate the development of applications that can be ported across different domains with ease. However, RDF is still unknown to most members of the target audience of educational applications. Thus, Linked Data has commonly been used for the description or annotation of educational data. Yet, Linked Data has (to the best of our knowledge) never been used as direct source of educational material. With ASSESS, we demonstrate that Linked Data can be used as a source for the automatic generation of educational material. By using innovative RDF verbalization and entity summarization technology, we bridge between natural language and RDF. We then use RDF data directly to generate quizzes which encompass questions of different types on user-defined domains of interest. By these means, we enable learners to generate self-assessment tests on domains of interest. Our evaluation shows that ASSESS generates high-quality English questions. Moreover, our usability evaluation suggests that our interface can be used intuitively. Finally, our test on DBpedia shows that our approach can be deployed on very large knowledge bases.

1 Introduction

The amount of RDF data available across the globe has grown significantly over the last years. As pointed out in previous works, a large portion of the open data available in this format is encyclopedic in nature [9]. While RDF data is being used for the description and annotation of education material and websites, this data format has (to the best of our knowledge) never been used as source of educational material. This is simply due to the target audience of educational material not being familiar with this technology. However, given (1) the large number of domains already described as RDF, (2) the large number of websites annotated with RDFa and (3) the simple structure of RDF statements, RDF

L. Bühmann and R. Usbeck—Both authors contributed equally to this work.

knowledge bases seem to be an optimal source or the automatic generation of educational material.

With ASSESS, we address the automatic generation of education material directly from RDF data. With this contribution, we aim to show that the Web of Data has the potential to contribute to the dissemination of educational materials across borders, especially to the less privileged, an endeavor in line with platforms such as Coursera¹ and EdX.² ASSESS' main contribution is the automatic generation of self-assessment tests containing a variety of user-selected question types directly from RDF. The intended users are consequently (1) persons who aim to assess their knowledge on a particular domain of interest (we call these persons *learners*) and (2) persons in charge of assessing the knowledge of learners (called *teachers* in the following). Manifold usage scenarios can be envisaged for the approach, including the preparation of tests on a particular domain, the training of employees on novel products, the generation of tests for exams, casual gaming, the extension of ones general knowledge and many more.

ASSESS achieves its goal by providing innovative solutions to the following:

1. *Automatic verbalization of RDF graphs*: While RDF benefits the applications by making the generation of questions easy, most users do not understand RDF. We are able to hide the RDF data completely from the end user while making direct use of this data to generate questions of different types. Our verbalization approach is generic and thus independent of the underlying knowledge base. In addition, it is time-efficient as demonstrated by our evaluation on selected knowledge bases.
2. *Entity summarization*: Our entity summarization approach allows detecting key properties for describing a resource. This allows for the generation of succinct (natural-language) descriptions of resources that can then be transformed into questions of different difficulty.
3. *RDF fragment extraction*: We use Concise Bound Descriptions (CBDs) of RDF resources to generate fragments of the input knowledge that contain resources which abide by the domain description provided by the user. These fragments are used both for the generation of questions and of hints towards the answer of questions.

By using these techniques our tool enables users to generate customized quizzes pertaining to a user-defined domain. Currently, our tool supports the generation of Yes/No, Jeopardy-style and Multiple-Choice questions. Other types of questions are being added. Our system is now a working system deployed at <http://assess.aksw.org/demo/>. The system is open-source and abides by the GPL license. The server code is part of the SemWeb2NL project and can thus be found at <http://github.com/AKSW/SemWeb2NL>. The client code can be found at <https://github.com/AKSW/ASSESS>. Throughout this paper, we assume that the learner is a medical student aiming to test his/her knowledge about the human circulatory system from DBpedia.

¹ <https://www.coursera.org/>

² <https://www.edx.org/>

2 System Description

ASSESS is an application for the generation of questions and answers from RDF. It abides by the client/server paradigm, where the client is a user interface that displays the information returned by the server. The communication between client and server is ensured by REST interfaces which consume and generate JSON messages. The server side of ASSESS consists of four layers.

2.1 Data Layer

The *data layer* ensures the communication with the knowledge base(s) and implements the extraction of CBDs for resources. The data layer can deal with one or several knowledge bases. As input, the layer expects a set of classes or an OWL class expression E . When provided with this description of the domain of interest for a user, the data layer selects the fragment of the underlying knowledge base which pertains to the users domain description. To this end, it begins by selecting all the resources that abide by the description E . For example, if E is simply the class `:Vein`, we begin by retrieving all instances of the class, including for example the *inferior vena cava* (IVC). The data layer then retrieves the CBDs of each of these resources via SPARQL queries. Depending on the required difficulty of the questions (which the user can set), the layer then performs further CBD retrieval for all the resources in the graph for which we do not yet have a CBD. For example, the CBD of the *abdominal aorta*, the artery of which the IVC is the venous counterpart, would be retrieved if a second round of CBD retrievals were required. By these means, ASSESS supports the generation of deeper queries over several hops.

For the sake of scalability, the data layer also implements several in-memory and hard drive-driven caching solutions. By these means, we ensure that our approach scales to large knowledge bases such as DBpedia (see Section 4 for our evaluation results). The output of the data layer is a graph which contains a set of CBD out of which the questions are generated. Note that we do not target data quality with ASSESS (see, e.g., [15] for approaches that do this). Instead, we assume that the underlying knowledge base is a curated knowledge base. We argue that such knowledge bases will be increasingly available in the near future (e.g., through projects such as SlideWiki³ or LinkedUp⁴). Still, we aim to add mechanisms for processing user feedback (e.g., flags for triples that lead to incorrect questions) in the near future.

2.2 Natural-Language Generation Layer

This layer provides the mechanisms necessary to verbalize RDF triples, SPARQL basic graph patterns (BGPs) as well as whole SPARQL queries. This layer is an extension of the SPARQL2NL framework [13], which has been shown to

³ <http://slidewiki.org/>

⁴ <http://linkedup-challenge.org/>

achieve high-quality verbalizations for the Question Answering on Linked Data (QALD) benchmark. This makes the framework particularly well suited for the generation of natural-language questions out of RDF. This layer contains two main components: a verbalizer and a realizer. Given a set of triples or BGPs, the verbalizer can generate a sentence parse tree by employing syntactic and semantic rules. This sentence parse tree is then the input for the realizer, which generates natural language.

The realization of a triple pattern or of a triple $\mathbf{s} \ \mathbf{p} \ \mathbf{o}$ depends mostly on the verbalization of the predicate \mathbf{p} . If \mathbf{p} can be realized as a noun phrase, then a possessive clause can be used to express the semantics of $\mathbf{s} \ \mathbf{p} \ \mathbf{o}$, as shown in 1. For example, if \mathbf{p} is a relational noun like `author`, then the verbalization is `?x's author is ?y`. In case \mathbf{p} 's realization is a verb, then the triple can be verbalized as given in 2. For example, if \mathbf{p} is the verb `write`, then the verbalization is `?x writes ?y`.

1. $\rho(\mathbf{s} \ \mathbf{p} \ \mathbf{o}) \Rightarrow \text{poss}(\rho(\mathbf{p}), \rho(\mathbf{s})) \wedge \text{subj}(\text{BE}, \rho(\mathbf{p})) \wedge \text{dobj}(\text{BE}, \rho(\mathbf{o}))$
2. $\rho(\mathbf{s} \ \mathbf{p} \ \mathbf{o}) \Rightarrow \text{subj}(\rho(\mathbf{p}), \rho(\mathbf{s})) \wedge \text{dobj}(\rho(\mathbf{p}), \rho(\mathbf{o}))$

The combination of verbalizations of single triple patterns is also carried out using rules. For example, the object grouping rule (also known as *backward conjunction reduction*) collapses the subjects of two sentences $\rho(\mathbf{s}_1\mathbf{p}_1\mathbf{o}_1)$ and $\rho(\mathbf{s}_2\mathbf{p}_2\mathbf{o}_2)$ if the realizations of the verbs and objects of the sentences are the same:

$$\rho(\mathbf{o}_1) = \rho(\mathbf{o}_2) \wedge \rho(\mathbf{v}_1) = \rho(\mathbf{v}_2) \wedge \text{cc}(\mathbf{v}_1, \text{coord}) \\ \Rightarrow \text{root}(Y, \text{PLURAL}(\mathbf{v}_1)) \wedge \text{subj}(\mathbf{v}_1, \text{coord}(\mathbf{s}_1, \mathbf{s}_2)) \wedge \text{dobj}(\mathbf{v}_1, \mathbf{o}_1),$$

where the `coord` \in `{and, or}` is the coordination combining the input sentences, and `coord` \in `{conj, disj}` is the corresponding coordination combining the subjects. Other rules can be found in the SemWeb2NL code as well as in [13].

2.3 Entity Summarization

This layer provides mechanisms for the detection of the most relevant predicates when aiming to describe a given resource \mathbf{s} . To this end, we begin by finding the most specific class to which this resource belongs, i.e., the class \mathbf{C} that is such that \mathbf{s} is an instance of \mathbf{C} but of none of its subclasses. Now, in addition to labeling predicates such as `rdfs:label`, we find the properties that are most frequently used in combination with instances of \mathbf{C} . The properties are sorted in descending order of usage frequency and the top k properties are retrieved (where $k = 5$ is our default setting). The approach relies on property frequencies in combination with labeling properties [6] to detect the most important properties for a resource. Note that by choosing only k properties, we can deal with large CBDs. Moreover, the system discards resources whose description is too short. We then retrieve (as far as they exist) the objects of these k predicates w.r.t. \mathbf{s} . Note that we select at most three of the objects of each of the k properties selected by our approach. The resulting set of triples is the summary for \mathbf{s} . The user can obviously choose the properties of interest for his/her learning/teaching

goal. For example, only the blood flow through veins might be of importance for our user, in which case he would choose the `dbo:flow` property as target property for learning. The generated summary is forwarded to the next layer.

2.4 Question and Answer Generation Layer

The last layer contains modules which implement a generic interface for question and answer generation. Each of these modules takes a set of resources as input and generates (1) a question, (2) a set of correct answers and optionally (3) a set of wrong answers. Note that while we currently support three types of questions, a multitude of other question types can be envisaged. The types of questions we implement at the moment are:

- *Yes/No questions*: These are statements to which the user has to state whether they are true or false (see Figure 1 for an example). Giving an entity summary, we begin by selecting a triple $(s\ p\ o)$. Then, we randomly decide on whether the question to be generated should have the answer true or false. If true is selected, then we generate “**Is the following statement correct:**”, followed by $\rho(s\ p\ o)$. Else, we begin by generating a wrong assertion by replacing the object of the triple with an object o' such that $(s\ p\ o')$ does not belong to the input knowledge base and there exists a s' for which $(s'\ p\ o')$ holds in the input knowledge base. Note that while we actually assume a closed world here, this approach works well in real use cases.

Fig. 1. Yes/No question

- *Jeopardy questions*: Jeopardy questions describe an entity without naming the entity itself. To generate such questions, we begin with a summary of the resource to describe. We then replaced $\rho(s)$ by expressions such as **This** $\rho(C)$ (where C is a class to which s belongs) or pronouns (it, he, she) depending on the type of resource at hand. The result is a summary of the entity that

does not contain its label, e.g., `This anatomical structure's dorlands prefix is v 05 and its dorlands suffix is 12851372`. The correct answer(s) are then the resource(s) which share with `s` all predicate-object pairs used to generate the description of `s` (in this case the hepatic portal vein). Negative answers are generated by looking for resources that share a large number of property-values pairs with positive answers but are not elements of the set of correct answers (e.g., the splenic vein).

Question 7/10
0 correct answers
00 : 01 : 05

Which vein matches the following description: This anatomical structure's dorlands prefix is v 05 and its dorlands suffix is 12851372. Its gray page is 681 and its gray subject is 174.

Hepatic portal vein.

Left gastroepiploic vein.

Superior rectal vein.

Pancreaticoduodenal veins.

Splenic vein.

Next

Fig. 2. Jeopardy question in ASSESS' client

- *Multiple-choice question:* This type of questions is commonly used in automatic assessment tests. Here, we generate questions by verbalizing a pair (s, p) , e.g., `jugular vein's artery`. Then, we retrieve all `o` such that (s, p, o) holds. These are the correct answers to the question. In addition, we find `o'` such that there (s, p, o') is not in the knowledge base but there is at least one other resource `s'` with (s', p, o) . The user is then to choose a subset of the provided answers (see Figure 3).

Each of the layers provides a REST interface with which it can communicate with other applications. Thus, the server side of ASSESS can be integrated in any educational application that requires the verbalization of the RDF triples, BGP's or SPARQL queries.

The current client side of the application was designed with portability in mind and can be used on both stationary and mobile devices. The main motivation behind this design choice was that in less privileged countries, mobile

Question 2/10
0 correct answers
00 : 09 : 19

Please select Internal jugular vein's artery.

Common carotid artery.

Internal thoracic artery.

Axillary artery.

Tubal branches of ovarian artery.

Lesser palatine arteries.

Next

Fig. 3. Multiple-Choice question in ASSESS' client

devices are the instrument of choice to access the Web [2]. Thus, our provision of a ubiquitous-device-friendly interface has the potential to support the utilization of our application across the world. We demonstrate the usability of ASSESS by deploying on DBpedia as it is a large knowledge base with a complex ontology.

3 Distinguishing Features

Overall, the most distinguishing feature of ASSESS is that it implements a bridge between RDF, SPARQL and natural language (in this case English). Therewith, it makes RDF amenable to be a source of content (and not only of metadata or descriptions) for educational applications. Still, the education material generated by content can be consumed by users with all possible range of expertise in Semantic Web technologies. In the following, we present how ASSESS addresses the challenge of providing useful educational content using Linked Data:

- **Innovation in Education:** ASSESS addresses the automatic generation of tests out of structured data, especially RDF. Its main innovation w.r.t. to education software pertains to the generation of the test questions directly out of RDF data. So far, Linked Data has been most commonly used to describe educational data. To the best of our knowledge, ASSESS is the first approach that uses state-of-the-art technology to extract and verbalize questions directly out of RDF. The resulting tests are available in the form of different types of quizzes, allowing users to prepare for specific types of tests. The learning process is improved in several ways. First, the users can tailor the tests to exactly the domain in which they are interested by describing

The screenshot shows a web interface for configuring a question set. At the top, a blue header reads "Create new set of questions". Below this, the "Domain" section features a tree view with the following structure:

- University (unchecked)
- Valley (unchecked)
- Vein (checked)
 - artery (checked)
 - dorlandsPrefix (checked)
 - dorlandsSuffix (checked)
 - drainsFrom (checked)
 - drainsTo (checked)
 - grayPage (checked)
 - graySubject (checked)
 - meshName (checked)
 - meshNumber (checked)
- Venue (unchecked)
- VideoGame (unchecked)

Below the domain tree, the "Number of Questions" is set to 10 in a text input field. The "Question Types" section has three buttons: "Multiple Choice", "Jeopardy", and "Yes/No". At the bottom, a blue button labeled "Generate Questions »" is visible.

Fig. 4. Configuration window of ASSESS

both the type of resources as well as the properties of these resources for which questions are to be generated. Moreover, the users can specify the type of questions they wish to answer as well as the number of questions. Therewith, users can easily tailor the assessment to exactly their needs. The online availability of the tests via a simple browser furthers ubiquitous learning, making the learning process more efficient. The interface supports a design-for-all paradigm by remaining consistent across devices and by being simple and yet powerful.

- **Audience:** Our framework is domain-independent and addresses all users who are interested in assessing their own knowledge of a particular domain or the knowledge of others. Consequently, it is suitable for learners and teachers of all age as long as they are able to read written natural language. Note that while the current natural-language generation layer only supports English as natural language, the verbalizer is currently being extended to support French. Learners can use ASSESS for the sake of self-assessment by generating tests that pertain to the areas in which they need to evaluate their knowledge. Teachers can use our tool for several purposes, including the following two: First, they can generate tests and make these available to students for the sake of assessing them. Moreover, they can make the learning material available as RDF and support the learners during the preparation

for exams by deploying assess on top of this learning material. For courses with pertain to general knowledge (such as geography, history, etc.), the teachers can easily reuse existing knowledge bases such as DBpedia and simply define the fragment of the knowledge base that contains the resources on interest for their course. Manifold other applications can be built on top of ASSESS. For example, we are currently exploring the use of our framework for schooling employees that need to learn the specifics of novel products that are being marketed by their company. We have evaluate the technology underlying ASSESS with 120+ users. The results of this evaluation are presented in [13]. We are currently planning the integration of ASSESS and SlideWiki [11]⁵ for the generation of questions out of the content of SlideWiki slides. Currently, SlideWiki has 3800+ presentations and more than 27.000 slides. The number of active users is around 1381.

- **Usability:** While designing the client-side of ASSESS, we wanted our end user to be confronted with a simple interface in a well-known look-and-feel and simple instructions. Moreover, we wanted the interface to be easily usable on both mobile and stationary devices. We thus chose to use mobile-optimised Java Script libraries to implement the interface of ASSESS. When using our tool, the user is first confronted with a simple configuration window that allows him/her to setup the test (see Figure 4) Here, descriptions are provided for each of the interface elements, making the configuration windows easy to use. The user is subsequently confronted with a single window per question. The verbalization of each question is displayed on the top of the page while the possible answers are displayed underneath. This approach to displaying quizzes is akin to the way questions are shown in exams. Moreover, it has been used in manifold previous applications and is thus well known and easy to use. Finally, we rely on the widely adopted Twitter Bootstrap for the interface design. These libraries have already been use to create manifold applications and thus has a look-and-feel that is familiar to most internet-affine users. The color coding is the street light coding (green for correct, red for false) and is thus easy to interpret. The summary window expressed statistics in the manner of a dashboard and can thus be easily read. The users are also given the option to export their results for future reference.
- **Performance:** Our approach is based on selecting a set of resources and using CBDs to generate questions about these resources. Hence, the time required to generate an assessment grows linearly with the number of questions. We deployed ASSESS on the large dataset DBpedia to show its scalability. Overall, we need around 5 seconds to initialize the class hierarchy and around 4 seconds/question to generate an assessment. Given that we rely on the SPARQL endpoint which contains the dataset upon which questions are to be generated and by virtue of the caching mechanisms employed in the tool, we scale as well as modern triple stores and can thus easily deploy ASSESS on billions of triples while still generating tests in acceptable times.

⁵ <http://slidewiki.org>, Statistics were collected on April 30th, 2015.

- **Data Usage and Quality:** Our application can be deployed on any set of RDF datasets. We rely directly on the ontology of the dataset to generate quizzes by using state-of-the-art verbalization techniques. Given that our tool is independent of the dataset used, there is no need for a documentation or version control of the underlying datasets. While ASSESS does not store the results of the users, it can be easily extended to do so (for example for the sake of statistical analysis by teachers). Further possible extensions would be user groups for the teachers to assess how well a group of students perform on a particular topic and thus supporting them while deciding on the content and didactics of future teaching plans.
- **Legal and Privacy:** We do not store any user data. Yet, the tool can be easily extended to do so. The terms of use state explicitly that the current version of the tool is free to use, that the tool is provided as-is and that no guarantees are provided. Still, our system does not rely on local client-side data and cannot harm our users' systems in any way. Given that we do not replicate the data contained in the endpoint (we verbalize it), we are immune against any license that prohibits the duplication of portions of the underlying knowledge base. The person who deploys ASSESS on knowledge bases with more restrictive licenses is required to restrict the accessibility of the tool to certain users (for example by using a server with a restricted access). The data generated by the ASSESS is free to use for any purpose.

4 Evaluation

In the following, we begin by presenting an evaluation of the most critical component of ASSESS, i.e., its verbalizer, as well as an usability study of the web-interface. Thereafter, we contrast our approach with related work. Finally, we present current limitations and possible extensions of our tool.

4.1 Evaluation of the Verbalizer

We have evaluated the technologies underlying ASSESS with 125 users (see Figure 5). Here, we were especially interested in knowing our well our verbalization framework performs. To this end, we used 200 queries of different complexity from the QALD-3 benchmark dataset⁶. Each user was confronted with 10 queries. We used the scale proposed by Doddington [5] to measure the fluency and adequacy of our approach. Overall, we achieved a fluency of 4.56 ± 1.29 (mean \pm standard deviation, see Figure 5), meaning that the language we generate can be understood by the users, even if it sometimes contains grammatical mistakes. Our adequacy results were 5.31 ± 1.08 (see Figure 5), which is a very positive result. This means that the expressions we use when verbalizing the classes and properties match the expected verbalization well. In 62% of the cases, we even achieve a perfect score. Further evaluation details can be found in [13].

⁶ <http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/>

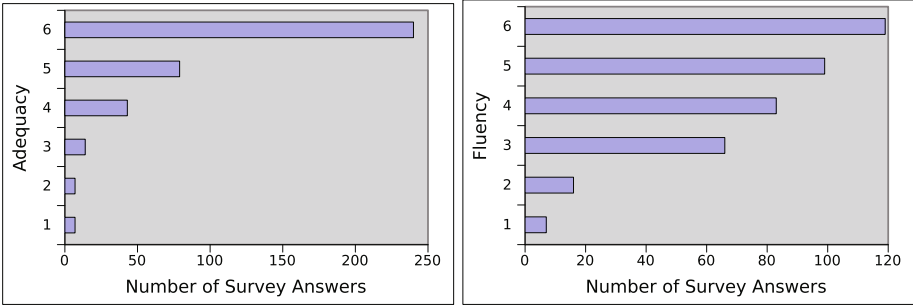


Fig. 5. Adequacy (left) and fluency (right) of ASSESS' verbalizer

4.2 Evaluation of the User Interface

We also performed a system usability study (SUS)⁷ to validate the design of our web interface. 7 users - with a good or no knowledge of natural language processing, language generation or e-learning - answered our survey resulting in a SUS-Score of 85.3. This score assign the mark *S* to the current interface of ASSESS and places it into the 10% best category of interface, meaning that users of the interface are likely to recommend it to a friend. Figure 6 shows the average voting per question and its standard deviation. All users found that they did not need to learn anything to use the tool and that the interface was intuitive. Moreover, none of the users thought that he/she would need the support of a technical person to be able to use this system (Q4) nor need to learn a lot of things before they could get going with this system (Q10). These results suggest that our interface can be deployed on a large number of learning scenarios.

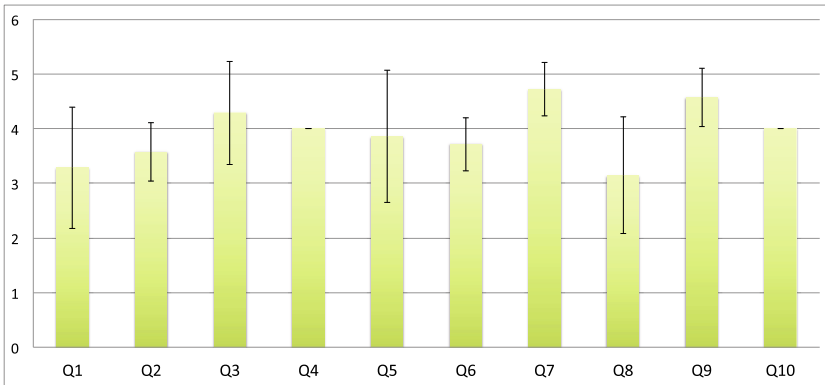


Fig. 6. Average SUS voting per question with standard deviation.

⁷ <http://www.measuringu.com/sus.php>

5 Related Work

The LinkedUp project⁸ (which ran the LinkedUp challenges) can be regarded as a crystallization point for innovative tools and applications for Open Web Data for educational purposes. Within its three consecutive challenges more than 40 submissions proofed the value of Linked Data to the educational community.

For example, Didactalia [10] is the worlds largest educational knowledge base with more than 85.000 resources created and used by more than 250.000 people. Its underlying 9 ontologies allow sophisticated information retrieval. However, the provided resources and questionnaires are static. LodStories [4] leverages Linked Open Data to build multimedia stories for art based on person, location and art data. Therewith, this project enables learning processes in the domain of art. MELOD [1] puts its emphasis on the importance of Linked Data in mobile contexts by presenting an mobile application for students visiting another city. To support this endeavor, MELOD provides ad-hoc information based on the user's location acquired from DBpedia, Europeana and Geonames.

To the best of our knowledge, the automatic generation of assessments has been addressed by a very small number of approaches. AutoTool⁹ can generate and evaluate tests that pertain to the domain of theoretical computer science. It uses a generate-and-test approach to test students answers to questions pertaining to formal grammars. Aplusix¹⁰ generates tests for the particular domain of algebra. GEPPET O[12] specializes on generating pen-and-paper tests with the aim of adapting to learners work sequences. The tool that is closest to ours in spirit is the CLAIRE framework [3], which implements a semi-automatic approach for the generation of self-assessment exercises. Yet, CLAIRE uses its own format to represent knowledge, making it more difficult to port than ASSESS. Furthermore, Foulonneau [7] has shown the value of DBpedia and the LOD Cloud to generate educational assessment items. To the best of our knowledge, existing approaches do not generate natural language questions directly out of RDF triples. This is one of the innovations of ASSESS.

6 Current Limitations and Possible Extensions

While our tool can already be used for educational purposes, it still has several limitations. First, ASSESS only generates questions in English. This limitation can be easily dealt with by extending the verbalizer with the grammar of other languages. The SimpleNLG framework¹¹ (on which we rely) has already been extended to generate French and German. These extensions will be added in future work. A further limitation of our tool is that it does not yet used OWL semantics (for example `owl:sameAs` links) to generate questions. This can yet be remedied easily during the CBD generation by (1) checking for `owl:sameAs`

⁸ <http://linkedup-challenge.org/>

⁹ <http://www.imn.htwk-leipzig.de/~waldmann/autotool/>

¹⁰ <http://www.aplusix.com/de/>

¹¹ <https://code.google.com/p/simplenlg/>

links that either go from a resource or point to a resource and (2) merging the CBD of the linked resources with that of the original resource. Our tool can be extended in several other ways. Most importantly, new question generation modules can be added. ASSESS provides a generic interface for question types. Thus, experts in need of other question types can simply implement the interface to their ends. New types of questions can include the following (note that novel user interfaces might be required for these questions):

- Relational questions: What’s the relation between two resources?
- Graphical games: Given a graph of resources and colored edges, color the missing edges correctly (each colour represents a certain relation). The game can be made more difficult by mixing missing resources and edges.
- Story-telling games: The idea here would be to verbalize a portion of the graph and replace a subset of the resources with variables. Then, the task would be to assign a resource to each of the variables in the story.

Especially in less rich countries, ASSESS presents an opportunity to distribute knowledge at low cost as it can support remote learning and online programs for students who cannot afford attending expensive universities. We aim to push towards such a use of ASSESS by combining with free lecture slides (such as SlideWiki for example) and providing means for generating ASSESS tests out of teaching material in natural language.

7 Conclusion

We presented ASSESS, a framework for the automatic generation of questions out of RDF data and SPARQL queries and query fragments. Our framework reuses and extends state-of-the-art verbalization frameworks such as SPARQL2NL. Moreover, it implements fully novel approaches for entity summarization and natural-language generation from RDF triples. We showed that ASSESS can be used on real data by evaluating (1) its kernel with 125 users on DBpedia and DBpedia queries and (2) its usability using the SUS scale. ASSESS is part of a larger agenda, in which we aim to use existing research to make educational material easily and freely available to learners all around the planet. In future work, we thus aim to combine ASSESS with teaching materials in natural language to provide both lecture slides and self-evaluation questions at low cost. Several problems remain to be solved to achieve this goal. Especially, we aim to deploy a generic approach to extract RDF triples from the pseudo-natural language used in slides leverages existing technologies like FOX [14] and BOA [8].

References

1. Arrigo, M., Taibi, D., Fulantelli, G.: A mobile environment for learning with linked open data (2013)
2. Bornman, E.: The mobile phone in africa: Has it become a highway to the information society or not. *Contemp. Edu. Tech.* **3**(4) (2012)

3. Cablé, B., Guin, N., Lefevre, M.: An authoring tool for semi-automatic generation of self-assessment exercises. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS, vol. 7926, pp. 679–682. Springer, Heidelberg (2013)
4. Chen, J., Liu, Y., Maulik, D., Xu, L., Zhang, H., Knoblock, C.A., Szekely, P., Vander Sande, M.: Lodstories: Learning about art by building multimedia stories1 (2014)
5. Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: Proceedings of the Second International Conference on Human Language Technology Research, pp. 138–145. Morgan Kaufmann Publishers Inc. (2002)
6. Ell, B., Vrandečić, D., Simperl, E.: SPARTIQUILATION: verbalizing SPARQL Queries. In: Simperl, E., Norton, B., Mladenic, D., Valle, E.D., Fundulaki, I., Passant, A., Troncy, A. (eds.) ESWC 2012. LNCS, vol. 7540, pp. 117–131. Springer, Heidelberg (2015)
7. Foulonneau, M.: Generating educational assessment items from linked open data: the case of dbpedia. In: García-Castro, R., Fensel, D., Antoniou, G. (eds.) ESWC 2011. LNCS, vol. 7117, pp. 16–27. Springer, Heidelberg (2012)
8. Gerber, D., Ngonga Ngomo, A.-C.: Bootstrapping the linked data web. In: 1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011 (2011)
9. Gerber, D., Hellmann, S., Bühmann, L., Soru, T., Usbeck, R., Ngonga Ngomo, A.-C.: Real-time RDF extraction from unstructured data streams. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 135–150. Springer, Heidelberg (2013)
10. Gómez, A.A.: Analyzing and producing educational resources for didactalia. net: a pilot project launched at the university of deusto (spain) with students from primary education degree. In: INTED2014 Proceedings, pp. 7186–7191 (2014)
11. Khalili, A., Auer, S., Tarasowa, D., Ermilov, I.: SlideWiki: elicitation and sharing of corporate knowledge using presentations. 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. 302–316. Springer, Heidelberg (2012)
12. Lefevre, M., Jean-Daubias, S., Guin, N.: Generation of pencil and paper exercises to personalize learners work sequences: typology of exercises and meta-architecture for generators. In: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, vol. 2009, pp. 2843–2848 (2009)
13. Ngonga Ngomo, A.-C., Bühmann, L., Unger, C., Lehmann, J., Gerber, D.: Sparql2nl - verbalizing sparql queries. In: Proc. of WWW 2013 Demos, pp. 329–332 (2013)
14. Speck, R., Ngonga Ngomo, A.-C.: Ensemble learning for named entity recognition. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 519–534. Springer, Heidelberg (2014)
15. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data. A survey. Semantic Web Journal (2015)

Ontology-Based Data Access

Ontology Based Access to Exploration Data at Statoil

Evgeny Kharlamov¹(✉), Dag Hovland², Ernesto Jiménez-Ruiz¹, Davide Lanti³, Hallstein Lie⁴, Christoph Pinkel⁵, Martin Rezk³, Martin G. Skjæveland², Evgenij Thorstensen², Guohui Xiao³, Dmitriy Zheleznyakov¹, and Ian Horrocks¹

¹ University of Oxford, Oxford, UK
evgeny.kharlamov@cs.ox.ac.uk

² University of Oslo, Oslo, Norway

³ Free University of Bozen-Bolzano, Bolzano, Italy

⁴ Statoil ASA, Stavanger, Norway

⁵ fluid Operations AG, Walldorf, Germany

Abstract. Ontology Based Data Access (OBDA) is a prominent approach to query databases which uses an ontology to expose data in a conceptually clear manner by abstracting away from the technical schema-level details of the underlying data. The ontology is ‘connected’ to the data via mappings that allow to automatically translate queries posed over the ontology into data-level queries that can be executed by the underlying database management system. Despite a lot of attention from the research community, there are still few instances of real world industrial use of OBDA systems. In this work we present data access challenges in the data-intensive petroleum company Statoil and our experience in addressing these challenges with OBDA technology. In particular, we have developed a deployment module to create ontologies and mappings from relational databases in a semi-automatic fashion, and a query processing module to perform and optimise the process of translating ontological queries into data queries and their execution. Our modules have been successfully deployed and evaluated for an OBDA solution in Statoil.

1 Introduction

The competitiveness of modern enterprises heavily depends on their ability to make use of business critical data in an efficient and timely manner. Providing this ability in data intensive enterprises is not a trivial task as the growing size and complexity of information sources makes data access and exploitation increasingly challenging. Indeed, data is often scattered across heterogeneous and autonomously evolving systems or has been adapted over the years to the needs of the applications they serve, making it difficult to extract data in a useful format for the business of the organisation.

Statoil is a large and data intensive enterprise where the workflow of many units heavily depends on timely access to data. For example, the job of exploration geologists is to analyse existing relevant data in order to find exploitable

accumulations of oil or gas in given areas. This process is typically done in two separate steps: first by gathering data from multiple sources, then by analysing the data using specialised analytical tools. As is often the case in large enterprises, naming conventions for schema elements, constraints, and the structure of database schemata are very complex and documentation may be limited or nonexistent. As a result, the data gathering task is often the most time-consuming part of the decision making process.

Ontology Based Data Access (OBDA) [21] is a prominent approach to data access in which an *ontology* is used to mediate between data users and data sources. The ontology provides ‘a single point of semantic data access’, and allows queries to be formulated in terms of a user-oriented conceptual model that abstracts away complex implementation-level details typically encountered in database schemata. Domain experts are thus able to express information needs in their own terms without any prior knowledge about the way the data is organised at the source, and to receive answers in the same intelligible form. The ontology is connected to the data via a set of *mappings*: declarative specifications that relate ontological terms with queries over the underlying data. OBDA systems automatically translate ontological queries, i.e., SPARQL, into database queries, i.e., SQL, and delegate execution of SQL queries to the database systems hosting the data. OBDA is a natural fit to address the Statoil data access challenges described above: if complex databases are presented to users via an ontology, then they can formulate queries in terms of classes and properties in an object-centric fashion, e.g., asking for all *wellbores penetrating a rock layer* of a specific *geological age*. Moreover, OBDA is a so-called *virtual* approach, providing an access layer on top of databases while leaving the data in its original stores. Thus, OBDA has the potential to improve data access with a minimal change to existing data management infrastructure.

OBDA has recently attracted a lot of attention and a number of systems have been developed, e.g., [1, 22]. However, to the best of our knowledge, the following two problems have attracted only limited attention:

- (i) How to *create* ontologies and mappings for a deployment of an OBDA system?
- (ii) How to ensure that OBDA query processing is *efficient* in practice?

At the same time, these problems have high practical importance for OBDA systems in general and in particular for deploying an OBDA system in Statoil. First, deployment of an OBDA system comes with a high modelling cost due to the complexity of the domain and of the database schemata. Second, unoptimised OBDA query processing may become impractical when the ontology and/or database are large [23].

In this paper we present our experience in the development of OBDA deployment and query optimisation modules, and their use in providing an OBDA solution for Statoil. In particular, our solution (i) applies a novel set of semi-automatic techniques to bootstrap new ontologies and mappings from relational databases and to integrate existing ones, and (ii) uses novel optimisation techniques to improve query processing for producing compact and efficient SQL queries.

Our goal is to improve the efficiency of the data gathering routine for Statoil geologists by focusing on their access to two prominent data sources: their *Exploration and Production Data Store (EPDS)*, and the *NPD FactPages*. EPDS is Statoil’s corporate data store for exploration and production data and their own interpretations of this data, and is thus an important and heavily used database. EPDS was created about 15 years ago and it currently has about 3,000 tables with about 37,000 columns, and contains about 700 GB of data. The NPD FactPages (NPD FP) is a publicly available dataset that is published and maintained by the Norwegian authorities. It contains reference data for many aspects of the Norwegian petroleum industry, and is often used as a data source by geologists in combination with EPDS. The NPD FP is converted into a relational database with about 70 tables, 250 columns and 50 MB of data [26]. As an example of the difficulty of the data gathering task at Statoil: due to the complexity of EPDS, writing queries over it is a significant effort that requires proficiency with all the variety of its underlying schema components, and it is common for SQL queries that gather data needed by geologists to contain thousands of terms and have 50–200 joins.

Our OBDA solution has been successfully deployed at Statoil over EPDS and NPD FP.¹ In particular, as a measure of success, we used a catalogue of queries collected from Statoil geologists which cover a wide range of typical information needs, and that are hard to formulate over EPDS and NPD FP; our deployment covers this catalogue. Finally, we demonstrate empirically that our optimisation techniques guarantee good execution times for the catalogue queries.

The paper is organised as follows: in Section 2 we present Statoil’s data access challenges, in Section 3 we give an overview of OBDA, in Section 4 we give technical background of our deployment and query processing solutions, in Sections 5 and 6 we share our experience on OBDA deployment and query execution at Statoil, and in Sections 7 and 8 we summarise the lessons we learned, propose future work, and conclude.

2 Data Access Challenges at Statoil

Following common practices of large enterprises, geologists at Statoil analyse data in two steps: they first gather relevant data from EPDS and other data sources, and then apply analytical reporting tools on top of the gathered data [9]. The first step is typically done via a variety of query interfaces and data extraction tools, such as geographical information system (GIS) tools and specialised data manipulation tools, that we shall collectively refer to as *access points*. The flexibility of the access points is limited and in general users can control them only by inserting values for certain query parameters. When information needs cannot be satisfied with any of the available access points, geologists, possibly with the help of IT staff, try to combine answers obtained from several access points. In some cases, geologists have to contact IT staff to provide a new access point. Developing new access points is a very time consuming process; in Statoil

¹ We also have a preliminary deployment of the solution at Siemens [16].

it commonly takes from several days to weeks to produce an access point that completely answers the required information need. It has been estimated that in the oil and gas industry 30–70% of the time geologists spend on analytical tasks used for gathering data [8]; this estimate is confirmed by Statoil.

To get a better understanding of why data gathering takes so long, consider a common procedure of creating an access point. Each such point is typically based on a materialised special purpose database view. The process of making such view over EPDS consists of the three ETL steps: (i) extracting, (ii) transforming, and (iii) loading data. For Step (i) Statoil IT staff locate relevant data in EPDS, other data sources or existing access points, and produce SQL code for its extraction. This is done using specialised data extraction tools, since directly accessing complex database schemata like EPDS' is prone to error, time consuming, and not always feasible due to its complexity and limited documentation. During Step (ii), using specialised tools, IT staff define how to preprocess the data extracted at Step (i). This adds another layer of data processing, this time over the relevant data, to perform projections, filtering, joins, schema renamings and complex computations such as geographical coordinate conversions. In Step (iii), IT staff populate the access point's view with the data: both data extraction and manipulation code is executed which materialises the view. In sum, building an ETL process that establishes an access point for a complex information need consists of a myriad of data access and processing steps, many of which require deep knowledge of the data that is being processed and how it is represented.

There are around 900 geologists and geophysicists at Statoil and accessing data is their daily routine. Since they often need new access points, hundreds of highly skilled employees often have to wait several days for accessing data before they can conclude analytical tasks. Reducing this time from days to hours would bring a significant saving and, more importantly, would improve the effectiveness of Statoil's exploration department, which is key to their overall competitiveness and profitability.

3 Ontology Based Data Access

An OBDA instance $\mathcal{S} = (\mathcal{D}, \mathcal{V}, \mathcal{O}, \mathcal{M})$ is a quadruple where \mathcal{D} is an RDB, \mathcal{V} is an ontological vocabulary, i.e., a set of classes and properties, \mathcal{O} is an ontology over \mathcal{V} , i.e., a set of axioms expressed in a fragment of first-order logic, and \mathcal{M} is a set of mappings between \mathcal{D} and \mathcal{V} , i.e., assertions of the form: $P(f(x), f(y)) \leftarrow \text{SQL}(x, y)$ or $C(f(x)) \leftarrow \text{SQL}(x)$ where C and P are class and property names from \mathcal{V} , $\text{SQL}(x)$ and $\text{SQL}(x, y)$ are SQL queries over \mathcal{D} with one and two output variables, and f is a function 'casting' values returned by SQL into URIs and values (e.g., strings, dates).

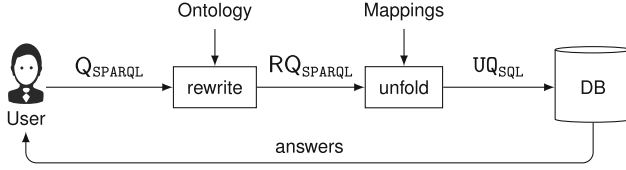


Fig. 1. Query processing in OBDA

In order to answer an ontological query Q expressed in terms of \mathcal{V} over $(\mathcal{D}, \mathcal{V}, \mathcal{O}, \mathcal{M})$, one can execute over \mathcal{D} the SQL queries occurring in \mathcal{M} , then use the computed answers to ‘populate’ the extensions of the corresponding classes and properties occurring in \mathcal{M} , thus creating a set of ontological facts \mathcal{A} ,² and finally evaluate Q over $\mathcal{A} \cup \mathcal{O}$. Since $\mathcal{A} \cup \mathcal{O}$ is a logical theory, query answering over it corresponds to logical reasoning and is defined in terms of *certain answers*. Intuitively, a tuple t is a certain answer to Q over $\mathcal{A} \cup \mathcal{O}$ if $Q(t)$ holds in every first-order model of $\mathcal{O} \cup \mathcal{A}$ [6]. Computation of \mathcal{A} and certain answers, however, can be very expensive, with worst-case complexity depending on both ontology and query language expressiveness. In particular, it was shown [6] that the computation is tractable in data complexity (i.e., in the size of \mathcal{D}) if ontological queries Q are conjunctive (CQs) and ontologies \mathcal{O} are expressed in OWL 2 QL. Computation of certain answers in this setting can be accomplished using the two stage approach of (i) *rewriting* and (ii) *unfolding* as depicted in Figure 1. Rewriting of ontological queries essentially corresponds to compilation of relevant ontological information into Q ; it is similar to the resolution procedure in Prolog, and can be accomplished with the *perfect reformulation algorithm* [6], that takes as input a conjunctive query Q and an OWL 2 QL ontology \mathcal{O} and returns a union of conjunctive queries RQ . Computation of certain answers for RQ over \mathcal{A} returns the same answers as for Q over $\mathcal{A} \cup \mathcal{O}$. Unfolding inputs RQ and \mathcal{M} and translates RQ into an SQL query UQ by essentially substituting occurrences of classes and properties in RQ with the SQL queries that they correspond to in \mathcal{M} . Evaluation of UQ over D effectively returns the certain answer computed by RQ over \mathcal{A} and thus by Q over $\mathcal{A} \cup \mathcal{O}$.

4 Technical Background of Our OBDA System

Our OBDA deployment in Statoil relies on two self-developed systems: BOOTOX [13] for creating ontologies and mappings, and *Ontop* [24] for performing query optimisation and SPARQL query processing. During the deployment and use of BOOTOX and *Ontop* in Statoil we experienced many practical challenges, which required substantial improvements to both systems.

² This process is called *materialisation* of the ontological facts from data via mappings.

4.1 Deployment

We support three deployment scenarios:

- (i) *Bootstrapping* is used for semi-automatic extraction of an ontology and mappings from an RDB. It takes as an input a dataset \mathcal{D} and returns a vocabulary \mathcal{V} , a set of mappings \mathcal{M} relating \mathcal{D} to \mathcal{V} , and an ontology \mathcal{O} over \mathcal{V} .
- (ii) *Merging* is applied to incorporate pre-existing ontologies in existing OBDA instances, e.g., by aligning it with the bootstrapped ontology. It takes as input an OBDA instance $(\mathcal{O}_1, \mathcal{M}, \mathcal{D})$ and an ontology \mathcal{O}_2 , and returns $(\mathcal{O}, \mathcal{M}, \mathcal{D})$ where \mathcal{O} is a ‘merger’ of \mathcal{O}_1 and \mathcal{O}_2 .
- (iii) *Layering* is used to ‘connect’ pre-existing ontologies directly to an RDB with semi-automatically generated mappings. It takes an ontology \mathcal{O} and a dataset \mathcal{D} as input and returns a set of mappings \mathcal{M} such that $(\mathcal{O}, \mathcal{M}, \mathcal{D})$ is an OBDA instance.

The mappings and ontologies we obtain in all the three scenarios require further inspection by ontology engineers and domain experts to detect the most promising classes, properties, axioms, and mappings, and to verify their quality. To the best of our knowledge existing bootstrapping systems provide limited or no support for the three deployment scenarios above, see, e.g., [25, 32] for an overview of such systems.

Bootstrapping. The goal of bootstrapping is to find patterns in \mathcal{D} , i.e., SQL queries $\text{SQL}(x)$ and $\text{SQL}(x, y)$ that correspond to meaningful classes and properties. We bootstrap three types of mappings depending on what relational algebra operators can appear in their SQL-parts: (i) *projection*, (ii) *selection*, and (iii) *full mappings* that allow any relational algebra operators.

A special kind of projection mappings, that are recommended by W3C as the standard way to export relational data in RDF, are *direct mappings*. They mirror RDB schemata by essentially creating one class for each table and one property for each column. Ontological vocabulary extracted by direct mappings can be enriched with axioms by propagating RDB constraints, e.g., a foreign key relating two tables could be propagated into a subclass assertion between the two corresponding classes. BOOTOX supports extraction of direct mappings and propagation of constraints; moreover, it outperforms existing bootstrapping systems that were available for benchmarking [19]. BOOTOX can also discover implicit constraints in databases, e.g., minimal primary keys in tables, candidate foreign keys by checking containment between (samples from) projections of tables [14]. While working with EPDS we found that the discovery of implicit constraints was practically important since prominent tables are materialised views without specified primary or foreign keys, and axioms constructed from such constraints are exploited in query optimisation. Note that the bootstrapped ontology can be quite close to the source schema and we see this as a natural outcome: bootstrapping is the first step in OBDA system deployment, and the resulting assets are by no means perfect, but provide a starting point for post-processing and extension.

Selection mappings are bootstrapped in order to create class and property hierarchies, e.g., we take a class C bootstrapped with direct mappings and verified by users, and in the corresponding tables we learn attributes whose values give good clustering of tuples; then, for each cluster we create a subclass of C . To name detected subclasses, we categorise attribute values that determine clusters by matching them to DBPedia. We also apply these techniques to other types of mappings.

In order to bootstrap full mappings, we analyse schema dependencies between tables and use statistical methods on data values. In particular, we learn chains of tables that are ‘well joinable’, that is, connected via foreign keys or with good overlap on some sets of attributes, and convert them into candidate classes and properties. For instance, for each chain we detect the ‘leading’ table T and relate the chain to a class by projecting it on T ’s primary key; then, we combine names of the tables in the chain and attributes on which they were joined to suggest a name for this class.

Alignment. A way to extend a bootstrapped ontology is to align it with an existing high quality domain ontology. For this purpose we extended an existing ontology alignment system LogMap [12] that aligns two ontologies \mathcal{O}_1 and \mathcal{O}_2 by deriving equivalence and sub-class(property) assertions between the terms from \mathcal{O}_1 ’s and \mathcal{O}_2 ’s vocabularies using the lexical characteristics of the terms and the structure of the ontologies. Our extension [27] is a highly scalable solution that ensures that after \mathcal{O}_1 and \mathcal{O}_2 are aligned the resulting ontology \mathcal{O} is a *conservative extension* of \mathcal{O}_1 and \mathcal{O}_2 w.r.t. atomic subsumptions, i.e., \mathcal{O} does not entail any sub-class(property) assertion over \mathcal{O}_1 ’s and \mathcal{O}_2 ’s vocabulary which is not already entailed by \mathcal{O}_1 or \mathcal{O}_2 . When experimenting with EPDS, we noticed that these logical guarantees are often a necessity since an alignment \mathcal{O} of a bootstrapped \mathcal{O}_1 with an imported domain ontology \mathcal{O}_2 that does not preserve consistency or conservativity gives the following side-effects: query answering over \mathcal{O} produces answers that are unexpected by domain experts, and that would not be obtained if one queries \mathcal{O}_1 alone, or \mathcal{O} entails axioms over \mathcal{O}_2 ’s terms that are unexpected and counter-intuitive for domain experts.

Layering. Our layering procedure uses a novel technique for the semi-automatic discovery of direct mappings between \mathcal{O} and \mathcal{D} by performing string matching of \mathcal{O} and \mathcal{D} ’s schemata enhanced with their structural comparison [20].

4.2 Query Processing Optimisation

Our query processing system relies on the two stage process with rewriting and unfolding as described above. It was observed [23] that a naive implementation of this approach performs poorly in practice; thus, we developed and implemented a number of techniques to optimise both stages of query processing, which we present in detail below. We empirically tested the efficiency of our optimisation techniques over EPDS and will present results on query execution in Section 7. We also evaluated our techniques in a controlled environment and our tests

show that thanks to these optimisation techniques our query processing solution can dramatically outperform existing OBDA solutions [18]. Furthermore, we observed that even after we optimise rewriting and unfolding, the SQL queries UQ we produce often return answers with duplicates; below we will discuss why this is an issue and how we addressed it.

Optimisation of Rewriting. We address two challenges:

- (i) *redundancy* in RQ: fragments of RQ may be subsumed by each other and thus evaluation of UQ over RDBs will require redundant computation;
- (ii) *inefficiency* of rewriting: computation of RQ is in the worst case exponential in the size of Q and \mathcal{O} , and thus its online computation is often slow for large Q and \mathcal{O} .

The main source of redundancy in RQ is that classes (properties) can participate in multiple mappings either directly, or indirectly via their multiple sub-classes (sub-properties).³ To avoid this, we minimise both the mappings and the UCQ RQ using query containment. To address the inefficiency, we proposed two novel techniques. Both techniques can be applied in isolation or combined. Our first technique is to improve computation of class hierarchies entailed by the ontology, which the rewriting heavily relies on, by applying graph reachability techniques to a DAG-encoding of dependencies among classes. The second one is to move part of online reasoning offline: for all atomic queries we perform expensive rewriting up front and compile the results of this computation into the existing mappings, and use these enriched mappings when user queries Q are unfolded, see [17] for details.

Optimisation of Unfolding. We address three challenges with query unfolding:

- (i) *redundant unions* due to redundancies in the bootstrapped ontology or mappings;
- (ii) *redundant joins*, that come from the fact that on the ontological level the data is modelled as a graph, i.e., as a ternary relation, while on the data level in RDBs it is modelled with n-ary relations, and thus an unfolding of RQ into SQL over an n-ary table naturally introduces n-1 self-JOIN operations;
- (iii) *inefficient joins* come from the so-called *impedance mismatch*, i.e., on the ontological level objects are represented with object ids URIs while in RDBs with tuples; thus, joins in RQ are unfolded into SQL joins over string concatenation that prevents RDBs from the use of existing indices.

To address these issues, we developed *structural* and *semantic* optimisation techniques. For structural optimisations we push joins inside the unions and special functions (such as URI construction) as high as possible in the query tree; we also detect and remove inefficient joins between sub-queries. Semantic optimisations remove redundant unions, joins, detect unsatisfiable or trivially satisfiable conditions, etc.

³ This issue is partially tackled at the bootstrapping stage [13].

Optimisation of distinct answers. Removing duplicates from query answers raises an interesting problem for OBDA systems. On the one hand, OBDA theory assumes set semantics for computation of query answers, that is, each answer occurs only once in the answer set. On the other hand answering queries over relational databases RDF databases is typically implemented under bag semantics, that is, any answer can occur multiple times in the answer set. In particular, evaluation of SQL queries produced by our query processing system returns answers with duplicates. From our experience with the OBDA deployment at Statoil, these duplicates bring a number of challenges: duplicate answers appear as noise to most end-users, visualisation systems are significantly slowed down when flooded with repeated answers, and the large number of such answers negatively affects network capacity and database connectivity. Using the `distinct` modifier in SPARQL queries to remove redundant answers is unfolded into the SQL `distinct`, which, in our experiments, was extremely detrimental to performance and led to a number of queries to time out. In order to overcome this problem, we propose to defer the removal of duplicate tuples to the OBDA system rather than the underlying database engine. In particular, we filter out redundant answers using predefined Java hash functions. This simple solution outperforms SQL `distinct` by orders of magnitude in most cases, and opens the door for further optimisation (see Section 6).

5 OBDA Deployment at Statoil

In this section we present our experience in deploying an OBDA instance over EPDS. We start with the requirements, then discuss how we developed the ontology and mappings for EPDS, and conclude with a quality assessment of the deployment.

5.1 Requirements

Our OBDA solution should enable efficient formulation of information needs from Statoil geologists. We gathered these needs via interviews with geologists and the IT experts who support them, which gave us a catalogue of 60 representative information requests in English. The following are examples of gathered information needs:

1. In my area of interest, e.g., the Gullfaks field, return wellbores penetrating a specific chronostratigraphic unit, and information about the lithostratigraphy and the hydrocarbon content in the wellbore interval penetrating this unit.
2. Show all the core samples overlapping with the Brent Group.
3. Show all permeability measurements in Brent.

The requests in the catalogue can be seen as ‘patterns’ of information needs, and each such request represents one topic that geologists are typically interested in.

Table 1. Ontology metrics.

| Ontology | Axioms | Classes | Object prop. | Datatype prop. |
|------------------------|--------|---------|--------------|----------------|
| EPDS, bootstrapped | 73,809 | 3,069 | 3,266 | 34,336 |
| EPDS, relevant excerpt | 16,592 | 627 | 864 | 7,634 |
| EPDS, manually built | 469 | 97 | 39 | 34 |
| NPD FactPages | 8,208 | 344 | 148 | 237 |
| – ISC | 6,644 | 18 | 4 | 4 |
| – GeoSPARQL | 191 | 78 | 43 | 9 |
| – BFO | 278 | 39 | 0 | 0 |

We verified with domain experts that the catalogue provides a good coverage for the information needs of Statoil geologists, and from this we derived the first natural minimum requirement for the ontology and mappings:

Requirement 1: *The ontology should enable formulation of queries corresponding to the catalogue’s requests and mappings should enable answering these queries.*

To fulfil Requirement 1, the ontology must contain all the terms occurring in the catalogue. For example, Information need 1 contains the terms *wellbores*, *penetrating*, *chronostratigraphic unit*, *lithostratigraphy*, *hydrocarbon content*, and *wellbore interval*. All in all the catalogue contains 140 relevant domain terms. As we verified with Statoil geologists, the terms occurring in the catalogue are important, but, as expected, do not provide a sufficient domain coverage; that is, geologists need many more domain specific terms for expressing their information needs. Via interviews with geologists, we determined six domains that should be reflected in the ontology: geospatial, geometrical, enterprise, production, seismic and oil related facilities, which gave the following requirement:

Requirement 2: *The ontology should cover a wide range of geological domain terms including the ones from the catalogue and the six relevant domains.*

5.2 Development of Ontologies and Mappings

The ontology developed for Statoil consists of a part which is bootstrapped from EPDS and a second part which is the NPD FactPages ontology developed in an earlier phase of the project [26]. Running the bootstrapper over EPDS extracted an ontology comprising 3,069 classes, 37,602 properties, and 73,809 axioms from explicit and 155,152 axioms from implicit constraints. Many of the bootstrapped classes and properties have names that are related to the specific structure of EPDS and thus are meaningless to geologists, e.g., `RCA_SO_DS` or `EXP_UNIQUENESS_RULE_ATT_00022_X`, while others have a clear geological semantics, e.g., `RESERVOIR` and `WELL_ELEVATION`. In order to mitigate the meaningless terms, we did a semi-automatic quality assessment and manual improvement of the bootstrapped ontology, and extracted a fragment with 627 classes and 8,498 properties that are relevant to the domain, and 16,592 axioms that

provide meaningful semantic relationships between these classes and properties. See Table 1 for the ontology metrics.

The NPD FP ontology helps us meet Requirement 2 by defining vocabulary about exploration and development facilities, oil company organisation, and seismic activities, and furthermore it contains geology and geometry terminology by importing *International Chronostratigraphic Chart* developed by the *International Stratigraphy Community* (ISC) and *GeoSPARQL*, which is used for representing basic geometric concepts and, in particular, information about topology. We partially aligned the ontology to the bootstrapped ontology and partially layered it to EPDS.

Most of the axioms in the ontologies, including all bootstrapped axioms, fall in the OWL 2 QL profile, which is required for OBDA to guarantee correctness query processing. Examples of OWL 2 QL axioms are $\text{NaturalGasLiquid} \sqsubseteq \text{Petroleum}$ (natural gas liquids are a sort of petroleum), $\text{Wellbore} \sqsubseteq \exists \text{hasLicense}$ (each wellbore has a license associated to it), and $\text{Company} \sqsubseteq \exists \text{hasName}$ (each company has a name). The ontology also contains 16 non-OWL 2 QL axioms, 13 of which we approximated in OWL 2 QL using the techniques of [7], and the other three involved universal restrictions, e.g., $\text{WlbCoreSet} \sqsubseteq \forall \text{member.WlbCore}$, which were dropped.

In order to complement the bootstrapped mappings, it was necessary to manually create complex mappings to cover 24 classes and 34 properties. These are classes and properties from the catalogue that were either not discovered by the bootstrapper or the bootstrapped mappings did not sufficiently reflect their relation to EPDS. On average the size of the SQL query in each mapping is 9 words (including keywords), and they are all conjunctive queries. The queries involve up to 6 tables, with an average of 3. The number of output variables SQL queries of mappings ranges from 2 to 4, with the average of 3. In order to develop the mappings we analysed predefined queries used by different information extraction tools over EPDS. In particular, we analysed the results of the predefined queries over EPDS and compared them with the expected results, by interviewing Statoil domain experts. This allowed us to fragment the predefined queries and extract small excerpts useful for OBDA mappings. The relatively small size of the SQL parts of mappings was dictated by two practical reasons: to make maintenance and documenting of the mappings easier, and to ensure efficiency of query processing.

5.3 Assessing the Quality of Our OBDA Deployment

We assess the quality of our deployment by first checking how good the ontology covers the query catalogue, and then how good our mappings cover the terms in the catalogue.

To evaluate the coverage of the query catalogue by the ontology, we aligned them by first (i) a syntactic match of their terms and then (ii) a structural comparison of neighbourhoods around terms that have a syntactic match. The alignment outputs a set of pairs of matched terms together with a score showing

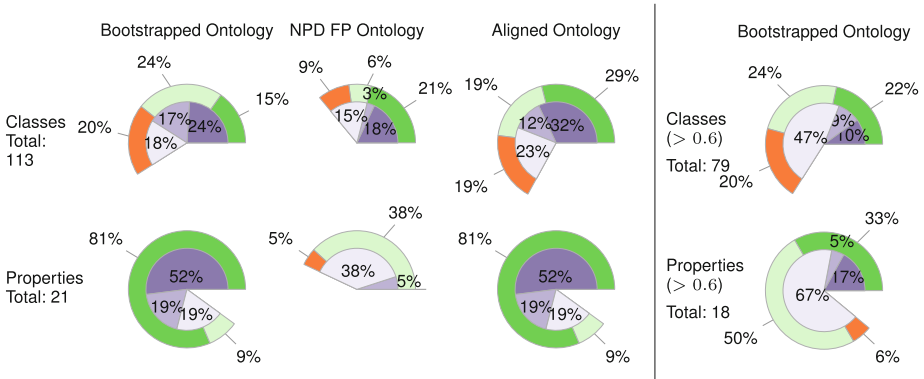


Fig. 2. Inner pie charts show coverage by lexical confidence using I-SUB [33]: ■ in $[0.9, 1.0]$, ■ in $[0.8, 0.9]$, □ in $[0.6, 0.8]$. Outer pie charts represent the quality of the terms with a coverage above 0.6: ■ *true positive*, ■ *semi-true positive*, ■ *false positive*. Note that semi-true positives are not clear-cut cases where the ontology term has a broader or narrower meaning with respect to the query term. Left part displays the coverage of terms from the query catalogue with terms from ontologies; Right part shows the overlap between terms from bootstrapped and imported ontologies that (with confidence > 0.6) occur in the query catalogue

how well they match. For this purpose we extended the ontology alignment system LogMap [27], that can perform both syntactic and structural matching of ontologies, so that it also can perform the required alignment of ontologies and query catalogues. The main challenge was to define the notion of a structural neighbourhood of a query term in a set of conjunctive queries. We introduced the following notion: given a set of queries S and a term t , its neighbourhood in S is the set of all terms t' occurring in some $Q \in S$ that contains t , together with the surrounding sequence of terms that is common in all such Q . We did the coverage assessment separately for the ontology bootstrapped from EPDS, the NPD FP ontology, and the alignment of these two. The results of the matching are in Figure 2, in the inner circles. The six pie charts on the left describe the coverage of the query terms by ontologies: the upper three show the coverage of classes by, respectively (left-to-right) bootstrapped, NPD FP, and aligned ontologies, the lower three show the coverage of properties. Finally, together with three domain experts we performed a manual assessment of each match for both classes and properties. The results of our assessments are also in Figure 2 in the outer circles. For example, manual assessment of coverage of classes by the bootstrapped ontology gave 15% of true positives (they are correct for domain experts), then, 24% of semi-true positives, and 20% are false-positives (the matches are wrong for domain experts). In the case of properties, most bootstrapped ones that were matched to the query catalogue terms, i.e., 81%, are true positive.

To evaluate the coverage of the query catalogue by the mappings, we used a different approach. If a query term is present in the ontology it can be used

in query composition. At the same time there might be no mapping relating this term to EPDS, and all queries that use this term will always return the empty answer set. Thus, we checked how well the query terms reflected in the aligned ontology are covered with mappings. Clearly, by the construction, all the query terms that are reflected in the bootstrapped ontology are related to EPDS via bootstrapped direct mappings. While, if a query term is reflected in the NPD FP ontology, then it does not have a mapping to EPDS unless it got aligned to a term in the bootstrapped ontology. Indeed, alignment sets a subset or equivalence relationship between terms, and thus a mapping for one aligned term may be reused by another term. We verified how many query terms reflected in the NPD FP ontology are also present in the bootstrapped ontology, and in Figure 2 the right two pie charts depict the outcome. In this experiment we verified the confidence of the alignment and did manual assessment with domain experts. From these experiments we conclude that for about 50% of query classes (and about 80% of properties) from the NPD FP ontology (that are aligned to the bootstrapped classes) we can reuse direct mappings of the bootstrapped ontology.

6 Query Answering over OBDA Deployment in Statoil

In this section we present query evaluation experiments with our OBDA solution over EPDS and NPD FP. We start with the requirements.

6.1 Requirements

The goal of our OBDA deployment is to make gathering of data from EPDS more efficient; this leads us to the next requirement:

Requirement 3: Queries from the Statoil catalogue expressed over the ontology should be much simpler than the data queries in corresponding access points. Execution time of these queries over our OBDA deployment should be similar to the data extraction time of the corresponding access points.

We start with analysing the structure of queries from the Statoil catalogue. Most of them, 73%, are either linear or three-shaped conjunctive queries, the others contain aggregate functions and negation. No query in the catalogue has a cycle. Expressing them over the ontology requires from 3 to 13 ontological terms, and the longest query contains 23 terms. In Figure 3, we illustrate the query from the catalogue that correspond to Information needs 3 from Section 5 expressed over the ontology. These queries are quite simple and contain 8 and 13 terms only. At the same time, the Statoil access points corresponding to these two queries are based on two complex SQL queries Q_{IN2} and Q_{IN3} , where Q_{IN2} involves 7 tables and 24 statements in the WHERE clause with 7 technical statements of the form ‘T.A is not null’ that ensure correctness of query execution, and 10 joins; Q_{IN3} involves 14 tables and 38 statements in the WHERE clause with 9 technical statements and 18 joins. Due to the space limit, we do not show here Q_{IN2} and

$Q_{IN3}(x_5, x_6, y_1) : -$ Core(x_1), extractedFrom(x_1, x_2), WellboreInterval(x_2),
 hasCoreSample(x_1, x_5), CoreSample(x_5),
 hasPermeabilityMeasurement(x_5, x_6),
 PermeabilityMeasurementResult(x_6), valueInStandardUnit(x_6, y_1),
 overlapsWellboreInterval(x_2, x_3), WellboreInterval(x_3),
 hasUnit(x_3, x_4), StratigraphicUnit(x_4), name(x_4 , "BRENT").

Fig. 3. Information need 3 from Section 5 expressed over the ontology vocabulary

Q_{IN3} . This provides clear evidence that the catalogue queries over the ontology are much simpler than the corresponding access point queries. Moreover, expressing catalogue queries over the ontology is relatively easy and could be done quite quickly by IT experts—we did it in one day. We also believe that even geologists could formulate these queries relatively quickly with appropriate tool support.

6.2 Running Queries over EPDS

We conducted three sets of experiments with the OBDA deployment using the Statoil query catalogue, and ontology and mappings relevant for the queries in the catalogue. The queries were run over the production server⁴ of EPDS; the results are presented in the top two plots of Figure 4. In the first set of experiments, that we refer to as *noDist*, we executed queries as they appear in the catalogue, while in the other two sets we executed these queries with *distinct*, where in *dbDist* experiments *distinct* is processed by the database engine, while in *obdaDist* by the OBDA engine, as discussed in Section 4.2. Each query in the experiments was executed four times with a 20-minute timeout.

In the *noDist* experiment, 17 out of 60 queries timed out. Out of the successful 43, for 2 queries the time is less than 1s, for 3 queries it is between 2m and 6m, while for most queries it is between 3s and 2m; the average time is 36.5s and the median is 12.5s. These numbers are impressive, as the SQL queries produced by the OBDA system and sent to EPDS are large: they have on average 51k characters and the largest query is 275k characters. Note that execution time here does not include the time for rewriting and unfolding of SPARQL queries, but only the actual execution of the resulting SQL queries; the maximum unfolding time is 187ms which does not add a significant time overhead to the reported query execution time. In order to understand how good the times reported in Figure 4 are for Statoil geologists, we gathered statistics on execution time for queries behind the access points available in Statoil. For the ones that correspond to the Statoil catalogue, execution time over EPDS is on average several seconds, which is comparable to the numbers in our experiments. Moreover, in general for the simplest access points the execution takes from seconds to minutes, and for the ones corresponding to complex analytical tasks it can take overnight; thus,

⁴ Oracle 10g, HP ProLiant server with 24 Intel Xeon CPUs (X5650 @ 2.67 GHz), 283 GB RAM

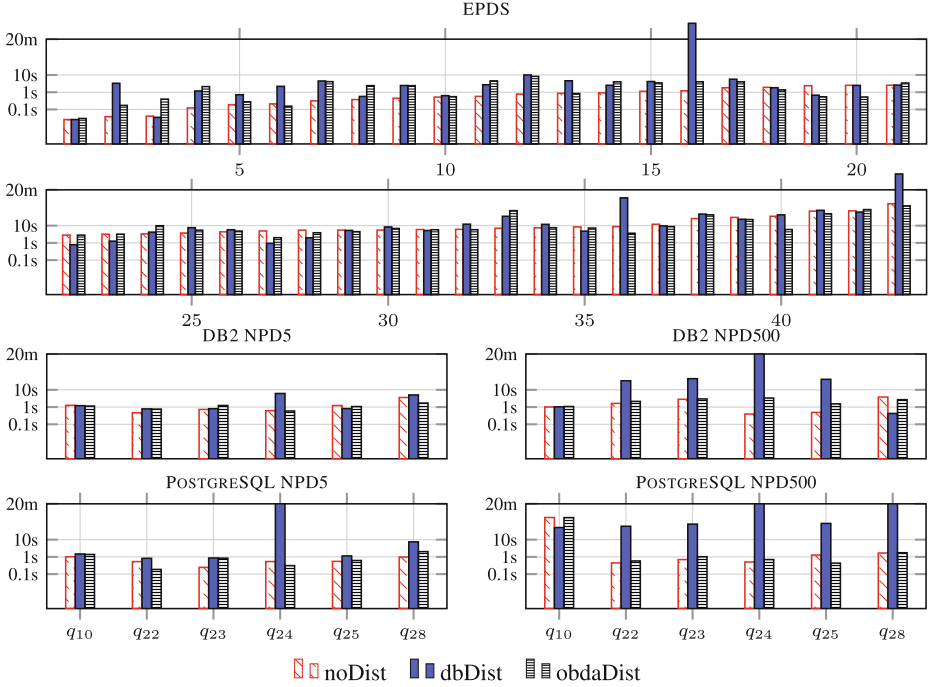


Fig. 4. Query execution times for 43 Statoil catalogue queries over EPDS, sorted after *noDist*; and 6 selected NPD benchmark queries over NPD5 and NPD500. Logarithmic scale

the response time for even the slowest of our queries should not be too slow for Statoil geologists.

We also analysed why some of our queries require longer execution times than others, and why some queries failed, i.e., timed out. The main reason for slow queries is redundant self-joins in the SQL queries generated after the SPARQL queries were rewritten and unfolded. The problem comes from the fact that for many EPDS tables used in mappings, primary keys are not declared; thus our optimisation techniques described in Section 4.2, that remove self-joins, become inapplicable. Another common pattern in the SQL queries that is responsible for slowing down execution time is redundancy of subqueries introduced by mappings, which is not detected by our query optimisation techniques and redundancy of query answers. The latter factor is important since the answer sets for 30 out of 43 queries that did not time out contained duplicates; among them, the mean ratio of redundant answers is 51.6% while the maximum is 99.8% or 83k redundant answers.

In the *dbDist* *obdaDist* experiments we eliminated the redundant answers at a reasonable cost: execution times of *dbDist* and *obdaDist* are similar to *noDist*. Moreover, *obdaDist* outperforms *dbDist* in 67% cases, and in 2 cases *dbDist*

execution gave a time out, while *obdaDist* never did so, which shows the benefits of our approach over *dbDist*.

6.3 Running Queries in Controlled Environment

We also conducted experiments in a controlled environment, which is important since EPDS is a production server, and hence the load on the server varies constantly and affects the results of experiments. For controlled experiments we used our own server and the NPD benchmark [18], a benchmark for OBDA systems. Compared to EPDS, our controlled setting contains smaller datasets: from 240MB to 25GB with +1.3B triples. We ran the experiments in an HP Proliant server with 24 Intel Xeon CPUs (144 cores@3.47GHz), 106GB of RAM and five 1TB 15K RPM HD using Ubuntu 12.04 64-bit edition. The tests were performed using DB2 and PostgreSQL as underlying database engines.

In the lower four plots in Figure 4 we present *noDist*, *dbDist*, and *obdaDist* experiments with 6 NPD queries whose performance was affected by the use of the `distinct`. Each experiment was conducted over two datasets: NPD5 and NPD500 corresponding to the scaling of NPD 5 and 500 times, and the results are presented separately for PostgreSQL and DB2. Note that there are 30 queries in the last version of the NPD benchmark (v1.7), while we report experiments with only the ones where the use of `distinct` had an impact (positive or negative) on performance.

Our experiments show that execution of the 6 queries without `distinct` is in most cases under 1s. At the same time if one uses `distinct` in a naive way, by delegating it to the database engines, it is extremely detrimental to performance, leading even more queries to time out (see *dbDist* experiments). Nevertheless, with our treatment of `distinct` (see *obdaDist*), in most cases the queries perform orders of magnitude better than with the naive implementation; as in the EPDS case, the optimisation seems to be more effective when the ratio of redundant answers is high, e.g., the highest ratio of redundant answers is 90% for query q_{24} , and the mean ratio for the queries is around 50%.

Importantly, compared to queries without `distinct`, the overhead of removing redundancy with our techniques is small. In some cases the execution time in *obdaDist* was even better than in *noDist* because the number of the returned results shipped to the end user is significantly reduced. However, there is one interesting exceptional case where the first query in *dbDist* over PostgreSQL (and the last query over DB2) performs better than *noDist* and *obdaDist*. This phenomenon still requires further investigation.

7 Lessons Learned and Future Work

During the course of the project with Statoil we had a unique opportunity to deploy OBDA technology in a real industrial setting, to understand the limitations of the current technology, to address these limitations, and to get new ideas for further research. Our OBDA solution for Statoil has improved efficiency of

data gathering for geologists by allowing them to (i) efficiently express information needs as ontological queries and (ii) efficiently execute these queries over the OBDA deployment.

Regarding Item (i), our experiments show that, although we achieved our objectives, there is still room for improvement. Indeed, our OBDA deployment addresses Requirements 1 and 2 from Section 5: it has enough ontological terms to express queries in the query catalogue, and they all are ‘connected’ to EPDS via mappings (thus addressing Req. 1); and the ontology underlying the deployment has a wide range of terms coming from several respected sources (thus addressing Req. 2). Importantly, most of the ontology was developed using our semi-automatic deployment module and only minor manual work was required to extend the ontology in order to cover the query catalogue. Of course, the resulting ontology by no means gives an exhaustive coverage of the oil and gas domain, but we see it as a good starting point for developing a more sophisticated ontology, and as a suitable point of entry for OBDA to Statoil’s EPDS database. Moreover, from the research point of view, we plan to develop ontology bootstrapping techniques that can be more targeted to specific scenarios, e.g., that could work not only with database schemata and data, but also with log files of queries, precomputed views and other assets specific for a given domain or scenario. We also plan to compare performance and quality of our bootstrapper with existing analogous systems. Finally, in order to improve usability of our OBDA deployment and to facilitate construction of ontological query to end users with limited system experience we work on intuitive query interfaces [1–4, 28–31].

Regarding Item (ii), the execution time of most queries from the Statoil catalogue was impressive and comparable to the performance of Statoil’s existing access points, thus addressing Requirement 3 from Section 6. Ensuring that the remaining queries do not time out requires further research and we currently look into this, e.g., we work on optimisation techniques to eliminate redundant self-joins. Besides, we plan to conduct experiments with Statoil queries using other available OBDA query processing engines and compare results with the outcome of experiments reported in this paper. Finally, our treatment of `distinct` led to a significant improvement in performance in comparison to standard processing of `distinct` in OBDA systems. In our opinion, not only `distinct`, but also other query operators can benefit if the query planning is at least partially delegated from SQL to OBDA processing engines, where the planner exploits statistics of concepts and properties and the structure of the SPARQL query; this, however, requires further research.

For future work we plan to integrate our OBDA deployment in the business processes of Statoil engineers and IT personnel. Moreover, we are working on a better integration of the deployment with existing Statoil analytical and data visualisation tools, and have already integrated our solution with a geospatial data visualisation tool. Another challenge that we plan to address in the remaining stages of the project is extension of the deployment from EPDS and NPD FP to other Statoil data sources. This will require development of a distributed

query processing infrastructure, extension of the Statoil query catalogue, and experiments with our deployment and query execution solutions. An important request that we got while evaluating the OBDA deployment with Statoil engineers is to allow for users' interaction with the system: engineers would like to send their feedback to the OBDA system, e.g., by saying that some ontological terms are missing, or that some answers are wrong, and to get explanations from the system, e.g., to get provenance to query answers that include the name of the database where the answers came from as well as the exact tables that were used in the mappings. Enabling such user interaction is also an area of future work. We also work on developing access control mechanisms for our OBDA deployment that can ensure that users can access only the data they are allowed to see [10,11].

8 Conclusion

In this paper we presented a description of a data analyses routine at Statoil and challenges with access to business critical data required for these analyses. These challenges delay the analytical tasks significantly, thus addressing them is of a high importance for Statoil. Additionally, the challenges are representative for large data intensive industries and a good solution would be beneficial not only for Statoil, but for a wide range of enterprises. We believe that OBDA technology is a promising way to address the challenges while to the best of our knowledge existing off-the-shelf OBDA solutions can not be directly applied to do the job. Thus, we developed an OBDA solution that is capable of dealing with the challenges and is equipped with a deployment module *BOOTOX* for semi-automatic creation of ontologies and mappings, and a query processing module *Ontop* that ensures efficient OBDA query processing. We deployed our solution at Statoil and evaluated it using three requirements that are based on interviews of Statoil engineers, analyses of their business processes, and in particular on a catalogue of typical information needs of Statoil geologists. Results of the evaluation suggest that our OBDA deployment in Statoil was successful, while there is still a number of both technical and research challenging that should be addressed to improve our solution.

Acknowledgements. This work was partially funded by the EU project *Optique* [14, 15] (FP7-ICT-318338) and the EPSRC projects *MaSI*³, *Score!*, and *DBOnto*.

References

1. Arenas, M., et al.: Enabling faceted search over OWL 2 with *semfacet*. In: *OWLED* (2014)
2. Arenas, M., et al.: Faceted search over ontology-enhanced RDF data. In: *CIKM* (2014)
3. Arenas, M., et al.: *SemFacet*: semantic faceted search over Yago. In: *WWW, Companion Volume* (2014)

4. Arenas, M., et al.: Towards semantic faceted search. In: WWW, Companion Volume (2014)
5. Calvanese, D., et al.: The MASTRO system for ontology-based data access. In: Semantic Web 2.1 (2011)
6. Calvanese, D., et al.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. JAR **39**(3) (2007)
7. Console, M., et al.: Efficient approximation in DL-Lite of OWL 2 ontologies. In: DL (2013)
8. Crompton, J.: Keynote talk at the W3C Workshop on Sem. Web in Oil & Gas Industry (2008)
9. Doan, A., et al.: Principles of Data Integration. Kaufmann (2012)
10. Grau, B.C., et al.: Controlled query evaluation for datalog and OWL 2 profile ontologies. In: IJCAI (2015)
11. Cuenca Grau, B., Kharlamov, E., Kostylev, E.V., Zheleznyakov, D.: Controlled query evaluation over OWL 2 RL ontologies. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 49–65. Springer, Heidelberg (2013)
12. Jimenez-Ruiz, E., et al.: Large-scale interactive ontology matching: algorithms and implementation. In: ECAI (2012)
13. Jiménez-Ruiz, E., et al.: BootOX: practical mapping of RDBs to OWL 2. In: ISWC (2015)
14. Kharlamov, E., et al.: Optique 1.0: semantic access to big data: the case of Norwegian petroleum directorate factpages. In: ISWC Posters & Demos (2013)
15. Kharlamov, E., et al.: Optique: towards OBDA systems for industry. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 125–140. Springer, Heidelberg (2013)
16. Kharlamov, E., et al.: How semantic technologies can enhance data access at siemens energy. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 601–619. Springer, Heidelberg (2014)
17. Kontchakov, R., Rezk, M., Rodríguez-Muro, M., Xiao, G., Zakharyashev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 552–567. Springer, Heidelberg (2014)
18. Lanti, D., et al.: The NPD benchmark: reality check for OBDA systems. In: EDBT (2015)
19. Pinkel, C., et al.: RODI: a benchmark for automatic mapping generation in relational-to-ontology data integration. In: Gandon, F., Sabou, M., Sack, H., d’Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, pp. 21–37. Springer, Heidelberg (2015)
20. Pinkel, C., et al.: IncMap: pay as you go matching of relational schemata to OWL ontologies. In: OM (2013)
21. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008)
22. Priyatna, F., et al.: Formalisation and experiences of R2RML-based SPARQL to SQL query translation using Morph. In: WWW (2014)
23. Rodríguez-Muro, M., Calvanese, D.: High performance query answering over DL-Lite ontologies. In: KR (2012)

24. Rodríguez-Muro, M., Kontchakov, R., Zakharyashev, M.: Ontology-based data access: *ontop* of databases. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 558–573. Springer, Heidelberg (2013)
25. Sequeda, J., et al.: Survey of directly mapping SQL databases to the semantic web. *Knowledge Eng. Review* **26**(4) (2011)
26. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the norwegian petroleum directorate’s factpages as semantic web data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 162–177. Springer, Heidelberg (2013)
27. Solimando, A.: Detecting and correcting conservativity principle violations in ontology mappings. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part II. LNCS, vol. 8797, pp. 545–552. Springer, Heidelberg (2014)
28. Soyly, A., et al.: OptiqueVQS: towards an ontology-based visual query system for big data. In: MEDES (2013)
29. Soyly, A., Giese, M., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: Towards exploiting query history for adaptive ontology-based visual query formulation. In: Closs, S., Studer, R., Garoufallou, E., Sicilia, M.-A. (eds.) MTSR 2014. CCIS, vol. 478, pp. 107–119. Springer, Heidelberg (2014)
30. Soyly, A., et al.: Why not simply Google? In: NordiCHI (2014)
31. Soyly, A., Skjæveland, M.G., Giese, M., Horrocks, I., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D.: A preliminary approach on ontology-based visual query formulation for big data. In: Garoufallou, E., Greenberg, J. (eds.) MTSR 2013. CCIS, vol. 390, pp. 201–212. Springer, Heidelberg (2013)
32. Spanos, D.-E., et al.: Bringing relational databases into the semantic web: a survey. In: *Semantic Web 3.2* (2012)
33. Stoilos, G., Stamou, G., Kollias, S.D.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)

BOOTOX: Practical Mapping of RDBs to OWL 2

Ernesto Jiménez-Ruiz¹(✉), Evgeny Kharlamov¹, Dmitriy Zheleznyakov¹,
Ian Horrocks¹, Christoph Pinkel², Martin G. Skjæveland³, Evgenij
Thorstensen³, and Jose Mora⁴

¹ Department of Computer Science, University of Oxford, Oxford, UK
`ernesto.jimenez.ruiz@cs.ox.ac.uk`

² Fluid Operations AG, Walldorf, Germany

³ Department of Informatics, University of Oslo, Oslo, Norway

⁴ Sapienza Università di Roma, Rome, Italy

Abstract. Ontologies have recently become a popular mechanism to expose relational database (RDBs) due to their ability to describe the domain of data in terms of classes and properties that are clear to domain experts. Ontological terms are related to the schema of the underlying databases with the help of mappings, i.e., declarative specifications associating SQL queries to ontological terms. Developing appropriate ontologies and mappings for given RDBs is a challenging and time consuming task. In this work we present BOOTOX, a system that aims at facilitating ontology and mapping development by their automatic extraction (i.e., bootstrapping) from RDBs, and our experience with the use of BOOTOX in industrial and research contexts. BOOTOX has a number of advantages: it allows to control the OWL 2 profile of the output ontologies, bootstrap complex and provenance mappings, which are beyond the W3C direct mapping specification. Moreover, BOOTOX allows to import pre-existing ontologies via alignment.

1 Introduction

The Semantic Web community has actively investigated the problem of bridging the gap between relational databases and ontologies. One of the main targets behind this effort is to enable the access to the data stored in databases via Semantic Web technologies. An advantage of this approach is that ontologies provide a formal specification of the application domain that is close to the end-users' view of the domain, while databases are oriented towards an efficient data storage and retrieval and thus represent the data using structures that often not intuitive to end-users.

Manually building an ontology and connecting it to the data sources via mappings is, however, a costly process, especially for large and complex databases (a typical scenario in industry [20,22]). The cost of this manual process will typically be even more severe when dealing with multiple databases, e.g., in the context of accessing the Deep Web [16]. To aid this process, tools that can extract a preliminary ontology and mappings from database schemata play a critical role.

In the literature one can find a broad range of approaches to bootstrap an ontology and mappings from a relational database schema. The interested reader may have a look at the following surveys [34, 41]. These approaches can be classified with respect to different aspects such as: *(i)* level of automation (i.e., manual, semi-automatic, automatic), *(ii)* type of mappings (i.e., complex or direct mappings), *(iii)* language of the bootstrapped mappings and the ontology, *(iv)* reuse of external vocabularies (e.g., domain ontologies or thesauri), and *(v)* purpose (e.g., OBDA, constraint validation, database integration, ontology learning).

In this paper we present BOOTOX¹ (not) yet another ontology and mapping bootstrapper. Our main motivation to implement a new bootstrapper is to give more flexibility with respect to the classification aspects described above. Most of the existing approaches commit to concrete purposes or to a concrete ontology expressiveness. BOOTOX allows to define different “profiles” depending on the application scenario and the required Semantic Web technologies. For example, if the bootstrapped ontology is to be used in a so-called Ontology Based Data Access (OBDA) scenario where the ontology provides a virtual access layer to the data, OWL 2 QL will be chosen as the ontology language as it is required by the query rewriting engine.² Nevertheless, if the data is materialised, one could opt for other OWL 2 profiles depending on the used query answering engine. BOOTOX also allows to import domain ontologies, which will be integrated to the bootstrapped one via alignment [18] or directly mapped to the database. BOOTOX follows the W3C direct mappings directives to connect the ontological vocabulary to the relational database; moreover, it offers a suit of advanced techniques for bootstrapping of mapping that are beyond the direct ones. Furthermore, it extends the bootstrapped mappings with provenance information.

We have tested BOOTOX and compared it to several bootstrapping systems over a number of databases and use cases from industrial and research contexts, including the relational-to-ontology benchmark suite RODI [29].

The paper is organised as follows. Section 2 introduces the concepts behind semantic access to databases. Section 3 presents the problem of bootstrapping. Section 4 describes the techniques implemented in BOOTOX. Section 5 presents the scenarios where we deployed BOOTOX and the conducted experiments to evaluate the quality of the bootstrapping. In Section 6 we provide a summary of relevant related work, and we conclude in Section 7.

2 Exposing Relational Data Through Ontologies

The main idea behind exposing relational data via an ontology is to provide the user with access to the data store via the use of a domain specific vocabulary of classes, i.e., unary predicates, and properties, i.e., binary predicates, that the user is familiar with. This vocabulary is related to the database schema via view definitions, called *mappings*; thus, technical details of the database schema

¹ <http://www.cs.ox.ac.uk/isg/tools/BootOX/>

² The language underlying OWL 2 QL has the first-order (FO) rewritability property [8].

| |
|--|
| Well (<u>id</u> , name ^{NN} , type) Field (<u>id</u> , name, status ^{CK} , intersects_field ^{FK}) Operator (<u>id</u> , name ^{NN}) Operator_Field (operator ^{FK} , field ^{FK}) Wellbore (<u>id</u> , name ^{UQ} , content ^{CK} , depth ^{CK} , well ^{FK} , location ^{FK}) ExplorationWellbore (wellbore ^{FK} , seismic_location) DevelopmentWellbore (wellbore ^{FK} , production_facility) |
|--|

Fig. 1. Relational Database Schema. Primary keys are underlined. FK stands for Foreign Key (the foreign key name indicates the target table), NN stands for Not Null, UQ stand for Unique and CK stands for Check

are hidden from end-users. Using an example from the oil industry domain, we now intuitively introduce the main concepts that are needed to understand the approach and refer the reader to [31] for details.

Relational Databases. In the relational data model, a database consists of a schema and instance, where the schema is a set of table names with corresponding attribute names and constraints, e.g., primary and foreign keys. The instance ‘populates’ tables of the schema with tuples by assigning values to the tables’ attributes. Figure 1 shows our running example, a fragment of a relational database based on the oil industry domain. For example, a Wellbore is given a name that is unique, has a content (e.g., GAS, OIL, or DRY), has a depth (e.g., 12,500 feet), belongs to a Well, and is located in a Field.

Ontologies. An ontology is usually referred to as a ‘conceptual model’ of (some aspect of) the world. It introduces the vocabulary of classes and properties that describe various aspects of the modelled domain. It also provides an explicit specification of the intended meaning of the vocabulary by describing the relationships between different vocabulary terms. These relationships are specified with special first-order formulae, also called axioms, over the vocabulary. An ontology can be thought of simply as a set of such axioms—i.e., a logical theory. Besides axioms, an ontology can contain ontological facts, which can be specified as first-order atoms with constants, but not variables. These constants are usually interpreted as (representations of) objects from the domain. Viewing an ontology as a logical theory opens up the possibility of using automated reasoning for different tasks, including checking an ontology’s internal consistency (checking whether the ontology does not entail ‘false’, i.e., if it is logically consistent), query answering, and other (non-)entailments. OWL 2 is a W3C standard ontology language and it is heavily used in the Semantic Web community. OWL 2 has profiles:³ EL, QL, and RL, that have different favourable computational properties.

³ OWL 2 profiles: <http://www.w3.org/TR/owl2-profiles/>

Mappings. Mappings declaratively define how ontological terms are related to terms occurring in the relational schema and are essentially view definitions of the following form that declare how to populate classes with objects—in OWL 2 objects are represented with Uniform Resource Identifiers (URIs)—and to populate properties with object-object and object-value pairs:

$$\begin{aligned} \text{Class}(f_o(x)) &\leftarrow \text{SQL}(x), \\ \text{objectProperty}(f_o(x), f_o(y)) &\leftarrow \text{SQL}(x, y), \\ \text{dataProperty}(f_o(x), f_v(y)) &\leftarrow \text{SQL}(x, y), \end{aligned}$$

where $\text{SQL}(x)$ and $\text{SQL}(x, y)$ are SQL queries with respectively one and two output variables, and f_o, f_v are functions that ‘cast’ values returned by the SQL queries into respectively objects, i.e. URIs, and values.⁴ Classes are populated with URIs $f_o(x)$ computed from the values x returned by $\text{SQL}(x)$. Properties can relate two objects, e.g., by stating that a wellbore has a particular location, or assigning a value to an object, e.g., stating that a field has a particular name (a string); and they are respectively populated with pairs of objects $f_o(x), f_o(y)$ or pairs of an object $f_o(x)$ and value $f_v(y)$ computed from the values x and y returned by the SQL query. For our running example, we may create the following mappings:

$$\begin{aligned} \text{Wellbore}(f_o(\text{id})) &\leftarrow \text{SELECT id FROM Wellbore}, \\ \text{hasLocation}(f_o(\text{id}), f_o(\text{location})) &\leftarrow \text{SELECT id, location FROM Wellbore}, \\ \text{hasContent}(f_o(\text{id}), f_v(\text{content})) &\leftarrow \text{SELECT id, content FROM Wellbore}. \end{aligned}$$

Query Answering. Consider a data access instance $(\mathcal{D}, \mathcal{V}, \mathcal{O}, \mathcal{M})$, where \mathcal{D} is an RDB, \mathcal{V} is an ontological vocabulary, \mathcal{O} is a set of ontological axioms over \mathcal{V} , and \mathcal{M} is a set of mappings between \mathcal{V} and \mathcal{D} . There are two approaches to answer a query Q over \mathcal{V} : (i) *materialisation*: ontological facts are materialised (i.e., classes and properties participating in mappings are populated with individuals by evaluating SQL queries participating in mappings) and this gives a set of ontological facts \mathcal{A} and then Q is evaluated over \mathcal{O} and \mathcal{A} with standard query-answering engines for ontologies, or (ii) *virtual*: Q should be first rewritten into an SQL query SQL using \mathcal{O} and \mathcal{M} and then SQL should be executed over \mathcal{D} .

In either case, the ontology and mappings are the backbone of query processing in this data access approach; thus, the problem of obtaining them has to be addressed in any implementation of such approach. The exact kind of ontology and mappings required for data access depends on the application scenario. For example, in a scenario where virtual query answering is required, the ontology should be in OWL 2 QL. In the case of materialisation the ontology should fall in the profile supported by the underlying systems, e.g., *EOLO* [42] can do query answering over OWL 2 EL ontologies, and thus the bootstrapping should output an EL ontology; a semantic faceted search system *SEMFACTET* [2–4, 15] relies on *RDFox* [28] that can do query answering over OWL 2 RL ontologies and thus the bootstrapper facilitating *SEMFACTET* should produce RL ontologies; *PAGODA* [47] does query answering over the whole OWL 2 and can work with any bootstrapper that produces OWL 2 ontologies.

⁴ These functions ensure coherent generation of URIs that respects primary and foreign keys.

3 Bootstrapping: Problem Definition

In this section we define the tasks for bootstrapping ontologies and mappings, or *assets*, from RDBs and quality metrics for these tasks. The tasks are the following:

- (i) *Semantic access instance generation*: Given a relational database \mathcal{D} , generate an instance $(\mathcal{D}, \mathcal{V}, \mathcal{O}, \mathcal{M})$. This task can be naturally divided into two sub-tasks.
 - *Vocabulary and Ontology generation*: Given \mathcal{D} , create a vocabulary \mathcal{V} and an ontology \mathcal{O} over \mathcal{V} .
 - *Mapping generation*: Given \mathcal{D} , \mathcal{V} , and \mathcal{O} create a set of mappings \mathcal{M} relating \mathcal{D} with \mathcal{V} .
- (ii) *Importing*: Given an instance $(\mathcal{D}, \mathcal{V}, \mathcal{O}_1, \mathcal{M})$ and an ontology \mathcal{O}_2 , return an instance $(\mathcal{D}, \mathcal{V}, \mathcal{O}, \mathcal{M})$, where \mathcal{O} is the alignment of \mathcal{O}_1 and \mathcal{O}_2 .

Task (ii) is important in applications where ontologies (partially) describing the domain of interest have been already created and users want to incorporate them into their semantic data access system.

The bootstrapping of the ontologies and the mappings enables a (preliminary) deployment of semantic data access system. However, the bootstrapped assets should meet some minimal requirements so that they can be usable in practice. We have identified the following *metrics* to measure the quality of generated assets:

- (1) *Ontology language*: compliance with standard ontology languages with well-defined semantics like OWL 2 and its profiles to enable the use of a wide-range of Semantic Web technologies. Note that the choice of the ontology language (e.g., one of OWL 2 profiles) also affects suitability of the different query answering engines.
- (2) *Mapping language*: compliance with standard directives like the W3C direct mapping specification⁵ and standard W3C mapping languages like R2RML.⁶
- (3) *Query coverage*: suitability of the ontology vocabulary to formulate the queries that the user is interested in.
- (4) *Query results*: accuracy of the query results obtained with the bootstrapped instance in the sense that the query answer should satisfy the user's expectations.

4 Bootstrapping Techniques in BOOTOX

In this section we present our bootstrapping techniques implemented in BOOTOX. Section 4.1 focuses on the techniques to generate the ontological vocabulary and the axioms over this vocabulary, where we give special attention

⁵ Direct mappings: <http://www.w3.org/TR/rdb-direct-mapping/>

⁶ R2RML language: <http://www.w3.org/TR/r2rml/>

to the concrete language of the generated ontology. In Section 4.2 we describe how the database terms are linked to the ontological vocabulary. Furthermore, we also present how mappings are enhanced with provenance information. Section 4.3 gives an overview of the incorporated ontology alignment techniques to import pre-existing ontologies. Finally, Section 4.4 summarises the conformance with the requirements presented in Section 3.

4.1 Vocabulary and Ontology Generation

The general rule to create ontological vocabulary from a relational schema would translate (i) each (non-binary) table into an OWL class; (ii) each attribute not involved in a foreign key into an OWL datatype property; (iii) each foreign key into an OWL object property. Generation of axioms, however, is usually a more involved process. BOOTOX follows Table 1 to automatically create the vocabulary and a set of OWL 2 axioms from the listed database features.⁷ One could opt for adding all the axioms associated with a feature or only a selection of them depending on the intended purpose. In the remainder of the section we will discuss the main considerations regarding bootstrapping that should be taken into account.

Closed-world vs Open-world Semantics. Modelling database features in OWL 2 inevitably leads to different interpretations due to the Closed-world (CWA) and Open-world assumptions (OWA). In a relational database, every fact that occurs in the database is true, and all other facts are false (CWA). In an ontology, the truth value of some facts may be unknown, i.e., they are neither provably true nor provably false (OWA). Moreover, constraints have a very different meaning in databases and ontologies [27, 34]. In a database, constraints define valid database states, and updates that would violate these constraints are simply rejected. In an ontology, constraints are treated as material implications, and may lead to the derivation of “new” facts that are not explicitly stated in the data. Such derivations may produce unintended consequences,⁸ as we discuss next.

Table 1 presents a possible encoding of relational database features as OWL 2 axioms. The encoding, in general, does not lead to only one option, but several possible OWL 2 axioms. For example, considering the running example of Figure 1, for the data attribute *name* in table *Field* we apply encoding pattern (3) in Table 1, which proposes to add five different axioms. The general rule translates the data attribute *name* as a data property declaration axiom. In addition one may opt to add global restrictions (e.g., ‘*name* Domain: *Field*’)⁹ and/or local restrictions (e.g., ‘*Field* SubClassOf: *name* some *xsdstring*’). The use of global and/or local restrictions will lead to different interpretations and

⁷ When not stated the contrary in Table 1, a class C_T , an object property P_f , a data property R_a and a datatype d_t represent the ontology encoding of a table T , a foreign key fk , a data attribute a , and an SQL type t , respectively.

⁸ Note that the use of disjointness axioms may help in the detection of such “errors”.

⁹ For writing OWL 2 axioms we use the Manchester OWL Syntax [17].

Table 1. Encoding of relational database features as OWL 2 axioms. OWL 2 axioms are expressed in the Manchester OWL Syntax [17]. * Enumeration with only one literal

| # | RDB feature | Ontology feature | OWL 2 axiom | OWL 2 profile | | |
|--|---|---|---|---------------|----|----|
| | | | | QL | RL | EL |
| (1) | Non-binary Relation / Table T | A class C_T for the non-binary table | Class: C_T | ✓ | ✓ | ✓ |
| (2) | Binary Relation or Many-to-Many Table referencing tables T_1 and T_2 | A property P (and its inverse Q) associated to the classes C_{T_1} and C_{T_2} with local and/or global constraints | ObjectProperty: P | ✓ | ✓ | ✓ |
| | | | Q InverseOf: P | ✓ | ✓ | - |
| | | | P Domain: C_{T_1} | ✓ | ✓ | ✓ |
| | | | P Range: C_{T_2} | ✓ | ✓ | ✓ |
| | | | C_{T_1} SubClassOf: P some C_{T_2} | ✓ | - | ✓ |
| C_{T_1} SubClassOf: P only C_{T_2} | - | ✓ | - | | | |
| (3) | Data attribute in table T of (sql) type t | A property R_a associated to the class C_T and datatype d_t with local and/or global constraints. | DataProperty: R_a | ✓ | ✓ | ✓ |
| | | | R_a Domain: C_T | ✓ | ✓ | ✓ |
| | | | R_a Range: d_t | ✓ | ✓ | ✓ |
| | | | C_T SubClassOf: R_a some d_t | ✓ | - | ✓ |
| | | | C_T SubClassOf: R_a only d_t | - | ✓ | - |
| (4) | Foreign Key in table T_1 , referencing T_2 , no intersection with or strict subset of primary key | A property P_f associated to the classes C_{T_1} and C_{T_2} with local and/or global constraints | ObjectProperty: P_f | ✓ | ✓ | ✓ |
| | | | P_f Domain: C_{T_1} | ✓ | ✓ | ✓ |
| | | | P_f Range: C_{T_2} | ✓ | ✓ | ✓ |
| | | | C_{T_1} SubClassOf: P_f some C_{T_2} | ✓ | - | ✓ |
| | | | C_{T_1} SubClassOf: P_f only C_{T_2} | - | ✓ | - |
| (5) | Foreign Key is the primary key in T_1 , ref. T_2 | Class C_{T_1} is subsumed by class C_{T_2} | C_{T_1} SubClassOf: C_{T_2} | ✓ | ✓ | ✓ |
| (6) | Foreign Key in table T referencing the same table | A property P_f associated to class C_T with a self-restriction. The property P_f may also be declared with several characteristics | C_T SubClassOf: P_f some C_T | ✓ | - | ✓ |
| | | | C_T SubClassOf: P_f some Self | - | - | ✓ |
| | | | P_f Characteristics: Transitive | - | ✓ | ✓ |
| | | | P_f Characteristics: Symmetric | ✓ | ✓ | - |
| | | | P_f Characteristics: Reflexive | ✓ | ✓ | ✓ |
| (7) | Primary Key or Unique constraint in table T_1 on a foreign key fk referencing T_2 | Key axiom for class C_{T_1} and property P_f . P_f is associated to (local and/or global) cardinality constraints | C_{T_1} HasKey: P_f | - | ✓ | ✓ |
| | | | P_f Characteristics: Functional | - | ✓ | - |
| | | | P_f Characteristics: InverseFunctional | - | ✓ | - |
| | | | C_{T_1} SubClassOf: P_f exactly 1 C_{T_2} | - | - | - |
| | | | C_{T_1} SubClassOf: P_f max 1 C_{T_2} | - | ✓ | - |
| C_{T_1} SubClassOf: P_f some C_{T_2} | ✓ | - | ✓ | | | |
| (8) | Primary Key or Unique constraint on a data attribute a of (sql) type t in table T | Key axiom for class C_T and data property R_a . R_a is associated to (local and/or global) cardinality constraints. | C_T HasKey: R_a | - | ✓ | ✓ |
| | | | R_a Characteristics: Functional | - | ✓ | ✓ |
| | | | C_T SubClassOf: R_a exactly 1 d_t | - | - | - |
| | | | C_T SubClassOf: R_a max 1 d_t | - | ✓ | - |
| C_T SubClassOf: R_a some d_t | ✓ | - | ✓ | | | |
| (9) | Composed Primary Key in table T | Key axiom for the class C_T and the data and object properties involved in primary key | C_T HasKey: $R_1 \dots R_n P_1 \dots P_n$ | - | ✓ | ✓ |
| (10) | Not Null Constraint on a data attribute in T , type t | Existential or cardinality restriction over R_a in C_T | C_T SubClassOf: R_a min 1 d_t | ✓ | - | - |
| | | | C_T SubClassOf: R_a some d_t | ✓ | - | ✓ |
| (11) | Not Null Constraint on a foreign key in T_1 , ref. T_2 | Existential or cardinality restriction over P_f in C_T | C_{T_1} SubClassOf: P_f min 1 C_{T_2} | - | - | - |
| | | | C_{T_1} SubClassOf: P_f some C_{T_2} | ✓ | - | ✓ |
| (12) | Check Constraint on data attribute a of type t in table T listing possible values $v_1 \dots v_n$ ($n \geq 1$) | Enumeration of literals: in a restriction in class C_T and/or as a range of R_a . Alternatively, one could create subclasses for each of the values | C_T SubClassOf: R_a some $\{v_1 \dots\}$ | - | - | ✓* |
| | | | C_T SubClassOf: R_a only $\{v_1 \dots\}$ | - | - | - |
| | | | C_T SubClassOf: R_a value v_1 | - | - | ✓ |
| | | | R_a Range: $\{v_1 \dots\}$ | - | - | ✓* |
| $C_{T_{v_i}}$ SubClassOf: C_T | ✓ | ✓ | ✓ | | | |
| (13) | Check Constraint on attrib. a in table T restricting numerical range of t | Datatype restriction: in a class restriction in C_T and/or as a range of R_a | C_T SubClassOf: R_a some $d_t [> x]$ | - | - | - |
| | | | C_T SubClassOf: R_a only $d_t [> x]$ | - | - | - |
| | | | R_a Range: $d_t [> x]$ | - | - | - |
| (14) | Several data attributes $a_1 \dots a_n$ in different tables $T_1 \dots T_n$ with the same name and type t | Group properties $R_1 \dots R_n$ under new superproperty R_a or merge $R_1 \dots R_n$ into new property R_a | R_i SubPropertyOf: R_a | ✓ | ✓ | ✓ |
| | | | R_a Domain: C_{T_1} or \dots or C_{T_n} | - | - | - |
| | | | C_{T_i} SubClassOf: R_a some d_t | ✓ | - | ✓ |
| | | | C_{T_i} SubClassOf: R_a only d_t | - | ✓ | - |
| (15) | T_1 and T_2 not involved in an inheritance relationship | Class C_{T_1} is disjoint with class C_{T_2} | C_{T_1} DisjointWith: C_{T_2} | ✓ | ✓ | ✓ |

potentially to different logical consequences when combined with data facts. For example, the fact ‘*operator987* Facts: *name* Statoil’ in combination with the domain axiom for *Field* and *name* will lead to the undesired consequence ‘*operator987* Types: *Field*’. In this case, if the property *name* is to be used in different contexts, using only local restrictions seems to be more appropriate.

Profiling the Ontology. BOOTOX takes into account the concrete language (i.e., one of OWL 2 profiles) that is generated from the relational database features, which will enable the use of different Semantic Web technologies (e.g., query answering engines). As mentioned above, database features can be encoded using different axioms which may lead to different OWL 2 profiles. For example, primary keys and unique constraints (patterns (7) and (8) in Table 1) can be modelled with the OWL 2 construct *HasKey* (e.g., ‘*Well* *HasKey*: *id*’), which is supported in OWL 2 RL and OWL 2 EL, but must be avoided if the target ontology language is OWL 2 QL. Alternatively (or in addition) one could use global and local cardinality restrictions (e.g., ‘*id* *Characteristics*: *Functional*’ and ‘*Well* *SubClassOf*: *id* exactly 1 *xsdint*’, respectively), which leads to similar issues since not all profiles support them. For example, none of the OWL 2 profiles support exact cardinalities. If a profile does not support a type of axiom, an approximation is used instead (e.g., ‘*Well* *SubClassOf*: *id* max 1 *xsdint*’ that is supported in OWL 2 RL, or ‘*Well* *SubClassOf*: *id* some *xsdint*’ that is allowed in both OWL 2 QL and EL profiles). However, a representative approximation or alternative may not be always available for a database feature, as for the *datatype restrictions* (pattern (13) in Table 1), which is not supported by any of the profiles. BOOTOX, when the required ontology output is one of the OWL 2 profiles, keeps this knowledge as OWL 2 annotation axioms (e.g., ‘*depth* *Annotations*: *annmax_value*“12,500”’). These axioms have no logical impact on the ontology, but they can potentially be used in user interfaces to guide the formulation of queries (e.g., [38–40]).

Selection of Ontology Axioms. BOOTOX, in the default setting, encodes the database features with all axioms suitable for a required OWL 2 profile. The user can optionally select the kind of preferred axioms (i.e., local or global restrictions). Additionally, the suggested OWL 2 axioms may require an extra validation step to accurately represent a database feature. This is the case of pattern (6) in Table 1 where the generated object property can be declared with different characteristics which may not necessarily hold in the database. For example, in our running example from Figure 1, the foreign key *intersects_field* in table *Field* represents a self-reference to the same table. In this case, the foreign key can be encoded as a *reflexive* (a field intersects with itself) and *symmetric* object property; transitivity, however, does not necessarily apply for this property.

Mapping SQL Datatypes to OWL 2. There is a clear mapping between SQL datatypes to XML schema datatypes. However, there are some XML schema datatypes that are not built-in datatypes in OWL 2. For example, *xsddate* is not supported in OWL 2. Furthermore, OWL 2 profiles also include specific

restrictions on the supported datatypes. For example, *xsdboolean* or *xsddouble* are not supported in OWL 2 QL nor OWL 2 EL. BOOTOX addresses this issue by mapping SQL datatypes, whose XML counterparts are not supported in OWL 2 (resp., OWL 2 profile), to the OWL 2 built-in datatype *rdfsLiteral*. This datatype denotes the union of all datatypes (i.e., top datatype). Note that a more intuitive approximation of unsupported datatypes (e.g., *xsddouble* to *xsddecimal*) is not possible since the value spaces of primitive datatypes are disjoint.¹⁰

4.2 Mapping Generation

Generation of Direct Mappings. BOOTOX follows the W3C direct mapping guidelines to generate mappings from a relational database to an ontological vocabulary. BOOTOX relies on the R2RML language to produce direct mappings, which are particular cases of the mappings that can be expressed in R2RML. Intuitively, an R2RML mapping allows to map any valid SQL query or view (i.e., logical table) into a target vocabulary.

The W3C direct mapping specification does not introduce specific restrictions on the used target ontological vocabulary, that is, it only requires to reference the ontological vocabulary via its URI. Hence, the generated mappings may reference the vocabulary of any given ontology. BOOTOX allows this behaviour and it can produce mappings for a vocabulary generated from the relational database (see Section 4.1) or for the vocabulary of an external domain ontology.

Generation of Mappings beyond Direct Ones. Besides direct mappings described above, that involve only projection operator in the SQL part, BOOTOX can help user to build *complex* R2RML mappings, that may also include selection and/or join operators. There are three general approaches that we use to find candidate mappings:

- (i) In the direct mapping approach, tables are mapped to classes. To generalise this, consider a table T and a table T' that has a foreign key constraint referencing T or that can be joined. By taking the left join of T and T' we obtain a subset of the tuples in T . If this subset of tuples is sufficiently big and sufficiently different from T itself, we obtain a subclass of the class T was mapped to. Furthermore, by letting this process to continue recursively, we obtain rich class hierarchies. Here, as in direct mappings, the many-to-many exception applies.
- (ii) Another way to discover subclasses in the database is to look for clusters of tuples in a table. To this end, we can consider a tuple to be a vector of numerical values (assigning distinct numbers to each value), and look for sets of tuples whose distance from each other is sufficiently small.
- (iii) Finally, one can also consider subsets of the attributes of a table, particularly when tables are not known to in standard normal forms, such as BCNF. In such tables, there will usually be tuples with repeating values in

¹⁰ <http://www.w3.org/TR/swbp-xsch-datatypes/>

attributes. Such sets of tuples can, again, be considered subclasses of the class the table itself was mapped to.

Note that the generation of these complex mappings heavily relies on interaction with the users since these techniques allow to generate only the SQL queries for mappings, while they do not offer the names of classes and properties defined by these queries; the names should be introduced by the users.

Generation of Provenance Mappings. BOOTOX automatically extends direct mappings with meta-information about provenance.¹¹ The provenance meta-information is modelled in the mapping assertion, adding, for instance, the source database from which the information is extracted, and more granular information, like table and column identifiers. Currently, provenance in BOOTOX comes into three different granularity levels, whose convenience will depend on the intended use of this information:

- (i) *URI level:* each generated URI is associated with provenance information. This is especially interesting when different database elements are merged into the same URI (e.g., see pattern (14) in Table 1).
- (ii) *Triple level:* triples are annotated with provenance via RDF reification (i.e., three new triples are added for each triple).
- (iii) *Graph level:* in this granularity, provenance is attached to RDF named graphs that group triples with similar provenance characteristics. For example, one could group triples generated by automatically generated mappings or triples derived from the same data source.

This meta-information can be later used for a wide range of purposes. For example it can be used to provide a richer query answering interface, to help in the debugging of a data access instance (i.e., identifying faulty ontology axioms or mappings), or to discard some sources based on a variety of criteria, e.g., licenses, access privileges, costs, etc.

4.3 Importing

It is increasingly often the case that a high quality ontology of (parts of) the domain already exists and captures the domain experts vocabulary better than the directly mapped ontology. When such an ontology is available, BOOTOX allows importing it to extend the automatically generated ontology.¹² To this end, BOOTOX integrates the ontology alignment system LogMap [18, 19], that aligns two ontologies \mathcal{O}_1 and \mathcal{O}_2 by deriving OWL 2 equivalence and sub-class(property) axioms between the entities from \mathcal{O}_1 's and \mathcal{O}_2 's vocabularies using the lexical characteristics of the terms and the structure of the ontologies. BOOTOX gives special care to avoid introducing unwanted consequences that may lead to unexpected answers. Thus, LogMap will discard alignment axioms that would lead to

¹¹ Based on the W3C recommendation PROV-O <http://www.w3.org/TR/prov-o/>.

¹² If the imported ontology is outside the desired target profile it should first be approximated using off-the-shelf semantic or syntactic approximation techniques (e.g., [11]).

inconsistencies, or faulty consequences like ‘*Well* SubClassOf: *WellBore*’. This is based on novel techniques to avoid violations of the so called consistency and conservativity principles (the interested reader please refer to [37]).

4.4 Conformance with Problem Requirements

BOOTOX covers the two bootstrapping tasks defined in Section 3 and fully meets the Metrics (1) and (2) regarding the ontology and mapping language.

Furthermore, as shown in Section 5, it provides reasonable results in practice with respect to Metrics (3) and (4). *Query coverage*, i.e., Metric (3), is enhanced thanks to the importing facility. Regarding *query results*, i.e., Metric (4), note that we supply the system with *provenance* capabilities, which are useful for analysis. In the case when query results contain unexpected answers, provenance will help the user to “trace” the source of these answers, thus helping to gain a better understanding of possible issues with the ontology or mappings.

5 BOOTOX at Work

BOOTOX has been tested with a number of databases and scenarios (e.g., virtual and materialised data access). In this section we present results with respect to the quality of the bootstrapped assets (i.e., vocabulary, ontology, and mappings). We have evaluated (i) the ability of formulating queries with the bootstrapped vocabulary in an industrial scenario (Section 5.1), and (ii) the ability of (enabling the) answering of queries with the bootstrapped ontology and mappings in a controlled scenario (Section 5.2).

We compared BOOTOX¹³ to three other bootstrapping systems: IncMap [30], MIRROR [26], and -ontop- [32]. IncMap is designed to directly map a relational database to a target ontology, but is focused on a semi-automatic, incremental/interactive approach rather than direct automated bootstrappings. Both -ontop- and MIRROR follow an approach that is similar to the one employed in BOOTOX, with respect to the generation of a semantic data access instance (i.e., vocabulary, ontology, and mappings).¹⁴

5.1 Query Coverage in the EU Optique Project

In this section we assess the quality of the bootstrapped ontology vocabulary to enable the formulation of queries. To this end we used the terms from the query catalogs available in the industrial scenario provided by the EU Optique project. We looked at how many terms from the catalogs were *covered* by the bootstrapped ontological vocabulary and then did manual verification of the quality of the coverage.

¹³ Note that we have evaluated BOOTOX with its automatic setting, that is, with the functionality to generate complex mappings turned off.

¹⁴ -ontop- generates only vocabulary, that is, an ontology containing only declaration axioms.

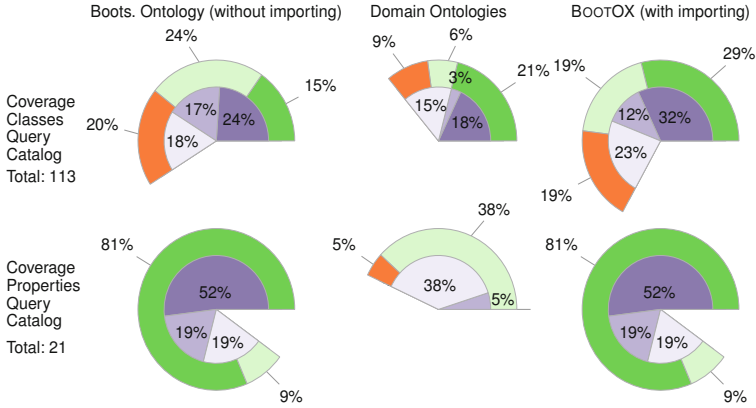
The figure displays three main components of the BOOTOX interface:

- Main Menu (Top Left):** A grid of icons for various bootstrapping and analysis tools: Logical Bootstrapping, Step-by-Step Bootstrapping, Import-Align Ontology, Import Layer Ontology, Approximate Ontology, Statistics Bootstrapping, Integrated Bootstrapping, Provenance Bootstrapping, Statistics Provenance, Bootstrapping for VQS, Extracted Annotations, and Coverage Query Catalog.
- Integrated Ontology and Mapping Bootstrapper (Top Right):** A form for configuring bootstrapping. It includes:
 - Available schemata:** A table with columns for Table name and Primary key. Example: `access_object` with primary key `access_object_s`.
 - Provenance mappings:** Checkboxes for At URI level and At triple level.
 - Extended features for the bootstrapped ontology:** Checkboxes for Extract annotations for QP1, Infer subclass relationships, and Merge properties with same name and range.
 - Import domain ontology:** A text input field with the URL `http://www.optique-project.eu/ontology/subsurface-exploration/Ont-20141125`.
 - OWL 2 profile:** A table with columns for OWL 2 DL, #Axioms, #Class, and #Object properties. Example: OWL 2 DL with 687 axioms and 97 classes.
 - OWL 2 QL approximation:** Checkboxes for Via ontology alignment and Via direct connection to the database.
 - Perform approximation of the bootstrapped ontology.
- Coverage of Query Catalogs (Bottom Left):** A section showing the selection of an ontology and a query catalog. It includes two pie charts:
 - Coverage of class terms:** A pie chart showing the distribution of class terms, with a legend for 'Not covered' and 'Covered (0.0, 1.0)'. The 'Covered' portion is further divided into 'Covered (0.0, 1.0)' and 'Covered (0.1, 1.0)'.
 - Coverage of property terms:** A similar pie chart for property terms.

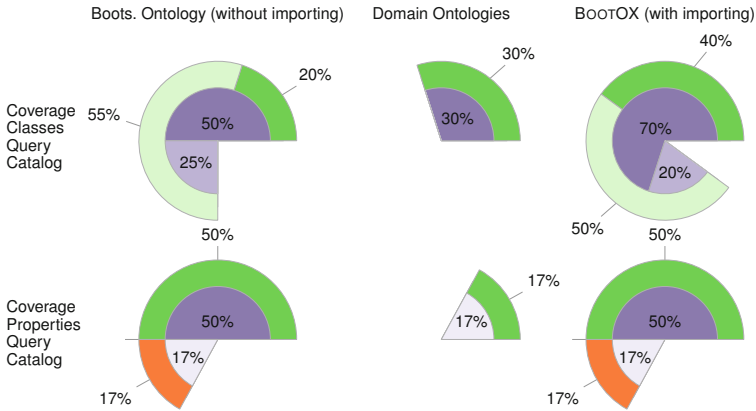
Fig. 2. BOOTOX interfaces in the Optique platform, from left to right, top to bottom: the main BOOTOX menu, the bootstrapping form, and the query catalog coverage

Industrial Scenario. The EU Optique project aims at facilitating scalable end-user access to big data in the oil and gas industry [14]. Optique advocates for an OBDA approach where the ontology provides a virtual access to the data and the mappings connect the ontology with the data source. The project is focused around two demanding use cases provided by Optique industry partners Siemens and Statoil. BOOTOX has been deployed in Siemens [20] and Statoil [22] as part of the “Ontology and mapping management module” provided by Optique’s platform [21]. Furthermore, BOOTOX has already shown to save time, in the creation of the initial OBDA assets, to the IT experts of both Statoil and Siemens. Note that the Optique scenario requires BOOTOX to bootstrap an OWL 2 QL as it is required by the query rewriting engine in order to rewrite the queries formulated over the ontology into queries on the database using reasoning [8].

BOOTOX in the Optique Platform. Figure 2 shows an overview of the BOOTOX related interfaces. The main menu presents to the user the available options currently supported in BOOTOX: automatic bootstrapper, guided bootstrapper, ontology alignment, provenance bootstrapping, bootstrapping related statistics, etc. The screenshot on the right shows the integrated bootstrapping form in BOOTOX where the user can select from a number of options including the database schema, imported ontology, provenance, OWL 2 QL approximation. The bottom screenshot shows the coverage of a selected query catalog by the vocabulary of a bootstrapped ontology.



(a) Statoil use case.



(b) Siemens use case.

Fig. 3. Coverage of terms from the Optique query catalog with terms from ontologies. Inner pie charts show coverage by lexical confidence using I-SUB [43]: ■ in [0.9, 1.0], ■ in [0.8, 0.9), □ in [0.6, 0.8). Outer pie charts represent the manual verification of the quality of the terms with a coverage above 0.6: ■ *true positive*, ■ *semi-true positive*, ■ *false positive*. Note that *semi-true positives* are not clear-cut cases where the ontology term has a broader or narrower meaning with respect to the query term

Coverage of Statoil Query Catalog. The query catalog at Statoil currently includes 60 queries that contain representative information needs from Statoil geologists. Most of the data needed by Statoil geologists is stored in the Exploration and Production Data Store (EPDS), Statoil’s corporate data store for exploration and production data and interpretations. The *NPD FactPages* [36] ontology is relevant to the domain of EPDS.

We bootstrapped ontological vocabulary from the relevant parts of EPDS using BOOTOX, MIRROR and -ontop-. The generated vocabulary, on average,

contained more than 3,000 classes, 3,500 object properties, and 42,000 datatype properties. Note that IncMap relies on the vocabulary of the available domain ontology. BOOTOX, unlike -ontop- and MIRROR, includes a built-in ontology alignment system which allows to import the vocabulary of the domain ontologies into the bootstrapped ontology.

The results of the query catalog coverage are summarised in Figure 3a. The first column represent the coverage of the bootstrapped ontologies computed by BOOTOX (without importing the domain ontologies), -ontop- and MIRROR. Since all three systems rely on the direct mapping directives, the bootstrapped vocabulary is, apart from minor differences, basically the same. The middle columns represent the coverage of the vocabulary of the domain ontology, which is equal to the coverage of IncMap. The third columns shows the coverage results achieved by the ontology bootstrapped by BOOTOX including importing. For example, 44% of the classes in the query catalog has a good lexical intersection (greater or equal 0.8) with terms of the ontology bootstrapped by BOOTOX; furthermore, 29% of the classes are fully covered (i.e., true positives).

Coverage of Siemens Query Catalog. The data in the Siemens use-case is stored in several databases with different schemata. Although the schemata are not specially large, the size of the data is in the order of hundreds of terabytes, e.g., there is about 15 GB of data associated to a single turbine, and it currently grows with the average rate of 30 GB per day [20]. As for the Statoil use case, we extracted the relevant terms of the query catalog, bootstrapped an ontology from one of the Siemens databases using BOOTOX, MIRROR and -ontop-. Additionally, BOOTOX performed alignment with two Siemens ontologies about diagnostic procedures and turbines. Finally, IncMap relied on the vocabulary of these ontologies. The results of the coverage are summarised in Figure 3b.

Quality Assessment. The experiments show that the bootstrapped ontologies without importing had a higher coverage than the domain ontologies in isolation, e.g., 39% of query class coverage against 27% in the Statoil use case. These results suggest that there is an adequate number of table and column names with potentially adequate semantic relations with the terms that domain experts at Statoil and Siemens have in mind when they access data, and thus, the ontology vocabulary computed by the ontology bootstrappers is indeed relevant to query formulation. Nevertheless, the domain ontologies naturally complement the vocabulary obtained from the database and hence BOOTOX is able to bootstrap and ontology with better coverage over the query catalog than the ones generated by -ontop- and MIRROR. For example, 48% of the classes (resp., 90%) in the Statoil (resp., Siemens) catalog are fully or partially covered in the bootstrapped ontology computed by BOOTOX.

Table 2. RODI results of all tests per scenario. Values in cells are the success scores

| Scenario | IncMap | MIRROR | -ontop- | BootOX |
|----------------------------|-------------|-------------|-------------|-------------|
| Adjusted naming | | | | |
| CMT | 0.5 | 0.28 | 0.39 | 0.39 |
| CONFERENCE | 0.26 | 0.27 | 0.37 | 0.37 |
| SIGKDD | 0.21 | 0.3 | 0.45 | 0.45 |
| Cleaned hierarchies | | | | |
| CMT | 0.44 | 0.17 | 0.28 | 0.28 |
| CONFERENCE | 0.16 | 0.23 | 0.3 | 0.3 |
| SIGKDD | 0.11 | 0.11 | 0.16 | 0.16 |
| Combined case | | | | |
| SIGKDD | 0.05 | 0.11 | 0.16 | 0.16 |
| Missing FKs | | | | |
| CONFERENCE | 0.03 | 0.17 | - | 0.17 |
| Denormalised | | | | |
| CMT | 0.22 | 0.22 | 0.28 | 0.28 |

5.2 Query Answering in a Controlled Scenario

In this section we assess the quality of the bootstrapped ontology and mappings to enable the answering of queries in a controlled scenario.¹⁵ To this end we ran experiments with a recently released relational-to-ontology benchmark suite, RODI [29], comparing BOOTOX to IncMap, -ontop- and MIRROR. RODI is designed to test relational-to-ontology mappings end-to-end: it provides an input database and a target ontology and requests complete mappings or mapped data to query. RODI is based on *scenarios*, with each scenario comprising several query tests. While RODI is extensible and can run scenarios in different application domains, it ships with a set of *default scenarios* in the conference domain that are designed to test a wide range of fundamental relational-to-ontology mapping challenges in a controlled fashion. The effectiveness of mappings is then judged by a score that mainly represents the number of query tests that return expected results on mapped data.

IncMap is designed to automatically map the target ontology directly to the input database, while BOOTOX approached this task in two steps: first, it bootstrapped an intermediate ontology and mappings from the database. Then, it aligned this intermediate, bootstrapped ontology to the target ontology as provided by the benchmark. As mentioned in Section 5.1, neither -ontop- nor MIRROR include a built-in ontology alignment system to support the importing of the target ontology provided by the benchmark. In order to be able to evaluate these systems with RODI, we aligned the generated ontologies by -ontop- and MIRROR with the target ontology using the LogMap system in a similar setup to the one used in BOOTOX.

Scenarios. RODI default scenarios are a selection of benchmark scenarios set in the conference domain, based on three different conference ontologies: CMT,

¹⁵ Note that assessing the quality of the bootstrapped ontology and mappings in an open scenario like Optique requires a huge involvement of domain experts, thus we leave it for future work.

CONFERENCE, and SIGKDD¹⁶. For each ontology, the benchmark provides a set of database instances that are to be mapped to the ontology vocabulary. While all databases are modelled to contain the same data that the ontologies might contain, they deviate largely in how close they are to their corresponding ontologies in terms of structure and modelling patterns. For each ontology, there is a number of mapping challenges that can be tested: (i) *Adjusted naming*. Here the identifier names in the databases are syntactically changed in comparison with the names in the ontology. (ii) *Cleaned hierarchies*. In this scenarios, the databases are remodelled from ground to follow widely used relational database design patterns. Most significantly, this includes cases where abstract parent classes have no corresponding table in the database, several sibling classes are jointly represented in a single table, etc. (iii) *Combined case* mixes changes made in scenarios with adjusted naming and cleaned hierarchies. (iv) *Missing FKs* represents the case of a database with no explicit referential constraints at all. (v) In the *denormalised* case, the database contains a few denormalised tables.

Results. Results show that the BOOTOX comes out in top position for seven out of nine tested scenarios (see Table 2). This shows that BOOTOX is well suited for meeting the requirements of an end-to-end scenario. Note that the results between BOOTOX and -ontop- may look very similar; however, currently RODI only provides the percentage of correctly answered queries (i.e., BOOTOX and -ontop- fail in the same queries). RODI is being extended to include a more fine grained evaluation in terms of Precision and Recall in order to take into account partially answered queries, which may reveal more differences among the evaluated systems. On a more generic note, however, results also demonstrate that none of the tested systems to date manages to solve relational-to-ontology mapping challenges with a score above 0.5. This confirms the need for specialised relational-to-ontology bootstrapping systems such as BOOTOX, which build the foundation for better solutions.

Provenance. When we evaluated BOOTOX with RODI without the use of provenance mappings, the results were slightly worse than the ones in Table 2: in three scenarios we detected unexpected answers. Then we made use of the provenance functionality of BOOTOX to analyse the source of these unexpected answers. As an outcome of such an analysis we identified and fixed the faulty mappings, hence improving the results for those scenarios.

6 Related Work

The implementation of BOOTOX has been motivated by the fact that existing ontology and mapping bootstrappers provide limited or no support for the bootstrapping tasks and quality requirements described in Section 3 (see, e.g., [34,41]). Most of the state of the art bootstrappers fail to conform with

¹⁶ <http://oei.ontologymatching.org/2015/conference/>

the ontology and mapping language standards, or they do not provide profiling capabilities for the output ontology. Moreover, to the best of our knowledge they do not provide bootstrapping of complex mappings. For historical reasons (i.e., OWL was not yet defined), former systems used RDFS and F-Logic axioms (e.g., [5,44]). Other systems have also used DLR-Lite based languages (e.g., [25]) and extensions based on SWRL (e.g., [13,24]). Regarding mapping generation, before R2RML became a W3C recommendation, system typically relied on their own native language to define mappings (e.g., D2RQ [7], Mastro [10]). To the best of our knowledge, currently only IncMap [30], MIRROR [26], -ontop- [32], and Ultrawrap [33,35] produce mappings in the R2RML language.

Among the systems using OWL or OWL 2 as the ontology language, only BOOTOX put special attention to the target ontology expressiveness. BOOTOX allows to output different ontology axioms to conform to the required OWL 2 profile as discussed in Section 4.1. Many bootstrapping systems typically use exact or min cardinality restrictions which fall outside the three OWL 2 profiles (e.g., [1,46]). Furthermore, other systems, like [6], produce an ontology that falls into OWL 2 Full due to the use of the *InverseFunctional* characteristic in both data and object properties. Finally, MIRROR, -ontop-, and Ultrawrap are conformant to the OWL 2 QL, but they do not support profiling to the other sublanguages of OWL 2.

As BOOTOX, systems like Automapper [13], Relational.OWL [23] and ROSEX [12] complement the automatically generated ontology with links to domain ontologies. However, none of these systems apply logic-based techniques to assess the consequences of such links to domain ontologies.

Special mention require the approaches in [9,36]. These approaches use (semi-automatic) ontology learning techniques to exploit the data and discover interesting patterns that can be included to enrich the ontology. Currently, BOOTOX only relies on a (fully-automatic) schema-driven generation of the ontology and mappings.

In the literature one can also find several approaches to overcome the OWA problem when dealing with data-centric applications (e.g., [27,45]). These approaches typically extend the semantics of OWL 2. The integration of these approaches in a bootstrapping scenario is, however, still an open problem.

7 Conclusions and Future Work

We presented BOOTOX, an automatic ontology and mapping bootstrapper. To the best of our knowledge, BOOTOX is the only bootstrapper that *(i)* profiles the concrete language of the output OWL ontology, *(ii)* puts special attention to the datatype conversion, *(iii)* enhances the direct mappings with provenance meta-information, *(iv)* bootstraps a range of complex mappings, and *(v)* includes a built-in logic-based ontology alignment system. Furthermore, we tested BOOTOX in a number of databases and test cases involving materialised or virtual data access. The evaluation suggests that automatic techniques to bootstrap an initial ontology and mappings are suitable to be used in practice.

We see bootstrapping as the first step towards the creation of a fully-fledged semantic data access system. Bootstrapped assets are by no means perfect, and thus they should be post-processed, validated, and extended. Our evaluation in both industrial (i.e., Optique) and research contexts has also served to guide the extension of BOOTOX with semi-automatic techniques. For example, while working with Statoil's EPDS database, we found that the discovery of implicit constraints represents a critical feature since EPDS has very few constraints, e.g., many tables are materialised views without specified primary or foreign keys. In the close future we aim at extending our evaluation to understand the limits of semi-automatic ontology and mapping bootstrappers to enable the answering of the queries in open scenarios like Optique.

Acknowledgments. This work was partially funded by the EU project Optique (FP7-ICT-318338), the Research Council of Norway through the project DOIL (RCN project n. 213115), and the EPSRC projects MaSI³, Score!, and DBOnto.

References

1. Alalwan, N., et al.: Generating OWL ontology for database integration. In: SEMAPRO (2009)
2. Arenas, M., et al.: Enabling faceted search over OWL 2 with SemFacet. In: OWLED (2014)
3. Arenas, M., et al.: Faceted search over ontology-enhanced RDF data. In: CIKM (2014)
4. Arenas, M., et al.: SemFacet: semantic faceted search over yago. In: WWW (2014)
5. Astrova, I.: Reverse engineering of relational databases to ontologies. In: Bussler, C.J., Davies, J., Fensel, D., Studer, R. (eds.) ESWS 2004. LNCS, vol. 3053, pp. 327–341. Springer, Heidelberg (2004)
6. Astrova, I.: Rules for mapping SQL relational databases to OWL ontologies. In: MTSR (2007)
7. Bizer, C., et al.: D2RQ-treating non-RDF databases as virtual RDF graphs. In: ISWC Posters (2004)
8. Calvanese, D., et al.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. English. JAR **39**(3) (2007)
9. Cerbah, F., et al.: Perspectives in ontology learning. In: Ontology Learning from Databases: Some Efficient Methods to Discover Semantic Patterns in Data. AKA / IOS Press. Serie (2012)
10. Civili, C., et al.: Mastro studio: managing ontology-based data access applications. In: PVLDB, vol. 6, no. 12 (2013)
11. Console, M., Mora, J., Rosati, R., Santarelli, V., Savo, D.F.: Effective computation of maximal sound approximations of description logic ontologies. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part II. LNCS, vol. 8797, pp. 164–179. Springer, Heidelberg (2014)
12. Curino, C., Orsi, G., Panigati, E., Tanca, L.: Accessing and documenting relational databases through OWL ontologies. In: Andreassen, T., Yager, R.R., Bulskov, H., Christiansen, H., Larsen, H.L. (eds.) FQAS 2009. LNCS, vol. 5822, pp. 431–442. Springer, Heidelberg (2009)

13. Fisher, M., et al.: Use of OWL and SWRL for semantic relational database translation. In: OWLED (2008)
14. Giese, M., et al.: Optique — Zooming In on Big Data Access. *Computer* **48**(3) (2015)
15. Grau, B.C., et al.: Querying life science ontologies with SemFacet. In: SWAT4LS (2014)
16. He, B., et al.: Accessing the Deep Web. *Commun. ACM* **50**(5) (2007)
17. Horridge, M., et al.: The manchester OWL syntax. In: OWLED (2006)
18. Jiménez-Ruiz, E., et al.: Large-scale interactive ontology matching: algorithms and implementation. In: ECAI (2012)
19. 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)
20. Kharlamov, E., Solomakhina, N., Özçep, Ö.L., Zheleznyakov, D., Hubauer, T., Lamparter, S., Roshchin, M., Soyly, A., Watson, S.: How semantic technologies can enhance data access at siemens energy. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 601–619. Springer, Heidelberg (2014)
21. Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., Bilidas, D., Giese, M., Haase, P., Horrocks, I., Kllapi, H., Koubarakis, M., Özçep, Ö., Rodríguez-Muro, M., Rosati, R., Schmidt, M., Schlatte, R., Soyly, A., Waaler, A.: Optique: towards OBDA systems for industry. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 125–140. Springer, Heidelberg (2013)
22. Kharlamov, E., et al.: Ontology based access to exploration data at statoil. In: ISWC (2015)
23. de Laborda, C.P., et al.: Database to semantic web mapping using RDF query languages. In: ER (2006)
24. Levshin, D.V.: Mapping Relational Databases to the SemanticWeb with Original Meaning. *Int. J. Software and Informatics* **4**(1) (2010)
25. Lubyte, L., Tessaris, S.: Automatic extraction of ontologies wrapping relational data sources. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 128–142. Springer, Heidelberg (2009)
26. de Medeiros, L.F., Priyatna, F., Corcho, O.: MIRROR: automatic R2RML mapping generation from relational databases. In: Cimiano, P., Frasincar, F., Houben, G.-J., Schwabe, D. (eds.) ICWE 2015. LNCS, vol. 9114, pp. 326–343. Springer, Heidelberg (2015)
27. Motik, B., et al.: Bridging the gap between OWL and relational databases. In: WWW (2007)
28. Motik, B., et al.: Parallel materialisation of datalog programs in centralised, main-memory RDF systems. In: AAAI (2014)
29. Pinkel, C., Binnig, C., Jiménez-Ruiz, E., May, W., Ritze, D., Skjæveland, M.G., Solimando, A., Kharlamov, E.: RODI: a benchmark for automatic mapping generation in relational-to-ontology data integration. In: Gandon, F., Sabou, M., Sack, H., d’Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 21–37. Springer, Heidelberg (2015)
30. Pinkel, C., et al.: IncMap: pay as you go matching of relational schemata to OWL ontologies. In: OM (2013)

31. Poggi, A., et al.: Linking Data to Ontologies. *J. Data Semantics* **X** (2008)
32. Rodriguez-Muro, M., et al.: Efficient SPARQL-to-SQL with R2RML Mappings. *J. Web Sem.* (to appear, 2015)
33. Sequeda, J., et al.: On directly mapping relational databases to RDF and OWL. In: *WWW* (2012)
34. Sequeda, J., et al.: Survey of directly mapping SQL databases to the semantic web. *Knowledge Eng. Review* **26**(4) (2011)
35. Sequeda, J., et al.: Ultrawrap: SPARQL Execution on Relational Data. *J. Web Sem.* **22** (2013)
36. Skjæveland, M.G., et al.: Engineering Ontology-Based Access to Real-World Data Sources. *J. Web Sem.* (to appear, 2015)
37. Solimando, A., Jiménez-Ruiz, E., Guerrini, G.: Detecting and correcting conservativity principle violations in ontology-to-ontology mappings. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) *ISWC 2014, Part II*. LNCS, vol. 8797, pp. 1–16. Springer, Heidelberg (2014)
38. Soyly, A., et al.: Experiencing optiqueVQS: a multi-paradigm and ontology-based visual query system for end users. In: *Univ. Access in the Inform. Society* (2015)
39. Soyly, A., et al.: OptiqueVQS: towards an ontology-based visual query system for big data. In: *MEDES* (2013)
40. Soyly, A., Giese, M., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: Towards exploiting query history for adaptive ontology-based visual query formulation. In: Closs, S., Studer, R., Garoufallou, E., Sicilia, M.-A. (eds.) *MTSR 2014*. CCIS, vol. 478, pp. 107–119. Springer, Heidelberg (2014)
41. Spanos, D.-E., et al.: Bringing Relational Databases into the Semantic Web: A Survey. *Semantic Web* **3**(2) (2012)
42. Stefanoni, G., et al.: Answering conjunctive queries over EL knowledge bases with transitive and reflexive roles. In: *AAAI* (2015)
43. Stoilos, G., Stamou, G., Kollias, S.D.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) *ISWC 2005*. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
44. Stojanovic, L., et al.: Migrating data-intensive web sites into the semantic web. In: *SAC* (2002)
45. Tao, J., et al.: Integrity constraints in OWL. In: *AAAI* (2010)
46. Tirmizi, S.H., Sequeda, J., Miranker, D.P.: Translating SQL applications to the semantic web. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) *DEXA 2008*. LNCS, vol. 5181, pp. 450–464. Springer, Heidelberg (2008)
47. Zhou, Y., et al.: Pay-as-you-go OWL query answering using a triple store. In: *AAAI* (2014)

Assessing and Refining Mappings to RDF to Improve Dataset Quality

Anastasia Dimou¹(✉), Dimitris Kontokostas², Markus Freudenberg²,
Ruben Verborgh¹, Jens Lehmann², Erik Mannens¹, Sebastian Hellmann²,
and Rik Van de Walle¹

¹ iMinds - Multimedia Lab, Ghent University, Ghent, Belgium
{[anastasia.dimou](mailto:anastasia.dimou@ugent.be),[ruben.verborgh](mailto:ruben.verborgh@ugent.be),[erik.mannens](mailto:erik.mannens@ugent.be),[rik.vandewalle](mailto:rik.vandewalle@ugent.be)}@ugent.be
² Institut Fur Informatik, AKSW, Universitat Leipzig, Leipzig, Germany
{[kontokostas](mailto:kontokostas@informatik.uni-leipzig.de),[freudenberg](mailto:freudenberg@informatik.uni-leipzig.de),[lehmann](mailto:lehmann@informatik.uni-leipzig.de),[hellmann](mailto:hellmann@informatik.uni-leipzig.de)}@informatik.uni-leipzig.de

Abstract. RDF dataset quality assessment is currently performed primarily after data is published. However, there is neither a systematic way to incorporate its results into the dataset nor the assessment into the publishing workflow. Adjustments are manually –but rarely– applied. Nevertheless, the root of the violations which often derive from the mappings that specify how the RDF dataset will be generated, is not identified. We suggest an *incremental, iterative and uniform validation workflow* for RDF datasets stemming originally from (semi-)structured data (e.g., CSV, XML, JSON). In this work, we focus on assessing and improving their mappings. We incorporate (i) a *test-driven approach for assessing the mappings* instead of the RDF dataset itself, as mappings reflect how the dataset will be formed when generated; and (ii) perform *semi-automatic mapping refinements* based on the results of the quality assessment. The proposed workflow is applied to diverse cases, e.g., large, crowdsourced datasets such as DBpedia, or newly generated, such as iLastic. Our evaluation indicates the efficiency of our workflow, as it significantly improves the overall quality of an RDF dataset in the observed cases.

Keywords: Linked data mapping · Data quality · RML · R2RML · RDFUnit

1 Introduction

The Linked Open Data (LOD) cloud¹ consisted of 12 datasets in 2007, grew to almost 300 in 2011², and, by the end of 2014, counted up to 1,000³. Although more and more data is published as Linked Data (LD), the datasets' quality and consistency varies significantly, ranging from expensively curated to relatively low quality datasets [29]. In previous work [21], we observed that similar violations can occur very frequently. Especially when datasets originally stem from

¹ <http://lod-cloud.net/>

² <http://lod-cloud.net/state>

³ <http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

semi-structured formats (csv, XML, etc.) and their RDF representation is obtained by repetitively applying certain mappings, the violations are often repeated, as well. semantically annotating data to acquire their enriched representation using the RDF data model. A mapping consists of one or more *mapping definitions* (MDS) that state how RDF terms should be generated, taking into account a data fragment from an original data source, and how these terms are associated to each other and form RDF triples.

The most frequent violations are related to the dataset’s schema, namely the vocabularies or ontologies used to annotate the original data [21]. In the case of (semi-)structured data, the dataset’s schema derives from the set of classes and properties specified within the mappings. A mapping might use a single ontology or vocabulary to annotate the data, or a proprietary vocabulary can be generated as the data is annotated. Lately, combinations of different ontologies and vocabularies are often used to annotate data [28], which increases the likelihood of such violations. A violation might derive from (i) *incorrect usage of schemas* in the MDS; and (ii) mistakes in the *original data source*. The second category of violations can be resolved by cleansing the data. In this work, we focus specifically on the first, which is directly related to the mapping process.

Only recently, several research efforts started focusing on formalising Linked Data quality tracking and assessment [29]. Nevertheless, such formalisation approaches remain independent of the Linked Data mapping and publishing process –quality assessment is not even mentioned in the best practices for publishing Linked Data [18]. Existing quality assessment refers to already published data and is, in most cases, performed by third parties rather than data publishers. Thus, incorporating quality assessment results corresponds to incorporating a *Linked Data feedback loop*: existing Linked Data infrastructures still neither intuitively process end-users’ input, nor properly propagate the modifications to the mapping definitions and original data. Consequently, the results are rarely and, if so, manually used to adjust the dataset, with the risk of being overwritten when a new version of the original data is published.

In this paper, we therefore propose a methodology that extends Linked Data quality assessment from data *consumption* to also cover data *publication*. We transform the assessment process normally applied to the final dataset so that it applies to the mappings as well. This allows publishers to discover mistakes in mapped RDF data –before they are even generated. Our methodology (i) augments the mapping and publishing workflow of (semi-)structured source formats with *systematic Linked Data quality assessments* for both the mappings and the resulting dataset; and (ii) *automatically suggests mapping refinements* based on the results of these quality assessments. We consider iterative, uniform, and gradual test-driven quality assessments to improve the dataset’s overall quality.

The paper is organized as follows: Section 2 details the need of quality assessment during the mapping process, followed by the introduction of a mapping workflow with quality assessment in Section 3. Next, Section 4 explains how quality assessments are applied to mappings, the results of which are used to refine mapping definitions in Section 5. Section 6 highlights different cases where

the proposed workflow was used, followed by an evaluation in Section 7. Finally, Section 8 and Section 9 summarize related solutions and conclusions.

2 Incorporating Quality in Mapping and Publishing

Data quality is commonly conceived as “fitness for use” for a certain application or use case [19]. A *data quality assessment metric, measure, or indicator* is a procedure for measuring a data quality dimension [3]. A *data quality assessment methodology* is defined as the process of evaluating whether a piece of data meets the information that consumers need in a specific use case [3]. In this respect, our use case is focused on the quality of the generated RDF dataset compared to the ontologies and vocabulary definitions of its schema. The uppermost goal is aiding data publishers to finally acquire valid and high quality Linked Data by annotating (semi-)structured data. We focus on the *intrinsic* dimension of data quality [29].

The earlier dataset quality is assessed, the better: we argue that mapping and publishing data can be considered software engineering tasks, and the cost of fixing a bug rises exponentially when a task progresses [4]. In software development, a common way to validate correct behaviour of a function is to accompany it by a set of unit tests. Similarly, a data mapping function can be accompanied by a set of test cases assigned to the mappings to ensure the correct generation of RDF datasets from input data. In this respect, incorporating quality assessment as part of the mapping and publishing workflow becomes essential, especially taking into account that it prevents the same violations to appear repeatedly within the dataset and over distinct entities. After all, in the mapping phase, structural adjustments can still be applied easily, since it allows us to pinpoint the origin of the violation, reducing the effort required to act upon quality assessment results.

Our approach has two main pillars: (i) *uniform quality assessment* of mapping definitions and the resulting dataset, as their quality is closely related; and (ii) *mapping definition refinements* to automatically improve mappings when problems are detected at the quality assessment.

Uniform Quality Assessment. Instead of assessing an RDF dataset for its schema quality, we apply the quality assessment to the mapping definitions directly, *before* they are used to generate the RDF dataset. Their assessment results are correlated, since MDs specify how the dataset will be formed. For example, violations of the range of a certain property can be assessed by inspecting the corresponding MD, which defines how triples with this property are generated. Even though quality assessment of MDs can cover many violations related to vocabularies and ontologies used to annotate the data, some schema-related violations depend on how the MDs are *instantiated* on the original data. For example, a violation occurs if an object of integer datatype is instantiated with a floating-point value from the original source. Therefore, a *uniform* way of incrementally assessing the quality of the RDF dataset and the mapping definitions should cover both the mappings and the dataset.

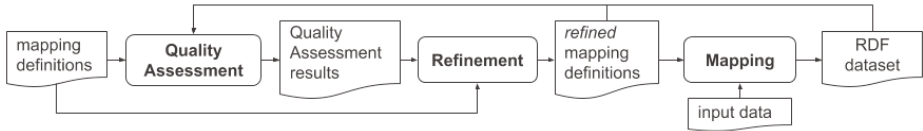


Fig. 1. Quality Assessment enabled Linked Data mapping and publishing workflow

Mapping Definition Refinements. If violations are only corrected in the resulting dataset, they will have to be corrected every time a new version of the dataset is generated. Also, when a violation is found, it is not straightforward to discover its cause, as the connection with the MDs and the source data is not apparent. A more effective approach is to refine the MDs that generate those triples, so the violation cannot occur in future versions. Furthermore, if the violation is associated with a MD, it can be addressed directly on the place where it occurred, and instead of having to regenerate the entire dataset, only the triples affected by the refinement need to be regenerated to correct the violation.

3 Linked Data and Mappings Assessment and Refinement

Uniform quality assessment requires that, in addition to the generated dataset, the mapping definitions themselves are also RDF triples. This way, the same RDF-based techniques can be applied. Additionally, performing (automated) refinement of MDs requires that machines can process and update them. Such direct processing of MDs is difficult if mappings are tightly coupled to the implementation, as is the case with most existing mapping solutions. In this respect, we focus on RDF-based mapping languages for stating the MDs. Below, we describe such a workflow (Section 3.1) and a solution that materializes it (Section 3.2).

3.1 Linked Data and Mappings Assessment and Refinement Workflow

We propose a *uniform, iterative, incremental assessment and refinement workflow* (Fig. 1) that produces, at the end, a high-quality RDF dataset. Its steps:

1. The schema, as stated in the MDs, is assessed against different quality assessment measures, as it would have been done if it was the actual dataset.
2. The Quality Assessment report lists each violation identified.
3. The Mapping Quality Assessment (MQA) results are used to refine the MDs. The MQA can be repeated until a set of MDs without violations is generated or if the MDs cannot be further refined.
4. A refined version of the MDs is generated and used to execute the mapping of data –or a sample of the data.
5. The generated RDF output is assessed, using the same quality assessment framework. The Dataset –and optionally the Mapping– Quality Assessment (DQA) can be repeated until an ameliorated set of MDs is generated.

```

1 <#Mapping> rml:logicalSource <#InputX> ;
2   rr:subjectMap [ rr:template "http://ex.com/{ID}"; rr:class foaf:Person ];
3   rr:predicateObjectMap [ rr:predicate foaf:knows;
4     rr:objectMap [ rr:parentTriplesMap <#Acquaintance> ] .
5 <#Acquaintance> rml:logicalSource <#InputY> ;
6   rr:subjectMap [ rml:reference "acquaintance"; rr:termType rr:IRI; rr:class ex:Person ]
  ] .

```

Listing 1. RML mapping definitions

- When the MDs are finalized, the actual mapping is performed and the RDF dataset is generated exempt of violations to the greatest possible extent.

3.2 Quality Assessment and Refinement with [R2]RML and RDFUnit

We provide a solution that implements the aforementioned workflow. The two main components of our solution are: the RML (Section 3.2) that uses mapping definitions expressed in RDF, a prerequisite for uniform quality assessment and automated refinements, as we discussed above, and the RDFUnit validation framework (Section 3.2) due to its associated test-case-based architecture [20]. A proof-of-concept implementation relies on the RMLvalidator which can be found at <https://github.com/RMLio/RML-Validator.git>.

RML. R2RML [6] is the only W3C standardised mapping language for defining mappings of data in relational databases to the RDF data model. Its extension RML [10] broadens its scope and covers also mappings from sources in different (semi-)structured formats, such as CSV, XML, and JSON. RML documents [10] contain rules defining how the input data will be represented in RDF. The main building blocks of RML documents are Triples Maps (Listing 1: line 1). A Triples Map defines how triples of the form (subject, predicate, object) will be generated.

A Triples Map consists of three main parts: the Logical Source, the Subject Map and zero or more Predicate-Object Maps. The Subject Map (line 2, 6) defines how unique identifiers (URIs) are generated for the mapped resources and is used as the subject of all RDF triples generated from this Triples Map. A Predicate-Object Map (line 3) consists of Predicate Maps, which define the rule that generates the triple’s predicate (line 3) and Object Maps or Referencing Object Maps (line 4), which define how the triple’s object is generated. The Subject Map, the Predicate Map and the Object Map are Term Maps, namely rules that generate an RDF term (an IRI, a blank node or a literal). A Term Map can be a *constant-valued term map* (line 3) that always generates the same RDF term, or a *reference-valued term map* (line 6) that is the data value of a referenced data fragment in a given Logical Source, or a *template-valued term map* (line 2) that is a valid string template that can contain referenced data fragments of a given Logical Source.

RDFUnit [21] is an RDF validation framework inspired by test-driven software development. In software, every function should be accompanied by a set of unit tests to ensure the correct behaviour of that function through time. Similarly, in RDFUnit, every vocabulary, ontology, dataset or application can be associated by a set of data quality test cases. Assigning test cases in ontologies results in tests that can be reused by datasets sharing the same ontology. This fits well to the mapping and publishing pipeline as we focus on the [R2]RML ontologies.

The test case definition language of RDFUnit is SPARQL, which is convenient to directly query for identifying violations. For rapid test case instantiation, a pattern-based SPARQL-template engine is supported where the user can easily bind variables into patterns. An initial library of 17 common patterns was developed in [21, Table 1] which is now further extended⁴. RDFUnit has a *Test Auto Generator* (TAG) component. TAG searches for schema information and automatically instantiates new test cases. Schema can be in the form of RDFS or OWL axioms that RDFUnit translates into SPARQL under Closed World Assumption (CWA) and Unique Name Assumption (UNA). These TCs cover validation against: domain, range, class and property disjointness, (qualified) cardinality, (inverse) functionality, (a)symmetry, irreflexivity and deprecation. Other schema languages such as *IBM Resource Shapes*⁵ or *Description Set Profiles*⁶ are also supported. RDFUnit includes support for automatic schema enrichment via DL-Learner [23] machine learning algorithms. RDFUnit can check an RDF dataset against multiple schemas but when this occurs, RDFUnit does not perform any reasoning/action to detect inconsistencies between the different schemas.

4 [R2]RML Mapping Definitions Quality Assessment

It is straightforward to process [R2]RML mapping definitions as datasets, because they have a native RDF representation and are written from the viewpoint of the generated triples. Our assessment process targets both (i) consistency validation of the mapping definitions against the R2RML and RML schema and, mainly, (ii) consistency validation and quality assessment of the dataset to be generated against the schema defined in the mapping definitions. This first point is handled directly by RDFUnit; the second point is handled by emulating the resulting RDF dataset to assess its schema conformance.

Consistency Validation of the Mapping Definitions. The validation of mapping definitions against the [R2]RML schema is directly handled by RDFUnit extending the supported OWL axioms. New RDFUnit TAGs were defined to support all OWL axioms in [R2]RML ontology, e.g., each *Triples Map* should have exactly one *Subject Map*, producing a total of 78 automatically generated test cases.

⁴ <https://github.com/AKSW/RDFUnit/blob/master/configuration/patterns.ttl>

⁵ <http://www.w3.org/Submission/2014/SUBM-shapes-20140211/>

⁶ <http://dublincore.org/documents/dc-dsp/>

Consistency Validation and Quality Assessment of the Dataset as Projected by Its Mapping Definitions. In order to assess a dataset based only on the mapping definitions that state how it is generated, we considered the same set of schema validation patterns normally applied on the RDF dataset (cf. Section 5). Nevertheless, instead of validating the predicate against the subject and object, we extract the predicate from the **Predicate Map** and validate it against the **Term Maps** that define how the subject and object will be formed. For instance, the extracted predicate expects a **Literal** as object, but the **Term Map** that generates the object can be a **Referencing Object Map** that generates resources instead.

To achieve this, the properties and classes in the MDS are identified and their namespaces are used to retrieve the schemas and generate the test cases as if they were the actual dataset. We extended the corresponding RDFUnit test cases to apply to the MDS, adjusting the assessment queries.⁷ For instance, the **WHERE** clause of the SPARQL test case that assesses a missing language is:

```
1 ?resource ?P1 ?c .
2 FILTER (lang(?c) = )
```

In order to detect the same violation directly from a mapping definition, the **WHERE** clause of the assessment query is adjusted as follows:

```
1 ?poMap rr:predicate ?P1 ;
2 rr:objectMap ?resource .
3 ?P1 rdfs:range rdf:langString .
4 FILTER NOT EXISTS {?resource rr:language ?lang}
```

The validation is *Predicate-Map-driven* in principle. The expected value (line 3), as derived from the **Predicate Map**, is compared to the defined one (line 4), as derived from the corresponding **Object Map**. The next example is an RDFUnit SPARQL test case for assessing if the **rdfs:type** of a triple's **ObjectMap** conforms to the **rdfs:range** definition of an object property. Applying this test case to the aforementioned MD (cf. Listing 1), a violation is registered, as **foaf:knows** has **foaf:Person** and not **ex:Person** as range – assuming the ontology does not define **ex:Person** as equivalent or subclass of **foaf:Person**.

```
1 SELECT DISTINCT ?resource WHERE {
2   ?mappingTo rr:subjectMap ?resource .
3   { ?resource rr:class ?T1 . } UNION {
4     ?mapping rr:predicateObjectMap ?classPoMap .
5     ?classPoMap rr:predicate rdf:type ;
6               rr:objectMap/rr:constant ?T1 . }
7   ?mappingFrom rr:predicateObjectMap ?poMap .
8   ?poMap rr:predicate/rdfs:range ?T2 ;
9         rr:objectMap ?objM .
10  ?objM rr:parentTriplesMap ?mappingTo .
11  FILTER NOT EXISTS {
12    ?T2 (rdfs:subClassOf|(owl:equivalentClass|^owl:equivalentClass))* ?T1.}}
```

In order for our assessment to be complete, the defined test cases cover all possible alternative ways of defining equivalent MDs that generate the same triples. For instance, the default way to generate the type for a resource is through the

⁷ <https://github.com/AKSW/RDFUnit/blob/master/data/tests/Manual/www.w3.org/ns/r2rml/rr.tests.Manual.ttl>

rr:class property in the Subject Map (e.g., line 2 of Listing 1). However, one may also define the type via a Predicate Object Map having `rdf:type` in its Predicate Map.

RDFUnit can annotate test cases by requesting additional variables and binding them to specific result properties. Using the example of Listing 4 we map, for instance, variable `?T1` as `spin:violationValue` and variable `?T2` as the expected class. When a violation is identified, the annotations are applied and a result like the following is registered:

```

1 <5b7a80b8> a rut:ExtendedTestCaseResult;
2   rut:testCase rutt:rr-produces-range-errors ;
3   # (...) Further result annotations
4   spin:violationRoot ex:objectMapX ;
5   spin:violationPath rr:class ;
6   spin:violationValue ex:Person ;
7   rut:missingValue foaf:Person ;
8   ex:erroneousPredicate foaf:knows ;

```

However, some of the test cases normally applied to a dataset rely on the final values or refer to the complete dataset and thus, can only be validated after the mapping is performed –detected at *data-level* quality assessment (DQA). Such examples are (qualified) cardinality, (inverse) functionality, (a)symmetricity and irreflexivity. For example, we cannot validate an inverse functional property such as `foaf:homepage` without the actual values. Invalid mappings can occur as the mapping definitions are instantiated based on the input source, even though the mapping definitions appear to be valid. For instance, if the input data returns a value like “American”, instead of “http://dbpedia.org/resource/United_States”, it would result in generating the URI `<American>`, which is invalid.

5 [R2]RML Refinements Based on Quality Assessment

The results of Mapping Quality Assessment (MQA) can be used to suggest modifications or even automatically refine mapping definitions. The RDFUnit ontology provides multiple result representations in different formats [20], including RDF-based serialisations (`rut:ExtendedTestCaseResult` result type). Therefore, its results are easily processed by an agent that can automatically add and delete triples or suggest actions to the data publisher. In Section 5, we outline all examined violation patterns and indicate which **Term Map** should be refined and how. The suggested refinements are the minimum required actions to be taken to refine the mapping definitions, e.g., turn an **Object Map** into generated resources instead of literals, and serve as indicative *proof-of-concept* of the automation’s feasibility.

Mapping Refinements. Dealing with *range-level* violations requires different actions, depending on the value of the **Object Map** or **Referencing Object Map**. The **Predicate Map** is used to retrieve the property and identify its range, which is then compared to the corresponding **Object Map** or **Referencing Object Map**.

If the **Predicate Map** contains an *object property*, for instance, but the object is generated by a **Referencing Object Map**, which generates resources with type

Table 1. Violations detected by assessing the mapping definitions. The first column describes the type of violation, the second its level (Warning or Error). The third specifies the expected RDF term according to the ontology or schema, while the fourth the term map defining how the RDF term is generated. The last specifies the refinement.

| OWL axiom – Violation type | Level | Expect | Define | Automatic refinement |
|--|-------|--------|-------------|--|
| class disjointness | E | SbjMap | SbjMap | – |
| property disjointness | E | PreMap | PreMap | – |
| <code>rdfs:range</code> – class type | E | PreMap | (Ref)ObjMap | DEL: ObjMap ADD: PreMap domain to RefObjMap |
| <code>rdfs:range</code> – IRI instead of literal | E | PreMap | (Ref)ObjMap | DEL: (Ref)ObjMap ADD: ObjMap with literal termType |
| <code>rdfs:range</code> – literal instead of IRI | E | PreMap | ObjMap | DEL: ObjMap ADD: (Ref)ObjMap or ADD: ObjMap with IRI termType |
| <code>rdfs:range</code> – missing datatype | E | PreMap | (Ref)ObjMap | DEL: ObjMap ADD: ObjMap with PreMap datatype |
| <code>rdfs:range</code> – incorrect datatype | E | PreMap | (Ref)ObjMap | DEL: (Ref)ObjMap ADD: ObjMap with PreMap datatype |
| missing language | E | ObjMap | ObjMap | – |
| <code>rdfs:domain</code> | E | PreMap | SbjMap | ADD: PreMap domain to SbjMap |
| missing <code>rdf:type</code> | W | SbjMap | SbjMap | ADD: PreMap domain to SbjMap |
| deprecation | W | PreMap | PreMap | – |
| <code>owl:complementOf</code> | W | PreMap | SbjMap | – |

different than the predicate’s range –as defined by the corresponding vocabulary or ontology, the predicate’s range is added as class to the Referencing Object Map. Such a violation was reported at the example mentioned in the previous section (Section 5). Besides manual adjustments like defining `ex:Person` as equivalent or a subclass of `foaf:Person`, the statement that the Referencing Object Map type should be a `ex:Person`, can be replaced by a `foaf:Person`:

```

1 DEL: ex:objectMapX rr:class ex:Person .
2 ADD: ex:objectMapX rr:class foaf:Person.
3 MOD: adjust the definition of ex:Person

```

Automatically refining *domain-level* violations requires comparing recursively the type(s) assigned to the Subject Map with each predicate’s domain, as specified at the different Predicate Maps. If not explicitly defined or inferred via a subclass, the predicate’s domain is additionally assigned. This also requires a follow-up check for disjoint classes, which is of crucial importance especially when composition of different vocabularies and ontologies occurs.

Mapping Refinements Based on Dataset Quality Assessment. Violations identified when the MDs are instantiated with values from the input source, can lead to a new round of refinements, if violations can be associated with a certain MD.

Mapping Refinements Impact on Dataset Quality. The number of automated resolutions for violations detected at the mapping level depends on (i) the number of iterations over the data chunks of the input source (e.g., number of rows), (ii) the number of references to the input source (e.g., number of referred columns) and

(iii) the number of returned values from the input source for each reference. To be more precise, if \mathcal{I} is the number of iterations, \mathcal{R} is the set of references the input source, and $\mathcal{V}(r)$ values are returned for $r \in \mathcal{R}$, then the total number of errors per violation is equal to the number of triples generated from this mapping definition: $\mathcal{I} \cdot \prod_{r \in \mathcal{R}} \mathcal{V}(r)$. This means that the number of errors per violation identified (and resolved) at mapping level grows linearly in function of the number of iterations, and geometrically, in the worst case, if multiple references and returned values occur. For instance, assuming a mapping definition with 2 references to the input, where up to 3 values can be returned for each reference, contains a violation. Applied to an XML file with 1,000 elements, this could cause up to 9,000 error-prone triples in the worst case.

6 Use Cases and Adoption

Our Mapping Assessment and Refinement workflow with RML and RDFUnit is already being used in multiple different contexts. The DBpedia community adapted our mapping assessment solution to improve its mappings. Other popular, medium-sized datasets also benefit of our solution to ameliorate their mappings, such as DBLP. Moreover, various projects fully relied on our solution for their dataset generation, such as CDFLG and iLastic. Last, the proposed workflow was used to refine a challenge submission. Every dataset is unique in the way mappings are applied and different types of errors arise in each case. We indicatively describe a number of representative use cases below.

DBpedia [24] provides a *collaborative mapping approach* of Wikipedia infoboxes to the DBpedia ontology⁸ through the DBpedia *mappings wiki*⁹. DBpedia uses a wiki markup syntax for the mapping definitions and the output is adjusted in conformance to the DBpedia ontology. Although DBpedia uses the same wikitext syntax as Wikipedia –its original source– to define the MDS, the quality of wikitext-based MDS cannot be assessed directly, and thus certainly not in the same way as their resulting dataset. Thus, we automated the conversion of all DBpedia mappings to RML in order to make them processable from our tool stack. We introduced *wikitext serialisation* as a new Reference Formulation, since RML can be extended to express MDS for any type of input source. In total, we generated 674 distinct mapping documents for English, 463 for Dutch and a total of 4,468 for all languages. We used the DBpedia 2014 release and focused on a complete evaluation on the English and Dutch language editions as well as a mapping-only evaluation of all languages supported in the DBpedia mappings wiki. DBpedia originates from crowdsourced and (semi-)structured content and can thus be considered a noisy dataset. The MQA report was provided to the DBpedia community¹⁰, who took advantage of it to manually refine DBpedia MDS. Automated refinements were not applicable in this case –as DBpedia framework still functions with the original MDS in wiki markup.

⁸ <http://wiki.dbpedia.org/Ontology>

⁹ <http://mappings.dbpedia.org>

¹⁰ <http://goo.gl/KcSu3E>

Faceted DBLP. The Computer Science bibliography (DBLP) collects open bibliographic information from major computer science journals and proceedings. Faceted DBLP builds upon the DBLP++ dataset, an enhancement of DBLP, originally stored in a mysql database. DBLP MDs are originally defined using D₂RQ [5] and were converted to RML using D₂RQ-to-R₂RML¹¹ to be processable by our workflow. DBLP, is a *medium-sized* dataset of very good quality according to our evaluation. Nonetheless, the workflow resulted in improvements.

Contact Details of Flemish Local Governments Dataset (CDFLG).¹² In the scope of the EWI¹³ project, the CDFLG dataset was generated using our workflow [7]. This is a real case of contact details for local governments in Flanders. CDFLG is annotated using the OSLO ontology¹⁴, defined by the Open Standards for Linking Governments Working Group (V-ICT-OR, OSLO) under the OSLO (Open Standards for Local Administrations) Programme. Two subsequent versions of its RML mapping definitions were used to generate this dataset were assessed for their quality. The decrease of mapping violations over the mapping evolution indicates that our methodology can correctly identify errors.

iLastic.¹⁵ Our methodology was used in a use case for iMinds¹⁶, a research institute founded by the Flemish Government, which published its own data regarding researchers, publications, projects, external partners etc., using the proposed workflow. The mapping definitions that specify how the data is mapped to the RDF model were stated using RML. After the primary mapping definitions were stated, they were fed to our proposed implementation and were refined twice, once based on the MQA results and once based on the DQA results, leading to their final version which is free of violations.

CEUR-WS. The ESWC2015 Semantic Publishing Challenge (SPC)¹⁷ is focused on refining and enriching CEUR-WS¹⁸ linked dataset originally generated at the ESWC2014 edition¹⁹. It contains Linked Data about workshops, their publications and their authors. The workflow was well aligned with the requirements of this year’s challenge and was used to evaluate last year’s submission based on RML [9] and refine it to produce the base for this year’s submission [16].

7 Evaluation and Discussion

The datasets mentioned in Section 6 were used for our evaluation. In this section, the results are described and certain observations are discussed in more details

¹¹ https://github.com/RMLio/D2RQ_to_R2RML.git

¹² <http://ewi.mmlab.be/cd/all>

¹³ <http://ewi.mmlab.be>

¹⁴ <http://purl.org/oslo/ns/localgov#>

¹⁵ <http://explore.ilastic.be/>

¹⁶ <http://iminds.be>

¹⁷ <https://github.com/ceurws/lod/wiki/SemPub2015>

¹⁸ <http://ceur-ws.org>

¹⁹ <http://challenges.2014.eswc-conferences.org/index.php/SemPub>

Table 2. Evaluation results summary. In the *Dataset Assessment* part, we provide the Size (number of triples), number of test cases, evaluation Time, Failed test cases and total individual Violations. In the *Mapping Assessment* part, we provide the mapping document Size (number of triples), evaluation Time, Failed test cases and Violation instances. Finally, we provide the number of dataset violations that can be addressed refining the mappings and estimated corresponding dataset violations that are resolved.

| Dataset | Dataset Assessment | | | | | Mapping Assessment | | | | | Affect. triples |
|---------|--------------------|--------|------|-------|-------|--------------------|------|-------|-------|------|-----------------|
| | Size | TC | Time | Fail. | Viol. | Size | Time | Fail. | Viol. | Ref. | |
| DBpEn | 62M | 9,458 | 16.h | 1,128 | 3.2M | 115k | 11s | 1 | 160 | – | 255k |
| DBpNL | 21M | 10,491 | 1.5h | 683 | 815k | 53k | 6s | 1 | 124 | – | 106k |
| DBpAll | – | – | – | – | – | 511k | 32s | 1 | 1,316 | – | – |
| DBLP | 12M | 462 | 12h | 7 | 8.1M | 368 | 12s | 2 | 8 | 6 | 8M |
| iLastic | 150k | 690 | 12s | 23 | 37k | 825 | 15s | 3 | 26 | 23 | 37k |
| CDFLG | 0.6k | 2068 | 7s | 15 | 678 | 558k | 13s | 4 | 16 | 13 | 631 |
| CEUR-WS | 2.4k | 414 | 6s | 7 | 783 | 702 | 5s | 3 | 12 | 7 | 783 |

for each dataset and overall. The results are aggregated in Section 7 and are available at <http://rml.io/data/ISWC15>. For our evaluation, we used an 8-core Intel i7 machine with 8GB RAM and 256 SSD HD.

Overall, it is clear that the *computational complexity and time are significantly reduced* when assessing the mapping definitions compared to the complete RDF dataset (cf. Section 7). It takes 11 seconds to assess the approximately 700 mappings of English DBpedia, compared to assessing the whole DBpedia dataset that takes of the order of several hours. In the latter case, the assessment requires examining each triple separately to identify, for instance, that 12M triples violated the range of `foaf:primaryTopic`, whereas with our proposed approach, only 1 triple needs to be examined. It is indisputable the workflow’s *effectiveness*, as, in all cases that the dataset generation fully relies on its mapping definitions, the majority of violations is addressed. Moreover, if a set of RML mapping definitions is assessed for its quality, for every other new data source also mapped using these mapping definitions, the quality assessment does not need to be repeated for that part. Next, we discuss the results for each dataset in details:

DBpedia. Most violations in DBpedia have a *range-level* origin. When RDF is generated from the wikitext, the object type is not known and may result in wrong statements, as the DBpedia extraction framework automatically adjusts the predicate/object extraction according to the DBpedia ontology definitions. *Domain-level* violations occur as well, because users manually provide the class a Wikipedia infobox is mapped to and the ontology properties each infobox property will use. Our framework can, in this case, identify mismatches between the user-defined class and the `rdfs:domain` of each provided property. We observe that 8% of the errors in DBpedia in English and 13% of DBpedia in Dutch can be fixed directly at *mapping-level*. Not only are the errors as such directly pinpointed, but it also takes negligible time to have the refinements of the violations accomplished. The evaluation of all mappings for all 27 supported language editions resulted in a total of 1316 *domain-level* violations.

DBLP dataset has 7 individual violations, leading to 8.1M violated triples. The `swrc:editor` predicate defined in a Predicate Map expects a resource of `swrc:Person` type for its domain instead of `foaf:Agent` as defined in the corresponding Subject Map causing 21k errors. Similarly, approximately 3M errors occurred because a Predicate Map exists with `dcterms:bibliographicCitation` as its value whose `rdfs:domain` is `bibo:BibliographicResource`. However, the corresponding Subject Map(s) generate resources of type `dcmitype:Text`, `foaf:Document` or `swrc:Book` but definitely not the expected one, thus data publishers should remain warned for potential contradictions. Moreover, the missing range of `foaf:page` and `foaf:homepage` can be fixed by refining the mapping definitions but, for links to external resources, it is common practice not to define their type. Except for 12k inverse functional violations for `foaf:homepage` that can not be addressed directly from the mapping definitions, all remaining violations (98%) could be refined.

CDFLG. In the first version of the CDFLG dataset, we found four violations: One caused by Predicate Object Maps that all have predicates that expect `oslo:Address` as their domain. However, the Subject Map is defined to be of type `oslo:BasicAddress`. In the same context, an incorrect range violation was identified for `oslo:availableAt` property. In general, violations related to Referencing Object Maps are among the most frequently encountered. Last, the object property schema:`nationality` was mapped as literal. The second version of CDFLG is a result of manually refining the mapping definitions according to the first mapping assessment's results. Besides the *domain level* violation, only few of the range violations remained (7%).

iLastic is particularly interesting because the workflow was used from the primary version of the mapping definitions, until they became free of violations. The first version was assessed and even contained R2RML schema violations, e.g., `rr:constant` had a string-valued object instead of a resource. If these mapping definitions were used, almost one fourth (25%) of its triples would be prone to errors. Every violation was fixed after a couple of iterations assessing and refining the mapping definitions. For example, `cerif:isClassifiedBy` expects a `cerif:Classification` and not a `skos:Concept`, while `bibo:uri` expects a literal and not a resource as range. Similarly, `dcterms:issued` expects `xsd:date` and not `xsd:gYear`. A violation that occurred repeatedly was associated with the `cerif:internalIdentifier` that requires a string-valued object, whereas it was associated with an Object Map that generated `xsd:positiveInteger` objects.

CEUR-WS. 12 violations were identified in the dataset generated using RML for ESWC 2014 challenge and 10 out of them could already be detected at the mapping definitions. Most of them (7) were *domain-level* violations, annotating, for instance, resources of type `bibo:Volume` with properties for `bibo:Document` or for `<http://www.loc.gov/mads/rdf/v1#Address>`, e.g., for specifying the city, implying unwittingly that resources are both *Documents* and *Addresses*. The rest of the detected violations were related to contradicted datatypes, for

instance, incorrectly specifying the datatype as `xsd:gYear`, while it is expected to be string. The mapping definitions for ESWC 2015 submission were produced using our workflow, were assessed and do not contain violations any more.

8 Related Work

We summarize the state of the art of the relevant fields: data mappings to the RDF data model and Linked Data quality assessment.

Mapping Languages. Several solutions exist to perform mappings from different data formats and serialisations to the RDF data model. In the case of data in XML format, existing XML solutions were used to define the mappings, such as XSLT, e.g., AstroGrid-D²⁰, or XPath, e.g., Tripliser²¹, while the only mapping language defined specifically for XML to RDF mappings is X3ML²². In the same context, existing querying languages were also considered to describe the mappings, e.g., XSPARQL [2] which is a language that combines XQuery and SPARQL or Tarql²³. Due to the lack of query languages or other ways to refer to data in CSV format or spreadsheets, different mapping languages were occasionally defined, e.g., the XLWrap’s mapping language [22] that converts data in spreadsheets to RDF, or the declarative OWL-centric mapping language Mapping Master’s M^2 [26] that converts data from spreadsheets into the Web Ontology Language (OWL). For relational databases, different mapping languages were defined [15], but the W3C-standardized R2RML prevailed.

Quality Assessment. Different approaches have been developed that try to tackle various aspects of Linked Data quality. These approaches can be broadly classified into (i) manual (e.g. [1, 3, 25, 29]); (ii) semi-automated (e.g. [11, 17]); or (iii) automated (e.g. [8, 14]) methodologies. These approaches introduce systematic methodologies to assess the quality of a dataset. Depending on the approach, we notice inability to produce easily interpretable results, a considerable amount of user involvement, application on specific datasets only or inability to evaluate a complete dataset during the assessment. SPIN²⁴ is a W3C submission aiming at representing rules and constraints on Semantic Web models using SPARQL. The approach described in [13] advocates the use of SPARQL and SPIN for RDF data quality assessment. In a similar way, Fürber et al. [12] define a set of generic SPARQL queries to identify missing or invalid literal values and datatypes and functional dependency violations. Another related approach is the *Pellet Integrity Constraint Validator*²⁵, which translates OWL integrity constraints into SPARQL queries. A more light-weight, although less expressive, RDF

²⁰ <http://www.gac-grid.de/project-products/Software/XML2RDF.html>

²¹ <http://daverog.github.io/tripliser/>

²² <https://github.com/delving/x3ml/blob/master/docs/x3ml-language.md>

²³ <https://github.com/cygri/tarql>

²⁴ <http://www.w3.org/Submission/spin-overview/>

²⁵ <http://clarkparsia.com/pellet/icv/>

constraint syntax that is decoupled from SPARQL is offered from *Shape Expressions* (ShEx) [27] and *IBM Resource Shapes*²⁶.

9 Conclusions and Future Work

In this paper, we propose a methodology for assessing Linked Data quality for data originally stemming from (semi-)structured formats. We propose a workflow that relies on assessing the mapping definitions, rather than the RDF dataset they generate. The assessment report points exactly to the root causes of the violations and can be actively used to refine the mapping definitions. The automation of refinements or suggestions is facilitated based on a comprehensive analysis of different cases, and encountered violations are addressed at the origin. This essentially allows publishers to identify and correct violations before they even occur; moreover, fixing violations early avoids propagation where one flawed mapping rule leads to many faulty triples. The evaluation shows that our methodology is applicable to (i) datasets without native [R2]RML mapping definitions, such as DBLP, (ii) large datasets, such as DBpedia, as well as (iii) datasets in the whole process of defining their mappings, such as iLastic. It was proven that assessing the quality of mapping definitions is more efficient in terms of computational complexity, and requires significantly less time to be executed compared to assessing the entire dataset. As our evaluation indicates, it takes only a few seconds to assess the mapping definitions, while it can be time-consuming and performance-intensive when this happens at dataset level. Especially with large datasets, this can take up to several hours. Our methodology was adopted by both the community of significant public datasets, such as DBpedia, and several projects, resulting in published Linked Data of higher quality. In the future, we plan to automate and improve the application of mapping definition refinements and integrate this step into the workflow of an interactive user interface.

Acknowledgements. This paper’s research activities were funded by Ghent University, iMinds, the Institute for the Promotion of Innovation by Science and Technology in Flanders, the Fund for Scientific Research-Flanders and by grants from the EU’s 7th & H2020 Programmes for projects ALIGNED (GA 644055), GeoKnow (GA 318159) and LIDER (GA 610782).

References

1. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S., Lehmann, J.: Crowdsourcing linked data quality assessment. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 260–276. Springer, Heidelberg (2013)
2. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. *Journal on Data Semantics* **1**(3), 147–185 (2012)

²⁶ <http://www.w3.org/Submission/2014/SUBM-shapes-20140211/>

3. Bizer, C., Cyganiak, R.: Quality-driven information filtering using the WIQA policy framework. *Web Semant.* **7**(1), 1–10 (2009)
4. Boehm, B.W.: *Software Engineering Economics*. Prentice Hall PTR (1981)
5. Cyganiak, R., Bizer, C., Garbers, J., Maresch, O., Becker, C.: *The D2RQ Mapping Language*. Technical report, March 2012
6. Das, S., Sundara, S., Cyganiak, R.: *R2RML: RDB to RDF Mapping Language*. Working Group Recommendation W3C, September 2012
7. De Vocht, L., Van Compernelle, M., Dimou, A., Colpaert, P., Verborgh, R., Mannens, E., Mechant, P., Van de Walle, R.: Converging on semantics to ensure local government data reuse. In: *Proceedings of the 5th workshop on Semantics for Smarter Cities (SSC14), 13th International Semantic Web Conference (ISWC) (2014)*
8. Debattista, J., Lange, C., Auer, S.: Representing dataset quality metadata using multi-dimensional views. In: *Proceedings of the 10th International Conference on Semantic Systems*, pp. 92–99 (2014)
9. Dimou, A., Vander Sande, M., Colpaert, P., De Vocht, L., Verborgh, R., Mannens, E., Van de Walle, R.: Extraction & semantic annotation of workshop proceedings in HTML using RML. In: *Sem. Publ. Challenge of 11th ESWC (2014)*
10. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: *Workshop on Linked Data on the Web (2014)*
11. Flemming, A.: *Quality characteristics of linked data publishing datasources*. Master's thesis, Humboldt-Universität of Berlin (2010)
12. Fürber, C., Hepp, M.: Using semantic web resources for data quality management. In: Cimiano, P., Pinto, H.S. (eds.) *EKAUW 2010*. LNCS, vol. 6317, pp. 211–225. Springer, Heidelberg (2010)
13. Fürber, C., Hepp, M.: Using SPARQL and SPIN for data quality management on the semantic web. In: Abramowicz, W., Tolksdorf, R. (eds.) *BIS 2010*. LNBIP, vol. 47, pp. 35–46. Springer, Heidelberg (2010)
14. Guéret, C., Groth, P., Stadler, C., Lehmann, J.: Assessing linked data mappings using network measures. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012*. LNCS, vol. 7295, pp. 87–102. Springer, Heidelberg (2012)
15. Hert, M., Reif, G., Gall, H.C.: A comparison of RDB-to-RDF mapping languages. In: *I-Semantics 2011*, pp. 25–32. ACM (2011)
16. Heyvaert, P., Dimou, A., Verborgh, R., Mannens, E., Van de Walle, R.: Semantically annotating CEUR-WS workshop proceedings with RML. In: *Semantic Publishing Challenge of the 12th ESWC (2015)*
17. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: *Linked Data On the Web (2010)*
18. Hyland, B., Ateazing, G., Villazón-Terrazas, B.: *Best Practices for Publishing Linked Data*. Working Group Note, W3C, January 2004
19. Juran, J., Gryna, F.: *Juran's Quality Control Handbook*. Industrial engineering series. McGraw-Hill (1988)
20. Kontokostas, D., Brümmer, M., Hellmann, S., Lehmann, J., Ioannidis, L.: NLP data cleansing based on linguistic ontology constraints. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) *ESWC 2014*. LNCS, vol. 8465, pp. 224–239. Springer, Heidelberg (2014)
21. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.: Test-driven evaluation of linked data quality. In: *Proceedings of the 23rd International Conference on World Wide Web*, pp. 747–758 (2014)

22. Langegger, A., Wöß, W.: XLWrap – querying and integrating arbitrary spreadsheets with SPARQL. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 359–374. Springer, Heidelberg (2009)
23. Lehmann, J.: DL-learner: Learning concepts in description logics. *Journal of Machine Learning Research* **10**, 2639–2642 (2009)
24. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Klef, P., Auer, S., Bizer, C.: DBpedia - a Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Sem. Web Journal* (2014)
25. Mendes, P.N., Mühleisen, H., Bizer, C.: Sieve: linked data quality assessment and fusion. In: EDBT/ICDT Workshops, pp. 116–123. ACM (2012)
26. O’Connor, M.J., Halaschek-Wiener, C., Musen, M.A.: Mapping master: a flexible approach for mapping spreadsheets to OWL. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 194–208. Springer, Heidelberg (2010)
27. Prud’hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: an RDF validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems, pp. 32–40. ACM (2014)
28. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
29. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data: A survey. *Semantic Web Journal* (2015)

A Generic RDF Transformation Software and Its Application to an Online Translation Service for Common Languages of Linked Data

Olivier Corby¹ (✉), Catherine Faron-Zucker², and Fabien Gandon¹

¹ Inria, Sophia Antipolis, France
olivier.corby@inria.fr

² University Nice Sophia Antipolis, Sophia Antipolis, France
faron@i3s.unice.fr

Abstract. In this article we present a generic template and software solution for developers to support the many cases where we need to transform RDF. It relies on the SPARQL Template Transformation Language (STTL) which enables Semantic Web developers to write specific yet compact RDF transformers toward other languages and formats. We first briefly recall the STTL principles and software features. We then demonstrate the support it provides to programmers by presenting a selection of STTL-based RDF transformers for common languages. The software is available online as a Web service and all the RDF transformers presented in this paper can be tested online.

Keywords: RDF · SPARQL · STTL · Transformation · Software · Tool

1 Introduction

The RDF standard [8] provides us with a general purpose graph-oriented data model recommended by the W3C to represent and interchange data on the Web. While the potential of a world-wide Semantic Web of linked data and linked data schemas is now widely recognized, the transformation and presentation of RDF data is still an open issue. Among the initiatives to answer this question there are extensive works for providing RDF with several varied syntaxes (XML, N-Triples, Turtle, RDFa, TriG, N-Quads, JSON-LD) and for linking it to other data sources (R2RML, CSV-LD, etc.). With the multiplication of data sources and data formats, developers of the Web of data now spend a lot of time and energy to build transformations to present RDF data to users and transform data from one source to another. Moreover, a special case of RDF data holds a very special potential as RDF is more and more used as a syntax to represent other languages. For instance in the domain of the Semantic Web alone, this is the case of three W3C standards: OWL 2 [13] is provided with several syntaxes, among which the Functional syntax, the Manchester syntax used in several ontology editors and RDF/XML and RDF/Turtle; the Rule Interchange Format (RIF) [10] is provided with several syntaxes among which an XML syntax, two compact

syntaxes for RIF-BLD and RIF-PRD and an RDF syntax; SPARQL Inference Notation (SPIN) is a W3C member submission [12] to represent SPARQL rules in RDF, to facilitate storage and maintenance. Many other languages can (and will) be “serialized” into RDF. For instance [9] is an attempt to represent SQL expressions in RDF, and we consider that RDF can then be viewed as a pivot language to represent the abstract syntax trees of expressions of other languages.

For this reason, we present a software solution for developers to support the many cases where we need to transform RDF. We rely on the SPARQL Template Transformation Language (STTL) which enables Semantic Web developers to write specific yet compact RDF transformers toward other languages and formats. As an example of application of STTL we can mention the implementation of a SPARQL tutorial that was successfully used in a Semantic Web MOOC with over 4000 students¹

Since the applications of STTL are varied we categorized them into five main families: the case of transformation between RDF syntaxes (e.g., RDF/XML to Turtle); the generation of presentation formats (e.g., HTML pages); the exports towards other data formats (e.g., CSV); the pretty-printing of statements of a given language represented in RDF syntax (e.g., SPIN to SPARQL); and the use of RDF as a pivot model between two languages. In addition to the above cited transformations which we already have developed, we are receiving expressions of interest for new transformations including the translation to graph file formats like DOT², to Topic Maps³, the generation of PROV-O and Ratio4TA annotations, the anonymization of data, etc. and several of these transformation are currently being developed. An alternative or complementary categorization is the distinction between domain independent transformations (e.g., transformations between RDF syntaxes, generic presentations of RDF triples into HTML tables, OWL/RDF to OWL/FS transformation) and domain or application dependent transformations (e.g., a customized export to CSV or HTML rendering).

In this article we briefly recall STTL principles. Then we present the software features and we demonstrate the support this software provides to programmers by presenting a selection of STTL-based RDF transformers for common languages. The software is available online as a Web service and all the RDF transformers presented in this paper can be tested online⁴. The paper is organized as follows: Sect. 2.1 presents the STTL language. Section 3 presents the generic transformation rule engine we developed to implement STTL and the Web service encapsulating it. The following sections present several specific RDF transformers written in STTL, illustrating the different families of transformations we identified. Section 7 concludes.

¹ https://www.france-universite-numerique-mooc.fr/courses/inria/41002/Trimestre_1.2015/about.

² <http://www.graphviz.org/Documentation.php>.

³ <http://www.topicmaps.org/>.

⁴ <http://corese.inria.fr>.

2 Transforming RDF

2.1 SPARQL Template Transformation Language (STTL)

STTL is a generic transformation rule language for RDF which relies on two extensions of SPARQL: an additional `TEMPLATE` query form to express transformation rules and extension functions to recursively call the processing of a template from another one.

A `TEMPLATE` query is made of a standard `WHERE` clause and a `TEMPLATE` clause. The `WHERE` clause is the condition part of a rule, specifying the nodes in the RDF graph to be selected for the transformation. The `TEMPLATE` clause is the presentation part of the rule, specifying the output of the transformation performed on the solution sequence of the condition. For instance, let us consider the OWL axiom stating that the class of parents is equivalent to the class of individuals having a person as child. Here are its expressions in Functional syntax and in Turtle:

```
EquivalentClasses(
  a:Parent
  ObjectSomeValuesFrom(a:hasChild a:Person))

a:Parent a owl:Class ;
  owl:equivalentClass
    [ a owl:Restriction ;
      owl:onProperty a:hasChild ;
      owl:someValuesFrom a:Person ]
```

The following template enables to transform the above `equivalentClass` statement from RDF into Functional syntax:

```
TEMPLATE {
  "EquivalentClasses("
  st:apply-templates(?in) ""
  st:apply-templates(?c) ")" }
WHERE { ?in owl:equivalentClass ?c . }
```

The value matching variable `?in` is `a:Parent` which is expected in the transformation output (the Functional syntax of the OWL 2 statement), while the value matching variable `?c` is a blank node⁵ whose property values are used to build the expected output. This is defined in another template to be applied on this focus node. The `st:apply-templates` extension function enables this recursive call of templates, where `st` is the prefix of STTL namespace⁶

More generally, `st:apply-templates` function can be used in any template t_1 to execute another template t_2 that can itself execute a template t_3 , etc. Hence,

⁵ Let us note that blank nodes are handled like any other node (URIs and literals). If needed, function `isBlank()` enables to detect them to apply specifically written templates on them.

⁶ <http://ns.inria.fr/sparql-template/>

templates call themselves one another, in a series of call, enabling a hierarchical processing of templates and a recursive traversing of the target RDF graph. Similarly, `st:call-template` function can be used to call named templates. Table 1 summarizes the list of the Core STTL functions.

Table 1. STTL Core Functions

| Name | Description |
|--------------------------------------|--|
| <code>st:apply-templates</code> | Apply current transformation on focus node |
| <code>st:apply-templates-with</code> | Apply given transformation on focus node |
| <code>st:apply-templates-all</code> | Apply all templates on focus node |
| <code>st:call-template</code> | Apply named template on focus node |
| <code>st:call-template-with</code> | Apply named template of given transformation on focus node |
| <code>st:define</code> | Define a template function (e.g. <code>st:process</code>) |
| <code>st:process</code> | Define the processing of template variables |
| <code>st:get</code> | Get a property value from Context |
| <code>st:set</code> | Set a property value into Context |
| <code>st:turtle</code> | Display Turtle syntax of focus node |
| <code>st:nl</code> | Insert newline |

Following the layer-cake standardization of the semantic Web, STTL is compiled into standard SPARQL. This allows the approach to be usable with different implementations of the standard, to benefit from its expressiveness, from the native extension mechanisms and also from the optimizations of the implementations. The compilation keeps the WHERE clause, the solution modifiers and the VALUES clause of the template unchanged and the TEMPLATE clause is compiled into a SELECT clause. This also allows STTL to benefit from all SPARQL features for instance, when needed, the DISTINCT solution modifier can be used in a nested subquery to avoid duplicates. For instance, the TEMPLATE clause of the following STTL template:

```

TEMPLATE {
  "ObjectSomeValuesFrom(" ?p "" ?c )" }
WHERE {
  ?in a owl:Restriction ;
  owl:onProperty ?p ;
  owl:someValuesFrom ?c }

```

is compiled into the following standard SPARQL SELECT clause:

```

SELECT
  (CONCAT("ObjectSomeValuesFrom(",
    st:process(?p), "",
    st:process(?c), ")") AS ?out)

```

The WHERE clause is unchanged.

2.2 Work Related to STTL

A complete description of STTL language is provided in [6] together with an extended presentation of the state-of-the-art approaches addressing the problem of RDF transformation. We briefly summarize here this state-of-the-art of languages addressing the problem of RDF transformation stressing that STTL is independent of the syntax of its RDF input and addresses the general problem of the *transformation* of RDF data into any output format.

OWL-PL [4] is an extension of XSLT for transforming RDF/OWL into XHTML; it is both tied to its RDF/XML input format and its output format. Fresnel [3] is an RDF vocabulary for specifying in RDF which data contained in an RDF graph should be displayed and how. Again, it is tied to a specific display paradigm and an XHTML-like output format.

SPARQL is provided with a CONSTRUCT query form which enables to extract and *transform* RDF data into RDF according to any other schema. [1] addresses the problem of generating XML from RDF data with an extended SPARQL query. Here again, the solution is specific to one output format. XSPARQL [2] is a combination of SPARQL and XQuery [15] enabling to query both XML and RDF data and to transform data from one format into the other. [16] proposes an XML-based transformation language, inspired by XSLT, that mainly matches types of RDF resources. [14] proposes an XML-based stylesheet language also inspired by XLST where templates match triple patterns and generate HTML.

Finally, there is quite a wide range of ad hoc RDF parsers and validators⁷, some of which enable to transform RDF data from one syntax into another. Among them, let us cite RDF Distiller⁸ and RDF Translator⁹. A review of these RDF-to-RDF converters can be found in [17]. Another famous example of specific-purpose RDF transformer is the RDF/XML parser in OWL API¹⁰ [11] which enable to transform OWL 2 statements in RDF/XML into the Functional syntax of the language.

3 STTL Engine

We implemented a STTL engine within the Corese Semantic Web Factory¹¹ [5, 7]. It comprises an STTL RESTful Web service to process STTL transformations. In this section, we first describe our implementation of STTL then the Web service which encapsulate the STTL engine.

3.1 Implementation of STTL

Algorithm. Basically, the STTL engine is called by `st:apply-templates` or other alike extension functions. Given an RDF graph with a focus node to be

⁷ <http://www.w3.org/2001/sw/wiki/Category:Tool>.

⁸ <http://rdf.greggkelllogg.net/distiller>.

⁹ <http://rdf-translator.appspot.com/>.

¹⁰ <http://owlapi.sourceforge.net/>.

¹¹ <http://wimmics.inria.fr/corese>.

transformed and a list of templates, it successively tries to apply them to the focus node until one of them succeeds. A template succeeds if the matching of the WHERE clause succeeds, i.e., returns a result. If no template succeeds, the `st:default` named template (if any) is applied to the focus node. Recursive calls to `st:apply-templates` within templates implements the graph recursive traversal with successive focus nodes. The engine keeps track of the templates applied to nodes in order to avoid cycles, i.e. to avoid to apply the same template on the same node twice in case the RDF graph is cyclic. If there is no template to be applied on a focus node that has not previously been applied on it, the transformer calls the `st:default` named template if any, otherwise the Turtle format of the focus node is returned.

Template Selection. By default, the STTL engine considers templates in order: given a focus node, in response to a call to the `st:apply-templates` function, it considers the first template that matches this node. Alternatively, the processing of a named template is commanded by a call to the `st:call-template` function. In both cases, the result of the transformation of the focus node is the result of the template.

In some other cases, it is worth writing *several* templates for a type of node, in particular when the node holds different graph patterns that should be transformed according to different presentation rules. Executing several templates on the focus node is done by calling the `st:apply-templates-all` function. The result of the transformation is the concatenation of the results of the successful templates.

A transformer can be used to transform a whole RDF graph — without any distinguished root node in the graph. For this purpose, the `st:apply-templates-with` function can be called without focus node and the transformer must then determine it. By default, the first template that succeeds is the starting point of the transformer; or a `st:start` named template can be defined to be executed first.

Transformation Settings. The `st:start` named template, if any, is selected at the beginning of the transformation process when no focus node is available. In that case, it is the first template executed by the template engine. The `st:default` template, if any, is executed when all templates fail to match the focus node.

The processing of a variable in the TEMPLATE clause by default consists in outputting its value in the Turtle format. The `st:profile` template can be used to overload this default transformation behaviour. For example, the following definition of `st:profile` specifies that processing variables, denoted by `st:process(?x)`, consists in the application of the `st:apply-templates` function to it.

```
TEMPLATE st:profile {
  st:define( st:process(?x) = st:apply-templates(?x) )
WHERE { }
```

3.2 STTL-Based RDF Transformers

In our approach of RDF transformation based on STTL, the STTL engine, i.e. the template processor, is generic: it applies to any RDF data with any set of STTL templates. What is specific to each transformation is the set of STTL templates defining it. In other words, each RDF transformer specific to an output format is defined by a specific set of STTL templates processed by the generic template processor implementing STTL. In Sects. 4 to 6, we present specific STTL-based RDF transformers. Each RDF transformer may be accessed as a Web service. We present in Sect. 3.4 our implementation of a STTL RESTfull Web service in the Corese Semantic Web Factory.

3.3 STTL Development Environment

The Corese Semantic Web Factory provides a standalone environment with a GUI, enabling the user to load RDF data and ontologies, to load or write SPARQL queries and STTL templates as well and easily test them against the loaded RDF data. This tool enables developers an easy handling, with a fast learning curve, of Semantic Web technologies in general, and of STTL in particular.

3.4 STTL Service

The Corese Semantic Web Factory provides a SPARQL endpoint by means of a RESTfull Web service which implements SPARQL 1.1 Protocol¹². In addition to this standard implementation, Corese proposes an STTL RESTful Web service to process STTL transformations on local or distant RDF dataset. Figure 1 presents the general architecture of the server.

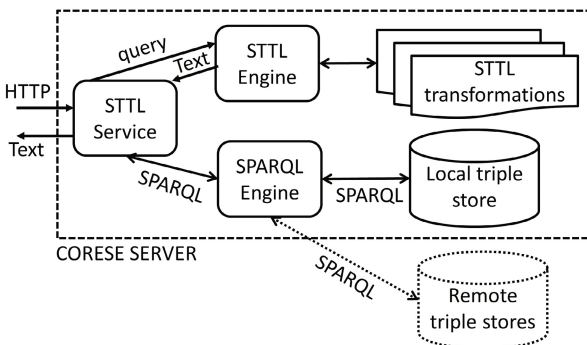


Fig. 1. Architecture of a Corese STTL-based RDF Transformation Server

¹² <http://www.w3.org/TR/sparql11-protocol/>.

A request for an STTL transformation of RDF data is conveyed to the transformation service in a URL whose hierarchical part ends with `/template`¹³ and whose query part comprises key-value pairs specifying the request to the service. The `query` key is reused from SPARQL 1.1 Protocol to indicate a SPARQL query to be performed by the server on its RDF dataset. The URL of the STTL transformation to be applied to the result of the SPARQL query (or to the RDF dataset) is specified as the value of a `transform` key. For instance, the following URL asks for the transformation of all the triples of the RDF dataset with a STTL transformation specified at `st:sparql`.

```
http://localhost:8080/template?
  query=SELECT * WHERE { ?x ?p ?y }&transform=st:sparql
```

In order to simplify the interaction with a STTL service, we define the notion of *profile* of a transformation, assembling an optional SPARQL query and a STTL transformation into a simple workflow. A profile is described in RDF as follows: a `st:query` property and a `st:transform` property associate a SPARQL query and a STTL transformation to a profile. Here is an example of a profile description:

```
st:dbpedia a st:Profile ; st:query <q1.rq> ; st:transform st:navlab .
```

with `q1.rq` containing for instance the following SPARQL query:

```
CONSTRUCT { ?x ?p ?y }
WHERE { SERVICE <http://fr.dbpedia.org/sparql> { ?x ?p ?y } }
```

In the URL conveying the request for a STTL transformation, the URI of a profile is indicated to the STTL service as the value of a `profile` key. Here is an example of such a URL:

```
http://localhost:8080/template?profile=st:dbpedia
```

We defined the notion of *context* of a transformation that enables the STTL service to send parameters to the transformer. This context is set up by the service and passed to the transformer. The transformer can access the context with the `st:get` extension function which we have defined to return the value of a context parameter. Context parameters are the name of the service, the service profile, the transformation, the query, and the focus URI. For instance here is the context of an RDF-to-HTML transformation described in Sect. 5.

```
st:get(st:service)    = /template
st:get(st:profile)   = st:dbpedia
st:get(st:transform) = st:navlab
st:get(st:query)     =
  CONSTRUCT { ?x ?p ?y }
  WHERE { SERVICE <http://fr.dbpedia.org/sparql> {?x ?p ?y} }
st:get(st:uri)       = http://fr.dbpedia.org/resource/Antibes
```

¹³ e.g., <http://corese.inria.fr/template>.

4 RDF-to-RDF Transformers

The first family of RDF transformations we identified in the introduction of this paper comprises RDF-to-RDF transformations, i.e., the transformation of RDF data from any RDF syntax into any other RDF syntax, e.g., RDF/XML-to-Turtle. In this family the transformations are completely domain-independent. There exist several online transformation services such as RDF Translator for instance but we show here how to implement your own transformation in a declarative way with STTL. As an exercise in style, we wrote RDF-to-Turtle and RDF-to-RDF/XML transformations. The main template of the RDF-to-Turtle transformation is shown below, it is available online¹⁴:

```

TEMPLATE st:start {
  st:apply-templates(?x) }
WHERE {
  SELECT DISTINCT ?x WHERE {
    ?x ?p ?y
    FILTER (
      isURI(?x) ||
      NOT EXISTS { ?z ?q ?x } ||
      ( isBlank(?x) &&
        ( EXISTS { ?a ?q ?x . ?b ?r ?x . FILTER(?a != ?b) } ||
          EXISTS { ?x ?q ?x } ) ) )
  } ORDER BY ?x}

```

A demo RDF/XML-to-Turtle transformation service is available online¹⁵.

We wrote an RDF-to-RDF/XML transformation as a set of 21 STTL templates available online¹⁶. Here is a key template of it to express the description of a resource as the subject of some RDF triples:

```

TEMPLATE st:resource(?in) {
  "<rdf:Description" ?att "=" str(?in) ">"
  box { st:call-template(st:property, ?in) }
  "</rdf:Description>"
}
WHERE { BIND ( if (isBlank(?in), rdf:nodeID, rdf:about) as ?att) }

```

Based on these STTL transformations, the Corese Semantic Web Factory enables to deploy RDF/XML-to-Turtle, RDFa-to-Turtle, Turtle-to-RDF/XML and RDFa-to-RDF/XML transformation services since the distribution is provided with a Turtle parser, an RDF/XML parser, and an RDFa parser.

5 RDF-to-HTML Transformers

The second family of RDF transformations identified in the introduction of this paper gathers the transformations of RDF data for presentation purposes, in

¹⁴ <http://ns.inria.fr/sparql-template/turtle>.

¹⁵ <http://corese.inria.fr>.

¹⁶ <http://ns.inria.fr/sparql-template/rdfxml>.

any given presentation format. In particular, it comprises RDF-to-HTML transformations. In this section, we present three examples of RDF-to-HTML transformations which enables to design Linked Data navigators. A demo server is accessible online¹⁷. Its source code is freely available within the Corese Semantic Web Factory¹⁸.

5.1 Principles of Linked Data Navigation: Dynamic HyperLinks to the STTL Service

The keys to build a Linked Data navigator is (1) to generate HTML pages for RDF resources and (2) to generate hyperlinks in the HTML code output. This is achieved with *href* attributes having as value a URL conveying a request for STTL transformation to the transformation service. Here is an example of a named STTL template to construct a hyperlink to a focus URI *?x*:

```
TEMPLATE st:link(?x) {
  "<a href='/template?profile=st:dbpedia&uri=" str(?x) "'>" str(?x) "</a>"}
WHERE { }
```

In order to avoid hardcoding the service and the profile names, these can be extracted from the context of the transformation. Here is an example of such a generic template:

```
TEMPLATE st:link(?x) {
  "<a href=''" st:get(st:service) "?profile=" st:get(st:profile)
  "&uri=" str(?x) "'>" str(?x) "</a>" }
WHERE { }
```

When applied on a given URI the two above templates would produce for instance the following output code:

```
<a href='/template?profile=st:dbpedia&uri=http://fr.dbpedia.org/resource/Antibes'>
  http://fr.dbpedia.org/resource/Antibes
</a>
```

5.2 Three Examples of Linked Data Navigators

Basic Navigator. The simplest navigator enables the user to send a SPARQL query to the server and get back the results presented in an HTML table in a completely domain-independent way. The results of SELECT queries are translated in RDF by using the Vocabulary for recording query result set published by the W3C RDF Data Access Working Group¹⁹. The *st:sparql* transformation²⁰ is then applied on this RDF result graph. The output HTML code contains hyperlinks to URLs, conveying further requests to the STTL server to apply a transformation to the resources involved in the description produced.

Here is an example of query solution in RDF:

¹⁷ <http://corese.inria.fr>.

¹⁸ <http://wimmics.inria.fr/corese>.

¹⁹ <http://www.w3.org/2001/sw/DataAccess/tests/result-set.n3>.

²⁰ <http://ns.inria.fr/sparql-template/sparql>.

```
prefix rs:<http://www.w3.org/2001/sw/DataAccess/tests/result-set#>
[] rs:resultVariable "x", "n" ;
  rs:solution [
    rs:binding [
      [rs:variable "x" ; rs:value ex:Auguste],
      [rs:variable "n" ; rs:value "Auguste"] ] ] .
```

Here is the main template of `st:sparql` that processes such an RDF graph solution of a SELECT query processed by the server.

```
prefix rs:<http://www.w3.org/2001/sw/DataAccess/tests/result-set#>
TEMPLATE {
  "<td>" COALESCE(st:call-template(st:display, ?val), "&nbsp;")
  "</td>" }
WHERE {
  ?x rs:solution ?in ; rs:resultVariable ?var .
  OPTIONAL { ?in rs:binding [ rs:variable ?var ; rs:value ?val ] }
} ORDER BY ?var
```

The WHERE clause focus is a solution `?in` which is a set of variable bindings. The OPTIONAL clause enumerates these bindings. This enumeration is optional because some variables (`?var`) may be unbound. For each binding, the TEMPLATE clause generates a cell, in an HTML table, with the value of the variable (`?val`). For unbound variables, a space character is generated in the cell.

DBpedia Navigator. Another kind of navigators are domain-specific Linked Data Navigators. We developed such a navigator — a server with its STTL service and the `st:navlab` RDF-to-HTML transformation — to browse the DBpedia dataset, specifically on persons and places. Figure 2 is the screenshot of an HTML page produced by this navigator. We wrote the `st:navlab` transformation as a set of 24 STTL templates which are available online²¹. Here is a template in `st:navlab`, to construct the table of resource descriptions; it recursively calls the `st:title` named template to output the title in HTML and the `st:descresource` to build the description of each resource selected in DBpedia.

```
TEMPLATE {
  st:call-template(st:title, ?in, ?label, (coalesce(?ic, "")))
  "<table>" st:call-template(st:descresource, ?in) "</table>" }
WHERE {
  ?in a <http://dbpedia.org/ontology/Resource> .
  ?in rdfs:label ?label FILTER(lang(?label) = 'fr')
  OPTIONAL { ?in <http://dbpedia.org/ontology/thumbnail> ?ic } }
```

The DBpedia SPARQL endpoint is accessed through a SERVICE clause in a predefined CONSTRUCT query to retrieve relevant data according to the types of resources that the application is interested in: our navigator focuses on historical

²¹ <http://ns.inria.fr/sparql-template/navlab>.



Fig. 2. DBpedia Navigator

people and places. Then the `st:navlab` transformation is applied to the resulting RDF graph. It generates a presentation in HTML format of the retrieved data, adapted to the type of targeted resources — people and places. In particular, the transformation localizes places on a map.

As it can be viewed in Fig. 2, when following the hyperlink generated by the DBpedia navigator, a request is sent to the STTL server to produce an HTML presentation of the DBpedia resource on Augustus, according to the `st:dbpedia` profile (embedding the `st:navlab` transformation). The interest of this STTL-based approach of DBpedia-to-HTML transformation is that it is declarative and can therefore easily be extended to handle the presentation of other types or resources by adding new dedicated templates.

History Timeline Navigator. We developed a third demonstrator of our service to browse an RDF graph combining a local RDF dataset about history linked with DBpedia. The local dataset contains a set of historical events and personalities with dates. The URIs of the resources are those of DBpedia (`fr.dbpedia.org`) in order to link data. Resource descriptions are stored in named graphs which are tagged with topics such as “France” or “Empire”. The `st:cdn` transformation generates an HTML page for each century, where resources are displayed in columns according to the topic of their named graph and in ascending order of dates. Hyperlinks to DBpedia resources are generated which are processed with the former `st:navlab` transformation. Figure 3 is a screenshot of an HTML page generated by the server.



Fig. 3. History Navigator

6 RDF-Syntax to Another-Syntax Transformers

The fourth family of RDF transformations identified in the introduction of this paper gathers the transformations of statements in languages with an RDF syntax into another syntax of the language. In that sense the transformations of this family are completely domain-independent. Here we give two examples of such transformations: the transformation of OWL statements from RDF syntax into OWL Functional syntax and the transformation of SPARQL statements from SPIN/RDF syntax into SPARQL concrete syntax.

6.1 OWL/RDF to OWL/Functional Syntax Transformer

In this section we present the `st:owl` transformation of OWL 2 statements from the OWL/RDF syntax into the OWL 2 Functional Syntax which belongs to this family. The transformation follows the W3C Recommendation *OWL 2 Web Ontology Language Mapping to RDF Graphs*²². We wrote it as a set of 73 STTL templates, structured into 5 subsets, available online²³.

Among them, here is one template enabling to transform OWL/RDF statements presented in Sect. 2.1 in OWL 2 Functional syntax. The template handles the transformation of equivalent classes (and datatype definitions) axiom, possibly involving an intersection or union class expression. For this case, the STTL engine is recursively called on variables `?in` and `?y`.

²² <http://www.w3.org/TR/owl2-mapping-to-rdf>.

²³ <http://ns.inria.fr/sparql-template/owl>.

```

TEMPLATE {
  IF (bound(?t), "DatatypeDefinition", "EquivalentClasses")("
    st:call-template(st:annotate, ?in, owl:equivalentClass, ?y)
    ?in ?y ") }
WHERE {
  ?in owl:equivalentClass ?y
  OPTIONAL { ?y a ?t FILTER(?t = rdfs:Datatype) }}

```

A demo OWL transformation service is available online²⁴.

We validated the `st:owl` transformation on the OWL 2 Primer ontology²⁵ containing 350 RDF triples: we first transformed this set of triples into OWL Functional Syntax with the `st:owl` transformation, then we loaded the output into Protégé and saved it in RDF/XML, then we transformed it again with `st:owl` and we checked that the two transformation outputs were equivalent. Let us note that the results are equivalent but not identical because some statements are not printed in the same order, due to the fact that Protégé does not save RDF/XML statements exactly in the same order and hence blank nodes are not allocated in the same order.

We tested this OWL/RDF transformer on several real world ontologies, among which a subset of the *Galen* ontology. The RDF graph representing it contains 33,080 triples, the size of the result is 0.58 MB and the (average) transformation time is 1.75 seconds. We also have tested our pretty-printer on the *HAO* ontology. The RDF graph representing it contains 38,842 triples, the size of the result is 1.63 MB, the (average) transformation time is 3.1 seconds.

In addition to the transformation of an RDF dataset representing OWL statements, the `st:owl` transformation can also be used when querying an OWL ontology stored in its RDF syntax, to present the results to the user in OWL 2 Functional syntax. This is done by calling in the `SELECT` clause of the query the `STTL` extension functions launching the transformer. As an example, the following query retrieves specific classes and displays them in Functional syntax:

```

SELECT (st:apply-templates-with(st:owl, ?x) as ?t)
WHERE { ?x a owl:Class ; rdfs:subClassOf* f:Human }

```

6.2 SPIN/RDF to SPARQL Concrete Syntax Transformer

In this section we present the `st:spin` transformation of SPARQL queries in SPIN RDF syntax [12], into SPARQL concrete syntax. This transformation of SPARQL statements, belongs to the same family as the transformation of OWL statements: the fourth one identified in the introduction of this paper gathering the transformations of statements in languages with an RDF syntax into another syntax. We wrote the `st:spin` transformation as a set of 64 `STTL` templates which are available online²⁶. Among them, the following template translates a union of alternative graph patterns in SPIN into SPARQL concrete syntax:

²⁴ <http://corese.inria.fr>.

²⁵ <http://www.w3.org/TR/owl2-primer>.

²⁶ <http://ns.inria.fr/sparql-template/spin>.

```
prefix sp: <http://spinrdf.org/sp#> .
TEMPLATE { ?e1 st:nl() "union" st:nl() ?e2 }
WHERE { ?in a sp:Union ; sp:elements (?e1 ?e2)}
```

We validated this transformation on SPARQL 1.1 W3C test suite: we translated each SPARQL query into SPIN by using the Corese library, then we translated it back into SPARQL using the `st:spin` transformation, and finally we executed it with the Corese SPARQL engine.

7 Conclusion and Future Work

In this paper we recalled the STTL principles, we presented the STTL engine and Web service and we demonstrated the support this software provides to programmers by presenting a selection of STTL-based RDF transformers for common languages: RDT-to-Turtle, RDF-to-HTML, OWL/RDF-to-OWL/FunctionalSyntax and SPIN-to-SPARQL/ConcreteSyntax transformations. The source code of this software is freely available within the Corese Semantic Web Factory; it is also available online as a Web service, and all the RDF transformations presented in this paper can be tested online²⁷.

As future work, regarding the performance of our generic transformation rule engine, we intend to improve it by implementing heuristics to optimize the selection of templates. We will also compare in the short term the performance of our generic transformation rule engine with that of existing tools for specific RDF transformations. For instance, we may compare the performance of our engine with that of the parser of the well known OWL API²⁸ for transforming large OWL 2 ontologies from RDF/XML syntax into Functional syntax.

Regarding the exploitation of our generic transformation rule engine to implement RDF transformers into specific languages, we intend to augment the number of STTL transformations available by writing STTL template sets for other formats and domains. In particular, we intend to define an RDF-to-CSV transformation and an RDF-to-JSON transformation. Finally, we will consider a sixth family of RDF transformations gathering RDF-to-RDF transformations for special purposes, e.g., to anonymize RDF datasets.

Acknowledgments. We thank Fuqi Song (Inria), Alban Gaignard (CNRS) and Eric Toguem (U. of Yaoundé, Cameroun) for the setup of the HTTP server.

References

1. Alkhateeb, F., Laborie, S.: Towards extending and using SPARQL for modular document generation. In: Proc. of the 8th ACM Symposium on Document Engineering. ACM Press (2008)

²⁷ <http://corese.inria.fr>.

²⁸ <http://owlapi.sourceforge.net/>.

2. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. *J. Data Semantics* **1**(3) (2012)
3. Bizer, C., Lee, R., Pietriga, E.: Fresnel - a browser-independent presentation vocabulary for RDF. In: *Second International Workshop on Interaction Design and the Semantic Web*, Galway, Ireland (2005)
4. Brophy, M., Heflin, J.: OWL-PL: A Presentation Language for Displaying Semantic Data on the Web. Technical report, Lehigh University (2009)
5. Corby, O., Faron-Zucker, C.: The KGRAM abstract machine for knowledge graph querying. In: *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence*, Toronto, Canada (2010)
6. Corby, O., Faron-Zucker, C.: STTL: A SPARQL-based transformation language for RDF. In: *Proc. of the 11th International Conference on Web Information Systems and Technologies (WEBIST)*, Lisbon, Portugal (2015)
7. Corby, O., Gaignard, A., Faron-Zucker, C., Montagnat, J.: KGRAM versatile data graphs querying and inference engine. In: *Proc. IEEE/WIC/ACM International Conference on Web Intelligence*, Macau (2012)
8. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. Recommendation, W3C (2014)
9. Follenfant, C., Corby, O., Gandon, F., Trastour, D.: RDF modelling and SPARQL processing of SQL abstract syntax trees. In: *Programming the Semantic Web, ISWC Workshop*, Boston, USA (2012)
10. Hawke, S., Polleres, A.: RIF In RDF. Working Group Note, W3C (2012)
11. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. *Semantic Web* **2**(1) (2011)
12. Knublauch, H.: SPIN - SPARQL Syntax. Member Submission, W3C (2011)
13. Patel-Schneider, P.F., Motik, B.: OWL 2 Web Ontology Language Mapping to RDF Graphs, 2nd edn. Recommendation, W3C (2012)
14. Peroni, S., Vitali, F.: RSLT: RDF stylesheet language transformations. In: *Proc. of the ESWC Developers Workshop 2015*, Portoroz, Slovenia (2015)
15. Robie, J., Chamberlin, D., Dyck, M., Snelson, J.: XQuery 3.0: An XML Query Language. Recommendation, W3C (2014)
16. Shapkin, P., Shumsky, L.: A language for transforming the RDF data on the basis of ontologies. In: *Proc. of the 11th International Conference on Web Information Systems and Technologies (WEBIST)*, Lisbon, Portugal (2015)
17. Stolz, A., Rodriguez-Castro, B., Hepp, M.: RDF Translator: A RESTful Multi-Format Data Converter for the Semantic Web. U. der Bundeswehr, Munich, Technical report (2013)

Ontology and Instance Alignment

Multilingual Ontology Mapping in Practice: A Support System for Domain Experts

Mauro Dragoni^(✉)

FBK-IRST, Trento, Italy
dragoni@fbk.eu

Abstract. Ontology mapping is a task aiming to align semantic resources in order to foster the re-use of information and to enable the knowledge and data expressed in the matched ontologies to interoperate. When ontology mapping algorithms are applied in practice, a manual refinement is furtherly necessary for validating the correctness of the resulted resource, especially, as it happens in real-world cases, when a gold standard cannot be exploited for assessing the generated mappings. In this paper, we present a suggestion-based mapping system, integrated as a component of a knowledge management platform, implementing an information retrieval-based (IR-based) approach for generating and validating, by experts, mappings between ontologies. The proposed platform has been evaluated quantitatively (i.e. effectiveness of suggestions, reduction of the user effort, etc.) and qualitatively (i.e. usability) on two use cases: *Organic.Lingua* and *PRESTO*, respectively an EU-funded and regional-funded projects. The results demonstrated the effectiveness and usability of the proposed platform in a real-world environment.

1 Introduction

Ontology mapping is a task aiming to enable interoperability between semantic resources by defining relationships that can be used for various tasks, such as ontology merging, data translation, query answering, or navigation on the web of data. This problem has been widely explored in the last years [1] from the research point of view, and several approaches in the literature have been proposed. However, issues that have to be addressed from the research point of view are amplified if ontologies have to be mapped manually. Indeed, the definition of such mappings is not a trivial task in real-world due to the different structures, contents, and sizes of the ontologies that have to be mapped. Knowledge experts have to deeply analyze the ontologies before creating new mappings, as well as, automatic systems need to take into account a lot of aspects in order to suggest suitable mappings between the concepts of the ontologies. This makes the scenario more complicated by considering, also, that experts knowledge is generally focused on particular domain/s and their level of expertise in knowledge management is, sometimes, inadequate for providing and validating mappings between ontology manually.

For these reasons, it is necessary to provide an instrument that is able to support experts in the mapping activity by providing facilities avoiding them to spend a lot of time for exploring the structure of ontologies, but, at the same time, allow to easily define and validate mappings in a reasonable time.

In this paper, we present a suggestion-based mapping system aiming to improve the mapping activity experience in real-world scenarios. Our system is split in two components: (i) a back-end module implementing an IR-based techniques built with the scope of suggesting sets of candidate mappings, and (ii) a set of user interface facilities that have been integrated in a knowledge management tool for supporting experts in the mapping activity. A further peculiarity of the proposed approach is that it works in a multilingual environment either to support experts in a multi-language collaborative working environment, as well as, to exploit the multilingual information contained in the ontologies for the computation of suggestions. The effectiveness and usability of the proposed platform has been evaluated on two funded projects, described in Section 2, from a quantitative and qualitative points of view.

The paper is structure as follows. Section 2 describes the two use cases where the proposed platform has been adopted and validated. In Section 3, we present the back-end approach used for computing the mapping suggestions; while, in Section 4, we describe the facilities for supporting experts in the mapping task that have been implemented in the used knowledge management tool. Section 5 discusses the evaluation procedure, shows the obtained results, and presents general insights we inferred from this experience. In Section 6, we present a general overview of the literature about ontology mapping and knowledge management tools. Finally, Section 7 concludes the paper.

2 Use Cases

Below, we present the two projects used as test benches for the implemented platform and, for each of them, we introduce the ontologies used in the evaluation.

2.1 PRESTO Project

The objective of the PRESTO (Plausible Representation of Emergency Scenarios for Training Operations) research project is the creation of a system for the customization of serious games scenarios based on virtual reality. The advantage of this system, compared to the state of the art, resides in the richness and the ease of defining the behavior of artificial characters in simulated scenarios, and on the execution engines able to manage cognitive behaviors, actions, and perceptions within a virtual reality environment.

Within the project, ontologies have been used for modeling virtual reality items, actions, behaviors, etc. and the mapping task was needed for defining mappings between the core virtual reality ontology (namely “PRESTO Ontology”) containing general 3d objects and classification schemata describing the set of items contained in 3d-specific libraries. In this use case, we focused the evaluation of the system on

the mapping between the “PRESTO Ontology” and the XVR-library¹ classification schema representing the ontological view of the 3d items modeled in the XVR library. Concepts defined in both ontologies are modeled in English and Italian.

2.2 Organic.Lingua Project

Organic.Lingua (<http://www.organic-lingua.eu>) is an EU-funded project that aims at providing automated multilingual services and tools facilitating the discovery, retrieval, exploitation, and extension of digital educational content related to Organic Agriculture and AgroEcology. More in concrete, the project aims at providing, on top of a web portal, cross-lingual facility services enabling users to (i) find resources in languages different from the ones in which the query has been formulated and/or the resource described (e.g., providing services for the cross-lingual retrieval); (ii) manage meta-data information for resources in different languages (e.g., offering automated meta-data translation services); and (iii) contribute to evolve the content (e.g., providing services supporting the users in the content generation).

The accomplishment of these objectives is reached in the Organic.Lingua project by means of two components: on the one hand, a web portal offering software components and linguistic resources able to provide multilingual services and, on the other hand, a conceptual model (formalized in the “Organic.Lingua ontology”) used for managing information associated with the resources provided to the final users and shared with other components deployed on the Organic.Lingua platform. In a nutshell, the usage of the Organic.Lingua ontology is twofold:

- Resource annotation: each time a content provider inserts a resource in the repository, the resource is annotated with one or more concepts extracted from the ontology.
- Resource retrieval: when web users perform queries on the system, the ontology is used, by the back-end information retrieval system, to perform advanced searches based on semantic techniques.

Concerning the specific ontology mapping task, one of the expected activity was the definition of the mappings between the Organic.Lingua ontology (described by using sixteen languages) and other ontologies related to the agricultural domain, in particular Agrovoc (28 languages) and Eurovoc (24 languages) with the aim of improving either the annotation and retrieval capabilities of the entire platform.

3 The Back-end IR-Based Approach For Multilingual Ontology Mapping

In this Section, we present the first component of the platform, i.e., the back-end system used for storing the structured representation of ontology concepts and for

¹ <http://futureshield.com/xvr-esemble.shtml>

generating and manipulating such representations for suggesting candidate mappings.

Before we present how information are structured, we want to introduce a formalization of what a *mapping* (or “match” or “alignment”) is. A popular definition is to consider a “mapping” as a set of *correspondences* between entities asserting that a certain relation holds between these two entities. Formally, given two ontologies O_1 and O_2 , a match M between O_1 and O_2 is a 5-tuple: $\langle id, e_1, e_2, R, c \rangle$ such that id is a unique identifier of the match, e_1 and e_2 are entities of O_1 and O_2 respectively, R is the matching relation between e_1 and e_2 (for example, equivalence (\equiv), more general (\supseteq), disjointness (\perp)), and c is a confidence measure, typically in the $[0, 1]$ range.

While, the general definition of “mapping” includes different kind of relationships, in this work we focused on realizing a system aiming to suggest only equivalence relationships between concepts.

Below, we described which ontological information have been exploited, how the indexes used for storing ontological information are constructed, and how the messages for requesting and sending the suggestions about candidate mappings are composed.

Exploited Information. The representation of each concept in the system is based on the exploitation of textual information (i.e. the set of “labels”) associated with each concept described in an ontology and its “context”.

For explaining what we mean as “context”, let’s consider the following example. Figures 1 and 2 show excerpts of two ontologies about business processes representing two concepts that might be considered good candidates for defining a new mapping: “Activity” and “Executable Activity”. As “context” of a concept C , we mean the set of concepts connected with C , where C is parent or child of another concept, or where C occurs in the domain or in the range of an object property. In particular, in this work we considered only the first degree of relationships of each concept C .

While, with the term “label”, we mean a string identifying the concept associated with its language tag (i.e. “concept_label@lang_code”) ².

The usage of multilinguality is one of the key-aspects of the concept representation. When a label (independently by the language) is chosen by experts during the creation of an ontology, they implicitly inject in their choice the knowledge about the equivalence of meanings between different translations of each label.

Even if the definition of mapping between ontologies is typically performed on compatible domains ([2]), the use of approaches exploiting label-based techniques, leads to problems with effectiveness. First of all, is that different concepts, especially in case of ontologies representing different domains, could have similar labels without being similar in their meaning. For instance, given two ontologies O_1 and O_2 , where O_1 describes the fruit domain O_2 the fishery one; if we consider the Italian concept “pesca”, we can notice that such a label is polysemic because it can denote

² The format used for the structured representation of thesauri within the system follows the SKOS model <http://www.w3.org/TR/skos-reference/>

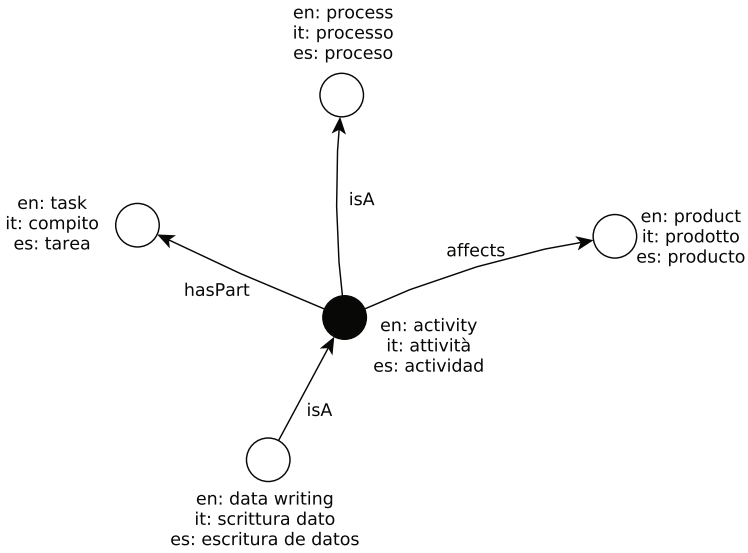


Fig. 1. Example of piece of Ontology 1 showing the context of the concept “Activity”.

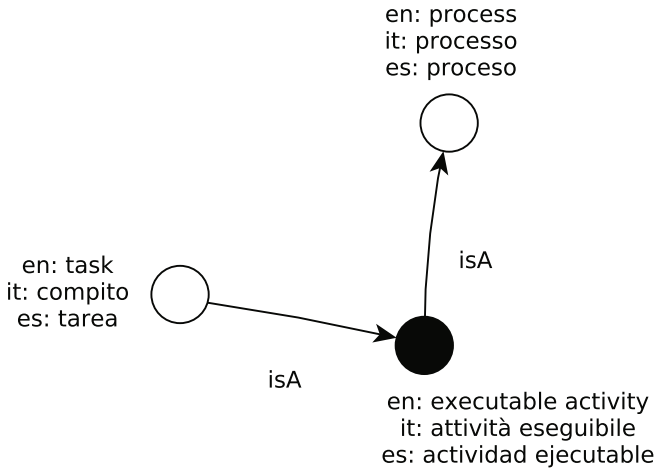


Fig. 2. Example of piece of Ontology 2 showing the context of the concept “Executable Activity”.

a fruit (the peach) in the ontology O_1 and the activity of fishing in the ontology O_2 . If O_1 and O_2 are multilingual, the difference between the two labels can be caught by considering their definition expressed in different languages. Particularly, for the example cited, the English labels “peach” and “fishing”. This aspect helps to solve ambiguities during the computation of new suggestions.

Index Construction. By starting from the information described above, for each concept defined in the ontology, we created a set of triples of the kind

“relation-label-language”, and, in case of synonymy, we may have more triples for the same language. Such labels are tokenized, stemmed³, and normalized by using natural language process libraries⁴.

By taking into account the example shown in Figure 1, information related to the context of the concept “Activity” are the following⁵:

```
[self-prefLabel] "activity"@en
[self-prefLabel] "attività"@it
[self-prefLabel] "actividad"@es
[parent-prefLabel] "process"@en
[parent-prefLabel] "processo"@it
[parent-prefLabel] "proceso"@es
[child-prefLabel] "data writing"@en
[child-prefLabel] "scrittura dato"@it
[child-prefLabel] "escritura de datos"@es
[hasPart-prefLabel] "task"@en
[hasPart-prefLabel] "compito"@it
[hasPart-prefLabel] "tarea"@es
[affects-prefLabel] "product"@en
[affects-prefLabel] "prodotto"@it
[affects-prefLabel] "producto"@es
```

where with the relationship identifier “self”, we indicate labels associated to the current concept description. Subsequently, such information are converted in the structured representation “field_name : value” as shown as follows:

```
self-label-en : activity
self-label-it : attività
self-label-es : actividad
parent-label-en : process
parent-label-it : processo
parent-label-es : proceso
child-label-en : data writing
child-label-it : scrittura dato
child-label-es : escritura de datos
hasPart-label-en : task
hasPart-label-it : compito
hasPart-label-es : tarea
affects-label-en : product
affects-label-it : prodotto
affects-label-es : producto
context-label-en : activity process data writing task product
context-label-it : attività processo scrittura dato compito prodotto
context-label-es : actividad proceso escritura de datos tarea producto
```

The fields “relationship-label-XX” are automatically generated during the creation of the record and they contain all label of the concept context. Such fields are useful when, as it will be shown in the explanation of how mappings are suggested, two concepts within the ontologies are linked with different relations.

³ The list of languages supported by the used stemming algorithm are Italian, French, German, English, Greek, Spanish, Portuguese, Polish, Dutch, Norwegian, Hungarian, Swedish, Latvian, Turkish, Czech, Russian, Romanian, Bulgarian, Arabic, Hindi, Chinese, Japanese, Danish, Finnish, Armenian, Indonesian, Thai.

⁴ The text processors included in the Lucene (<http://lucene.apache.org>) library have been used. In case of unavailability of libraries for a particular language, the original label is indexed as it is without being processed.

⁵ SKOS notation is used.

After the construction of the structured representation for all concepts, they are indexed as documents by using the inverted index algorithm [3]. This operation is performed for each ontology stored in the platform with the result of having, for each ontology, a dedicated index. Indexes are stored in an architecture based on Apache Solr⁶.

Matches Definition. Once the indexes are created, the suggestion of candidate mappings is done by performing queries using information extracted from the concept used as starting point for the mapping operation. Such a query is created by building a structured representation compliant with the one described above by using information defined in the concept used as starting point.

Therefore, similarly to the creation of the indexed records, by taking into account the example shown in Figure 2, the query for the concept “Executable Activity” is built as follows, firstly we extract all information related to the concept:

```
[self-prefLabel] "executable activity"@en
[self-prefLabel] "attività eseguibile"@it
[self-prefLabel] "actividad ejecutable"@es
[child-prefLabel] "task"@en
[child-prefLabel] "compito"@it
[child-prefLabel] "tarea"@es
[parent-prefLabel] "process"@en
[parent-prefLabel] "processo"@it
[parent-prefLabel] "proceso"@es
```

and, then, we create query:

```
proc(self-label-en:"executable activity" OR self-label-it:"attività eseguibile" OR
self-label-es:"actividad ejecutable" OR child-label-en:task OR
child-label-it:compito OR child-label-es:tarea OR
parent-label-en:process OR parent-label-it:processo OR
parent-label-es:proceso OR
context-label-en:"executable activity task process" OR
context-label-it:"attività eseguibile compito processo" OR
context-label-es:"actividad ejecutable tarea proceso")
```

where `proc()` is the function representing the set of textual preprocessing activities performed on the terms contained in the query.

Here, we may see that in this ontology, the concept “Task” is represented as child of “Executable Activity”; while in the ontology used as example for the index construction, the concept “Task” is represented as part of the concept “Activity”. In scenarios like these, the use of “relationship-label-XX” fields is important for avoiding loss of information during the query phase.

When the back-end component receives the request for suggesting new mappings, it performs a search operation in the index. As a result, a rank ordered by confidence score is produced and returned by the system. Such a score is computed by applying the scoring function shown in the Equation 1.

$$\text{score}(R_{c_1}, R_{c_2}) = \text{coord}(R_{c_1}, R_{c_2}) \cdot \sum_{x \in R_{c_1}} (\text{tf}(x \in R_{c_2}) \cdot \text{idf}(x))^2 \quad (1)$$

⁶ <http://lucene.apache.org/solr/>

where R_{c_1} and R_{c_2} are respectively the representations of concepts defined in the source and in the target ontologies, $tf()$ and $idf()$ are the standard “term frequency” and “inverse document frequency” functions used in the IR field [3], and $coord(R_{c_1}, R_{c_2})$ represents the number of label defined in the representation of c_1 occurring in the representation of c_2 . The implemented version of the system returns the five candidate mappings with the highest score.

4 User Facilities of the Knowledge Management Tool

The back-end component described in the previous Section can be accessed through the user facilities that have been integrated as extensions of the MoKi [4] tool.

Figure 3 shows the process about how the entire suggestion-base mapping service works.

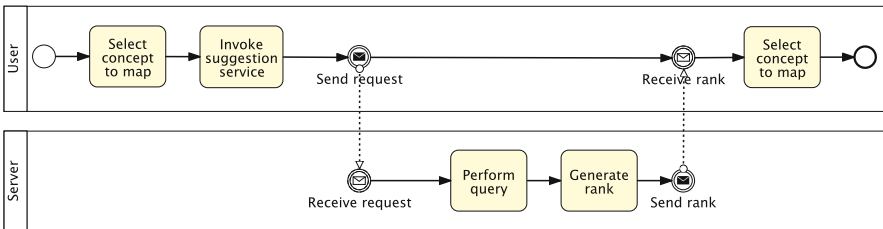


Fig. 3. Process describing the message exchange between the client and server sides

Once the expert selects the concept to map, a request message is sent to the server. Such a message contains the representation of the current concept and its context, structured as described in Section 3, and codified by using the JSON format. The server consumes the request and it generates a rank containing the five suggestions ordered by their confidence score. The rank is then encoded in JSON and it is sent back to the user interface that will show it to the user. Screens concerning the user facilities involved in this process are shown below.

For managing the mappings, a dedicated section in the concept modeling page as been integrated as shown in Figure 4. Here, the expert is able to see which are the concepts that have been already mapped with the current one and to decide if to maintain such mappings or to remove them. For creating a new mapping, the expert has to choose which ontology to use for requesting mapping suggestions, and then to click on the “Add New Mapping” button for invoking the suggestion service.

When the request is sent, on the background the structured representation of the current concept is converted into a query (as described in Section 3) which is performed on the index containing the concepts of the ontology specified by the expert. When the rank of the suggestions is composed, it is proposed to the expert as shown in Figure 5. For each suggestion, ordered by confidence score, the expert



Fig. 4. User facility for invoking the suggestion service.eps

is able to open the concept description page (if available) by clicking on the concept URI, and to eventually define a new mapping by clicking on the “Create Mapping” button.

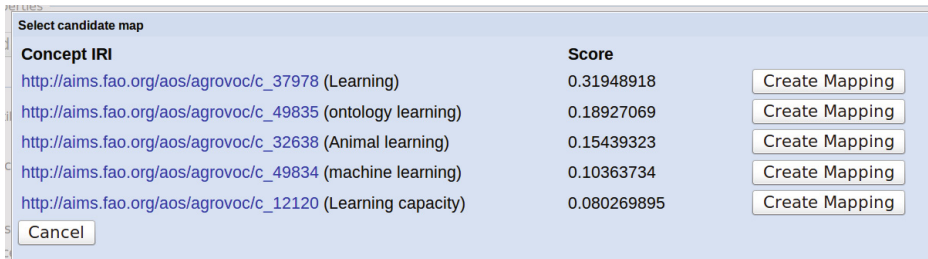


Fig. 5. Example of rank produced by the IR system containing the five suggestions for mapping the concept “Learning” defined in the Organic.Lingua ontology with a concept coming from the Agrovoc one.eps

Besides this, in a separated module, experts are able to upload new ontologies and creating the related indexes. Figure 6 shows the interface used by the experts for uploading a new ontology in the repository. This facility allows to convert the uploaded ontology in the structured representation described in Section 3 and to store it in a dedicated index. From this interface, experts are able to manage the ontologies already stored in the repository by viewing some basic information about them and, eventually, to delete one or more ontologies.

For adding a new ontology to the repository, experts have to select the file containing the ontology, write a description, decide an acronym for referring the ontology in the other sections of the tool, and, finally, press the “Save” button.

5 Evaluation

The presented platform has been evaluated in order to understand if the proposed suggestion-based mapping service provides an effective support to the mapping activity and if the facilities designed for supporting such an activity have been judged usable by the experts.

In detail, we are interested in answering two main research questions:

Ontology Manager

Number of loaded ontologies: 2

| ID | Ontology Namespace | Ontology Acronym | Description | | |
|----|----------------------------------|------------------|-------------------|---------|--------|
| 1 | http://aims.fao.org/aos/agrovoc/ | Agrovoc | Agrovoc ontology. | Explore | Delete |
| 2 | http://eurovoc.europa.eu/ | Eurovoc | Eurovoc ontology. | Explore | Delete |

Add a new ontology to the repository

| Namespace | Acronym | Description | File |
|----------------------|----------------------|----------------------|--|
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="button" value="Browse..."/> No file selected. <input type="button" value="Save"/> |

Fig. 6. Interface used for loading a new ontology within the system.

RQ1 Does the proposed approach provide an *effective* support, in terms of concept suggestions, to the mapping of multilingual ontologies?

RQ2 Are the provided interfaces *usable* for supporting the experts in the mapping activity between ontologies?

In order to answer these questions, we performed two types of analysis:

1. Quantitative: we collected objective measures concerning the effectiveness of the mappings suggested by the implemented approach and by providing a comparison between the time needed to map the resources manually, with respect to the time needed to define the mappings with our platform.
2. Qualitative: we collected subjective judgments from the experts involved in the use cases that have been asked to evaluate the general usability of the components and to provide feedbacks for future actions.

Concerning the qualitative evaluation of the suggestion-based mapping service, experts were asked to fill a questionnaire aiming at investigating their perception about the mapping process through the usage of the MoKi tool. Questions were organized in three parts: (i) one collecting information on the experts' background; (ii) one on the subjects' evaluation about MoKi and the usability of its different functionalities for accomplishing the mapping task; and (iii) a last one for retrieving impressions, and questions related to the work performed for the definition of new mappings between ontologies. Some of the questions were provided in the form of open questions, while most of them were closed questions. The latter type mainly concern the experts' evaluation of the tool usefulness on a scale from 1 to 5, varying according to the target of the evaluation (e.g., 1 = *extremely ease/useful/effective*, ... , 5 = *extremely useless/difficult/ineffective*).

It is important to highlight, that all experts were already familiar with the main functionalities of the MoKi tool, therefore, opinions about such functionalities were not collected through the questionnaire.

In Sections 5.1 and 5.2, we provide the description about how the evaluation on the two use cases has been organized; Section 5.3 presented the evaluation results, while in Section 5.4, we wrap up the consideration about what we experienced during the evaluation procedure.

5.1 Use Case 1: The **Organic.Lingua** Project

In the **Organic.Lingua** use case, six experts in the agricultural domain have been involved for defining the mappings between the **Organic.Lingua** ontology and the **Agrovoc** and **Eurovoc** ones. The **Organic.Lingua** ontology contains 291 concepts and it is focused on organic agriculture and sustainable agricultural processes. All concepts have been translated in 16 languages. The **Agrovoc** ontology is the wider available resource concerning food, nutrition, agriculture, fisheries, forestry, and environment. It is composed by around 32,000 concepts translated in 28 languages. While, the **Eurovoc** ontology is focused on modeling the general terminology used by the EU government including concepts related to agricultural policies that have to be adopted in the EU territory. It contains around 7,000 concepts translated in 24 languages.

Experts have been divided in two groups: three experts defined the mappings manually by using a well-known ontology editor tool (i.e. *Protégé* [5]); while, the other three, used the **MoKi** tool interface. The mapping operation has been performed in two steps: firstly, each expert mapped the **Organic.Lingua** ontology with the **Agrovoc** and **Eurovoc** ones independently by using the assigned tool; secondly, for cases where a full agreement were not found, experts discussed about the right mapping to provide. After the conclusion of the mapping activity, also the first group of experts performed the mapping between the ontologies by using the **MoKi** tool in order to allow them to compile the questionnaire about the tool.

First two rows of Table 1 contain information about the number of mapped concepts between the **Organic.Lingua** ontology and the **Agrovoc** and **Eurovoc** ones; while, rows 4 and 5 contain the measure of the inter-annotator agreement computed for each ontology. The averaged response times of the back-end component to requests performed by experts are shown in rows 7 and 8, respectively, for the **Agrovoc** and **Eurovoc** ontologies.

Finally, rows 1 and 2 of Table 2 show the results concerning the effectiveness of the ranks produced by the suggestion service, and the comparison of the time effort needed by the experts for completing the mapping operation.

5.2 Use Case 2: The **PRESTO** Project

In the **PRESTO** use case, three experts (that for this use case have an ontology engineering profile) have been involved for mapping the **PRESTO** core ontology with the classification schema extracted from the **XVR 3d-library**. The core **PRESTO** ontology contains 311 concepts described in 2 languages and it is focused on describing general items that can be used in virtual reality scenarios. The **XVR** classification schema contains around 1200 descriptions of 3d elements available in the environment defined in the **XVR** framework. Such elements provide descriptions in Italian and English languages.

For this use case, one expert has been assigned to the manual mapping task; while the other two performed the mapping activity by using the **MoKi** tool. Also in this case, all experts performed the mapping operation independently and, in case of disagreement, they discussed about the right mapping to write in the final

Table 1. Information about the number of mapped concepts, the measured inter-annotator agreement, and the average response time of the back-end component to mapping suggestion requests.

| # | Indicator | Value |
|---|---|--------|
| 1 | Number of concepts mapped with Agrovoc: | 161 |
| 2 | Number of concepts mapped with Eurovoc: | 94 |
| 3 | Number of concepts mapped with the XVR classification schema: | 285 |
| 4 | Inter-annotator agreement on Agrovoc mappings | 94.41% |
| 5 | Inter-annotator agreement on Eurovoc mappings | 97.87% |
| 6 | Inter-annotator agreement on the XVR classification schema | 97.54% |
| 7 | Average response time for querying the Agrovoc repository (seconds) | 1.27 |
| 8 | Average response time for querying the Eurovoc repository (seconds) | 0.94 |
| 9 | Average response time for querying the XVR repository (seconds) | 0.91 |

ontology. Moreover, the expert assigned to the manual mapping, performed the mapping activity also by using the MoKi tool in order to allow her to compile the evaluation questionnaire.

The number of mapped concepts of the PRESTO ontology is shown in the third row of Table 1, with the related inter-annotator agreement (row 6) and the average response time of the back-end component to the expert requests (row 9). While, the ranks effectiveness and the comparison of the time effort are shown in the third row of Table 2.

5.3 Quantitative and Qualitative Results

The registered average response time satisfies the experts need of being able to work quickly concerning the definition of new mappings. Indeed, by having an average response time around one second avoid downtime for users during the usage of the platform, aspect that has to be taken into account when web-based systems are used.

Concerning the analysis of the effectiveness results, in Table 2 we evaluate the precision of the ranks provided to experts by adopting standard information retrieval measures. In particular, here we evaluated if the correct mapping was placed respectively at the top of the rank, in the first three positions, or if it was at least provided in the rank (i.e. $Prec@5$, that in this case coincides with the Recall values). Results demonstrated the effectiveness of the back-end approach used for providing the suggestions to the experts. Indeed, almost in all cases the correct suggestions was presented to the users; moreover, encouraging results were obtained by observing the precision related to the top suggestion. This aspect allows to state that, in general, an approach based on IR techniques is a good direction for addressing the ontology mapping problem.

By considering the reduction of the time effort, we may see how the usage of the platform allows to strongly reduce the time necessary for completing the mapping between the ontologies, with a peak reduction of almost 75% of time. This result is

Table 2. Precision and recall values concerning the effectiveness of the suggestion retrieval system and the comparison (tool supported vs. manual) of the time effort needed by the experts for completing the mapping operations.

| Mapped Ontology | Suggested Ranks | | | | Time Effort Comparison (minutes) | | |
|-----------------|-----------------|--------|--------|--------|----------------------------------|----------------|----------------|
| | Prec@1 | Prec@3 | Prec@5 | Recall | Avg. Manual | Avg. With MoKi | Difference (%) |
| Agrovoc | 0.81 | 0.91 | 0.97 | 0.97 | 193 | 49 | -74.61% |
| Eurovoc | 0.90 | 0.94 | 0.98 | 0.98 | 174 | 45 | -74.14% |
| XVR | 0.85 | 0.97 | 1.0 | 1.0 | 197 | 67 | -65.99% |

Table 3. MoKi functionality effectiveness in supporting the mapping activity

| Ontology Loading | Mapping Management | Mapping Browsing |
|------------------|-----------------------------|--|
| <i>Effective</i> | <i>Absolutely effective</i> | <i>Neither effective nor ineffective</i> |

important for demonstrating the viability of the proposed platform for implementing it in real-world environments.

Concerning the qualitative evaluation, from the results collected through the questionnaire fulfilled by experts, in order to evaluate the statistical significance of the positivity/negativity of the collected results we applied the (one-tailed) Mann-Whitney test [6] verifying the hypothesis that $\tilde{F} \leq 3$, where \tilde{F} represents the median of the evaluations for the factor F and 3 is the intermediate value in the 1 to 5 Likert scale. All the analyses are performed with a level of confidence of 95% (p-value < 0.05), i.e., there is only 5% of probability that the results are obtained by chance.

To better understand the relationship between the role of the tool in supporting the experts during the mapping activity, we asked them to express their evaluation about the effectiveness of the support provided by each functionality. Table 3 reports the corresponding evaluations.

The results show, in general, a good perception of the implemented functionalities, in particular concerning the procedure of defining a new mapping in the ontology. However, the browsing facility (i.e. the possibility of opening the description page of suggested concepts) did not convince the experts that asked for a graphical support able to quickly show the context of the suggested concepts. Indeed, they object about the fact that sometimes external pages describing suggested concepts might not be available and that the consult of the concept description might be time-consuming if context information are not clearly explained in the suggested concept pages.

5.4 Findings and Lesson Learned

The quantitative and qualitative results demonstrated the viability of the proposed platform in real-world scenarios and, in particular, its effectiveness in the proposed use cases. Therefore, we can positively answer to both research questions, **RQ1**: the back-end component provides effective suggestions for performing the mapping

activity, and **RQ2**: the provided interfaces are usable and useful for supporting the experts in the mapping activity.

Besides these, there were other insights, either positive and negative, emerged during the subjective evaluation that we conducted.

The main positive aspect highlighted by the experts was related to the easy and quick way of defining a new mapping with respect to other available knowledge management tools (see details in Section 6) due to the missing, in them, of specific support for the mapping activity. The suggestion-based mapping service allowed to strongly reduce the amount of the most time-consuming activities, i.e., the navigation through the ontologies for analyzing candidate concepts for mappings. Indeed, while for relatively small (or less-deeper) ontologies, it is quite easy to detect which is the branch of the ontology containing candidate mappings. However, the same is not true for big (or high-deeper) ontologies where there is a significant time-overhead just for reaching the potential mapping candidates. Moreover, the ontology browsing aspect increases when the description of concept becomes more complex, i.e., when many relationships are modeled. In these cases, the visualization features implemented in knowledge management tools should be able to allow to show the context of each concept quickly. By adopting the proposed system, this problem can be avoided because the score computed for each suggestion already takes into account all relationships between the suggested concept and its directly connected ones.

However, even if from one side these suggestions effectively help the work of experts, they have been seen by them as a black box and, sometimes, when more than one suggestions are good candidates for defining a new mapping, experts requested to have a more immediate view of the context of the suggested concepts. As seen in Figure 5, the interface provides the possibility of opening the actual page containing details about the suggested concepts. This facility has been considered improvable by the experts for two reasons: (i) it may happen that the target page is not available, by blocking the work of the experts; and (ii) from the content of the target page may be not immediate to understand which concepts are in relationship with the suggested one and which kind of relationships they have. Experts proposal was the implementation of a graphical support showing the context (or a portion of the entire ontology branch) of each suggestion in order to a more clear picture of the “area” where the candidate mapping is placed. This feature has been judged valuable for improving the general overview of each suggested concept.

Finally, the only clearly negative aspect raised by the experts was the difficult to define a new mapping when the list of the suggested concept does not contain correct suggestions. Indeed, in this case, experts are not able to navigate through other suggestions unless they decide to open the ontology with the ontology management tool and start to navigate through it. This issue will be addressed by implementing the possibility of navigating through further set of suggestions that is a hypothesis that will be discarded in the beginning just for trying to find a good compromise between the number of suggestions and their effectiveness in order to avoid the consultation of a lot of suggestions by the experts.

6 Related Work

In this work either ontology mapping approaches, in a multilingual fashion, and knowledge management tools have been mentioned.

Literature about ontology mapping is very large and many systems and algorithms have been proposed. Surveys offering different perspectives can be found in [1] [7]. Concerning the use of multilinguality, research started to take it into account in the last fifteen years. First efforts on building and mapping multilingual ontologies have been conducted by using WordNet [8] [9] which kicked off several projects focused on the creation of different language versions of WordNet.

The two most significant ones are EuroWordNet and MultiWordNet. These projects adopted two different models of multilinguality: the one adopted in the EuroWordNet project [10] (EWN) consists in building language-specific wordnets independently from each other, and trying in a second phase to find correspondences between them. While the one adopted in the MultiWordNet project [11] (MWN), consists in building language-specific wordnets while keeping them aligned as much as possible to the synsets and semantic relations available in the Princeton WordNet (PWN).

After these, multilingual ontology mapping has been applied on several problems, as described in [12], where the authors address the problem of building conceptual resources for general multilingual applications. Examples of application fields in which multilingual ontology mapping has been applied are cross-language information retrieval [13], folksonomies [14], and specific domains-based applications [15] [16] [17] [18].

Concerning knowledge management tool, a lot of software born in the last decade supporting in different ways the modeling and knowledge sharing activities.

*Knoodl*⁷ facilitates community-oriented development of OWL based ontologies and RDF knowledge bases. It also serves as a semantic technology platform, offering a Java service-based interface or a SPARQL-based interface so that communities can build their own semantic applications using their ontologies and knowledge bases.

Protégé [5] is an open source visual ontology editor and knowledge-base framework. Recently, Collaborative *Protégé* has been released as an extension of the existing *Protégé* system. It supports collaborative ontology editing as well as annotation of both ontology components and ontology changes. In addition to the common ontology editing operations, it enables annotation of both ontology components and ontology changes. It supports the searching and filtering of user annotations, also known as notes, based on different criteria.

NeOn [19]. It is a state-of-the-art, open source multi-platform ontology engineering environment, which provides comprehensive support for the ontology engineering life-cycle. The last version of the toolkit is based on the Eclipse platform and provides an extensive set of plug-ins covering a variety of ontology engineering activities.

⁷ <http://www.knoodl.com>

While all these tools effectively support the ontology modeling activity, none of them provide specific facilities for supporting the mapping task in an easy and quick way, besides the classic axiomatization of the mapping relations, a discussion about the tool support for ontology mapping is presented in [20]. Indeed, as discussed in the previous section, the definition of a new mapping requires time-consuming activities by the experts that could be avoided by using facilities like the ones implemented in the discussed version of the MoKi tool.

7 Conclusions

In this paper, we presented a suggestion-based mapping service for supporting the ontology mapping problem in real-world scenarios with an emphasis on the importance of exploiting multilingual information provided by ontological models. Our platform, composed by an IR-based approach integrated in a knowledge management tool, namely MoKi, has been presented and its effectiveness and usability concerning the mapping process have been discussed. The platform has been quantitatively and qualitatively evaluated on two use cases, the *Organic.Lingua* and *PRESTO* projects, by demonstrating the usability and the effectiveness of the proposed suggestion-based mapping service. Therefore, by starting from the obtained results, the proposed system will be implemented in further uses cases in order to extend its overall evaluation. Finally, future work on the platform will be driven by the inferred lesson learned with the aim of improving either the quality and the usability of the entire system.

References

1. Euzenat, J., Shvaiko, P.: *Ontology matching*, 2nd edn. Springer-Verlag, Heidelberg (DE) (2013)
2. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. *SIGMOD Rec.* **35**(3), 34–41 (2006)
3. van Rijsbergen, C.J.: *Information Retrieval*, Butterworth (1979)
4. Ghidini, C., Rospocher, M., Serafini, L.: Modeling in a wiki with moki: Reference architecture, implementation, and usages. *International Journal On Advances in Life Sciences* **4**(3&4), 111–124 (2012)
5. Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubézy, M., Eriksson, H., Noy, N., Tu, S.: The evolution of protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.* **58**(1), 89–123 (2003)
6. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers (2000)
7. Bellahsene, Z., Bonifati, A., Rahm, E. (eds.): *Schema Matching and Mapping*. Springer (2011)
8. Fellbaum, C.: *WordNet: An Electronic Lexical Database*. Bradford Books (1998)
9. Daudé, J., Padró, L., Rigau, G.: Mapping multilingual hierarchies using relaxation labeling. *CoRR cs.CL/9906025* (1999)
10. Vossen, P.: Introduction to eurowordnet. *Computers and the Humanities* **32**(2–3), 73–89 (1998)

11. Magnini, B., Strapparava, C., Ciravegna, F., Pianta, E.: Multilingual lexical knowledge bases: applied wordnet prospects. In: Proceedings of the International Workshop on The Future of the Dictionary (1994)
12. Dorr, B.J., Levow, G.-A., Lin, D.: Building a chinese-english mapping between verb concepts for multilingual applications. In: White, J.S. (ed.) AMTA 2000. LNCS (LNAI), vol. 1934, pp. 1–12. Springer, Heidelberg (2000)
13. Zhang, L., Wu, G., Xu, Y., Li, W., Zhong, Y.: Multilingual collection retrieving via ontology alignment. In: Chen, Z., Chen, H., Miao, Q., Fu, Y., Fox, E., Lim, E. (eds.) ICADL 2004. LNCS, vol. 3334, pp. 510–514. Springer, Heidelberg (2004)
14. Jung, J.J.: Matching multilingual tags based on community of lingual practice from multiple folksonomy: a preliminary result. In: García-Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J.M., Ali, M. (eds.) IEA/AIE 2010, Part II. LNCS, vol. 6097, pp. 39–46. Springer, Heidelberg (2010)
15. Galatescu, A.: Multilingual semantics in the request-service matchmaking. In: ECOWS, pp. 275–286. IEEE Computer Society (2006)
16. Stock, K., Cialone, C.: An approach to the management of multiple aligned multilingual ontologies for a geospatial earth observation system. In: Claramunt, C., Levashkin, S., Bertolotto, M. (eds.) GeoS 2011. LNCS, vol. 6631, pp. 52–69. Springer, Heidelberg (2011)
17. Jiun Chen, S., Hua Chen, H.: Mapping multilingual lexical semantics for knowledge organization systems. *The Electronic Library* **30**(2), 278–294 (2012)
18. Jung, J.J.: Exploiting multi-agent platform for indirect alignment between multilingual ontologies: A case study on tourism business. *Expert Syst. Appl.* **38**(5), 5774–5780 (2011)
19. 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)
20. Falconer, S.M., Noy, N.F.: Interactive techniques to support ontology matching. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds.) Schema Matching and Mapping. Data-Centric Systems and Applications, pp. 29–51. Springer (2011)

Data Access Linking and Integration with DALI: Building a Safety Net for an Ocean of City Data

Vanessa Lopez^(✉), Martin Stephenson, Spyros Kotoulas, and Pierpaolo Tommasi

Smarter Cities Technology Centre, IBM Research, Dublin, Ireland

{vanlopez,martin_stephenson,spyros.kotoulas,ptommasi}@ie.ibm.com

Abstract. DALI is a practical system that exploits Linked Data to provide federated entity search and spatial exploration across hundreds of information sources containing Open and Enterprise data pertaining to cities, which are stored in tabular files or in their original enterprise systems. Our system is able to lift data into a meaningful linked structure with explicit semantics, and support novel contextual search and retrieval tasks by identifying related entities across models and data sources. We evaluate in two pilot scenarios. In the first, data-engineers bring together public and enterprise datasets about public safety. In the second, knowledge-engineers and domain-experts, build a view of health and social care providers for vulnerable populations. We show that our approach can re-use data assets and provides better results than pure text-based approaches in finding relevant information, as well as satisfying specific information needs.

1 Introduction

Smart City applications rely on large amounts of data retrieved from sensors, social networks, or government authorities. Such information is often published in open data portals to promote transparency and enable innovation, as well as inviting a large community to explore how new insights can be derived from existing datasets and their combinations. For example, the NYC data platform [19] allows users to explore datasets through keyword search or by navigating through their catalogues.

Cities need to exploit this valuable resource in combination with data from their existing enterprise systems. Open data is often published in the form of tabular data, with little or incomplete schema information, while enterprise applications typically rely on complex relational schemas. There is a clear need to make city-specific information easy to consume and combine at low cost, but this proves a difficult task. To fulfill the potential of exploiting large volumes of data and obtain insights, in response to complex information needs, the following challenges are to be tackled:

- *Data Discovery.* How to discover datasets and facts for different user tasks, given the complexity of the domain, extreme heterogeneity, diversity of the data, lack of a priori defined schemas, and poor semantic catalogues.
- *Data Integration.* How can data be understood in order to uncover relationships, in face of a dynamic and open environment, the infeasibility of creating a single model to cover the entire domain and the poor scalability of N-to-N integration approaches.

- *Data Exploitation*. How to create actionable views to provide relevant insights across all data sources, for a broad set of tasks, with minimal user effort?

In this ocean of data, Linked Data technologies can improve interoperability and discoverability of datasets by reusing standard vocabularies, linking to external sources, as well as enabling richer querying [5][17]. In this paper, we present and evaluate DALI, a system that puts together existing semantic techniques to offer a lightweight and incremental information sharing approach, on top of heterogeneous enterprise city data and selected well-formed open data in tabular form, as well as an end-user application, to search and consume city data online. Our contributions are:

- *Open Distributed Modeling*. Organizations can expose structured and semi-structured information based on their models (ontologies) of choice. The system ingests and integrates data in an incremental manner, lowering the entry cost by importing datasets as they are, and mapping them to other sources as needed.
- *Web of Data Integration*. By lifting data to existing models and exploiting overlap across ontologies, hidden links across entities are uncovered, in response to user searches or explorations in the context of an existing dataset. Also, using LOD URIs as target vocabularies enables to uniquely identify and organize topics and to access more information about them when needed, fully reusing the Web-wide wealth of resources.
- *Fit-for-Use*. Search and exploration interfaces allow users to profit from the expressive power of semantic standards, answering to complex information tasks, while hiding the complexity behind the data representation and services exposed.

This paper is structured as follows. Two motivating scenarios, not currently addressed without the adoption of semantics, and our approach are presented in Section 2. The architecture and components for lifting, mining annotations and contextual retrieval across distributed sources are presented in Section 3. Experimental evaluation, discussion and our position against related work are presented in Sections 4 and 5.

2 Motivating Scenarios and Approach

We present DALI in the context of two representative industry scenarios, driven by IBM solutions, that require tackling the *discovery-integration-exploitation* challenges discussed above. The first is to allow data-engineers using IBM Intelligent Operation Center (IOC) [11] enhance enterprise data with open data. The second is to support data-engineers build a Safety Net of health and social care providers and community services from public sources. This Safety Net can be used to support care workers finding services targeted to vulnerable populations in a city, and to create personalized care plans based on patient needs, in the context of IBM Cúram [13].

Scenario 1. Enterprise data obtained from IBM IOC in Minneapolis, and stored in IBM DB2 relational tables, is enriched with relevant open city data, which comes in the form of spreadsheets made for consumption by humans. The enterprise data, pertaining to events in the city, describes, among others, a point in time, a location, and a type (e.g., police calls reporting different incidents, events in an stadium like a lost child or a spec-

tator requiring medical assistance, licenses granted to establishments, etc.). These events from multiple sources can be visualized in a map or a dashboard.

Consider a field worker concerned about safety issues in the city. In order to prioritize onsite inspection, she has access to an enterprise dataset about police call outs related to safety issues, that she would like to overlay with other open datasets with relevant information, such as places with higher average rates of crime. She may also be interested in the location of hospitals near the places generating most ambulance and emergency call outs, or with higher response times. In this scenario, users explorations have a geographical focus and relevant links are uncovered on-demand. The required information is coming from different domains and sources that one can typically find in data-hubs for a given city (covering domains such as health, environment, public safety, education, recreation, etc.). In particular, we included datasets from geocommons.com, as well as some national-level datasets for the USA.

Scenario 2. Integrated Care aims at improving patient care by coordinating social and health care services for the vulnerable populations like the elderly or homeless [15]. We exploit open data to build a Safety Net knowledge graph to support care for seniors and their families in New York City. A Safety Net is a foundational and extensible “map” of all known care services, their characteristics and connections for a target vulnerable population and area. To build a Safety Net we can bring in multiple resources for social and health related services published in the NYC data portal [19], Geocommons, as well as Medicare public data on quality of care across the U.S. [20]. Starting with this siloed, noisy, heterogeneous data with no single schema, entities are automatically extracted to understand the type of organization (e.g., hospitals, pharmacies, day care centers, meal delivery services, etc.), the services they offer and other attributes – such as contact details, opening hours, private or non-profit-, or particular attributes for each provider across the different datasets – like readmission rates and main specialities for hospitals (cardiology, neurology, etc.), or targeted populations (families, youth, immigrant, elderly) for community resources.

Existing models are used to annotate, catalog and link the data, therefore mapping each entity into a linked Safety Net graph. Data services are provided on top to create views, with information about providers and services available. Experts can then search, select and constrain the relevant criteria (facets) for each provider, like estimated cost, distance or hospital ratings for given medical speciali-

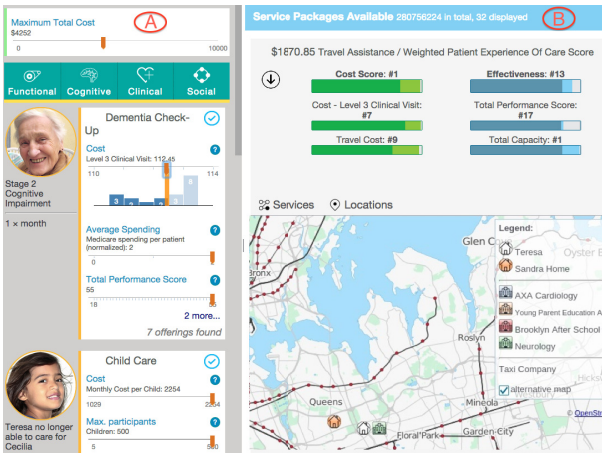


Fig. 1. Use case demonstrator for social care planning

ties. These views can be exported in a JSON format to feed analytics and optimization algorithms to calculate the optimal services and plans, according to the combination of the various criteria and constrains on providers specified by the care team. Fig.1.A shows a screenshot on the imported views with the providers and attributes relevant to care for two needs of a given patient and her family. In this example, the selection of hospitals that care for dementia is based (among others not shown) on their average spending and total performance score, as stated across different Medicare datasets. The criteria for childcare providers are based on a cost range and max. number of participants. In Fig.1.B, one can see the combination of providers and their locations on the map, in the context of a planning component (that is beyond the scope of this paper).

The first scenario motivated the development of the system, and it is used to show-case examples in the rest of the paper. The second scenario shows that the system is not tailored to a particular use case. The resulting consolidated, multi-faceted, linked information is used to bootstrap search and exploration and expose it to users, moving from catalogue-based content management to searching and querying for entities and their relations across sources, aggregating information into customized views.

We propose a data-centric approach that consists of 3 steps as explained next. Firstly, raw tabular data is ingested and semantically lifted. Secondly, the entities and relations are automatically annotated and aligned to well-known vocabularies and widely used Linked Open Data (LOD) resources. Thirdly, different spatial views and exploration paths are exposed according to dynamically chosen models, other related datasets, and interaction paradigms, such as keyword and faceted search.

3 Architecture and Components

We present a flexible architecture (Fig. 2), in which the following functionality and main contributions are exposed:

1. *Distributed data ingestion and virtualization (Data Server)*. Enterprise relational data and tabular open data files are accessed and exposed as virtualized RDF via SPARQL end points. The distributed nature of RDF allows access to linked information across silos and from different agencies. An initial semantic uplift is done at this point, to identify entities, labels, *datatypes*, and geo-temporal data.
2. *Identification and semantic uplift of entities from open and enterprise data to an open set of specified ontologies (Application Server)*. External LOD sources and ontologies are used to annotate the data, providing meaning, context and links across sources and entities exposed from open data.
3. *Contextual information retrieval (RESTful APIs)*. Efficiently retrieve entities based on space and semantics relatedness, given a user query or through explorations of related entities within some geographic proximity. Functionality is exposed as RESTful APIs for easy developer consumption (no semantic knowledge required).

The **Data Server** component abstracts from the infrastructure of each source, the information is accessed from distributed sources as RDF by exposing virtualized

SPARQL end points. The **Application Server** component then accesses the exposed SPARQL end points to extract semantic annotations (using the reference ontologies) and schema information. These annotations and schema are stored in a centralized context store based on Jena TDB, where different graphs are created and associated for each distributed source to keep provenance. The context store is indexed using LARQ [16] that enables to perform text searches on all labels, as part of SPARQL queries.

This architecture allows for incremental integration. New datasets, reference ontologies (annotation sources) can be configured and added at any time. The system automatically lifts, exposes and annotates new datasets, or if new reference ontologies are added, the system aligns each data source with the new models, adding the new annotations in the context store. Multiple data repositories are maintained and queried in a federated manner using the REST services to exploit DALI semantic capabilities.

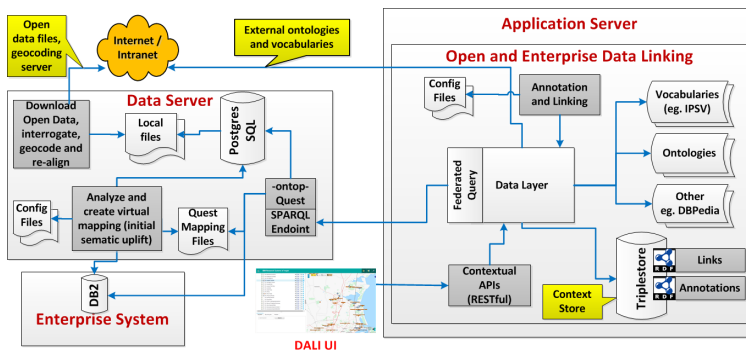


Fig. 2. Architecture and component diagram

For the prototypes and setup used in our **experiments**, for scenario 1 (public safety in Minneapolis), over a hundred datasets (from one customer’s DB2 enterprise database and open datasets) were automatically integrated, semantically annotated and linked in less than one hour, producing approximately 1 million virtualized data triples in two SPARQL end points (one for open data and one for customer data). For those datasets, almost 190.000 triples and annotations were extracted and stored in the centralized and indexed context store. For scenario 2 (NYC Safety Net) 34 datasets were ingested into one SPARQL end point, consisting on almost 114,440 data points (“rows” of data) and 3.5 million triples. However, any number of SPARQL end points can be configured, in order to meet potential scalability requirements.

3.1 Distributed Data Ingestion and Virtual RDF (Data Server)

The semantic layer enables de-coupling from the infrastructure of each source. While original enterprise data resides in the original relational systems and is accessed though virtual RDF, tabular files are automatically downloaded and linked to a relational database (currently PostgreSQL9.3). As stated in [26] having a semantic representation on top of a relational one improves data quality without adding much over-

head when converting CSV to a simple database schema. The datatypes are determined by examining the data: numbers, booleans, dates are converted into the correct format.

We use `-ontop-Quest`[24] as a virtualization technology, although, due to our flexible architecture, we can set as many different types of SPARQL endpoints as needed (e.g., interfacing directly to other triple stores, or other virtualized Enterprise DBs). Mapping files (i.e. files specifying one-to-one mappings between database property values and known RDF properties) for the virtualization servers are generated by our system using a rule and pattern-based entity extraction mechanism to detect: (1) **geographical entities** (using WGS84[29] to create properties for certain header labels with cell values corresponding to decimals number between -90 and 90); (2) the column with **names for instance labels** (*rdfs:label*); (3) columns (properties) with **contact** information: emails, addresses, phone numbers (named using VCARD[28]); (4) **temporal properties** (named using OWLTime[21]). Often, temporal properties (dates, month, year) are not part of the table itself but they need to be inferred from the table titles (e.g., *Crime Stats May 2013*); and (5) **object properties**, those columns for which values are mapped to instance URIs instead of literals, as for **datatype properties**, that's the case if the column is a foreign key, or for string (non-numeric) repeating values (below a threshold variance percentage) - e.g., city names. In addition to virtualization, this step includes **geocoding** of addresses.

3.2 Entity Uplift and Linking to the Web of Data (Application Server)

For each dataset in the virtualized RDF repositories, the schema information is extracted and stored it in the centralized Context Store: types, datatypes and object properties, and their set of possible instances, domains and ranges, together with entities' labels, if known, for indexing purposes. While some of the properties could be mapped to the W3C vocabularies in the previous step, to create a richer representation the entities in the Context Store are annotated and linked to an open set of both general and domain specific ontologies (that may vary according to the application domain). We use index searches and string similarity metrics [2] on the localname or label to annotate classes and properties with URIs found in the external sources used as annotators, as well as to find *owl:sameAs* links across instances. As such, we can detect synonyms and interpret acronyms (e.g., an instance named "PTSD" will be annotated with "Posttraumatic stress disorder", a DBpedia [1] redirect of the former).

Reusing well-known external sources to annotate the data adds significant value in terms of interoperability and discoverability, providing global meaning and common anchors across otherwise isolated data sources, without requiring the creation of a common model. In our scenarios, we use the Integrated Public Service Vocabulary (IPSV)[14], schema.org, WordNet[9] and DBpedia, which provides a wide domain coverage and geographical information. Specialist domain-knowledge models can also be used according to the use case. In particular for the Safety Net scenario we added the Social Care Taxonomy extracted from [27].

The annotations obtained for class labels that correspond to the table titles, often indicate the topic of the dataset. They are used to populate the Dublin Core [6] property *dcterms:subject*. Besides string similarity, the structure of the ontologies is used

to disambiguate and assign a confidence score to the candidate annotations. For example, for the dataset class labeled *Minneapolis Crime Data* various *dcterm:subject* properties are added to link this term to the DBpedia terms *Category:Crime_Data*, *dbp:Crime*, and *dbp:Minneapolis*, as well as the IPSV term *Crime*, among others¹. The annotations *IPSV:Theft_and_burglary* and *dbp:Robbery*, for a property in this dataset labeled *Robbery*, will have a high confidence score both as a good syntactic and semantic mappings, capturing how semantically close the URIs are in the original graph (the subject annotation *IPSV:Crime* is a broader term for the property annotation *IPSV:Theft_and_Burglary*, similarly *dbp:Crime* relates to the property annotation *dbp:Robbery* through a common broader term *dbp:Criminal_law*). These annotations, linking the source URI and the annotation URI, and their assigned confidence scores are stored in an annotation graph in the Context Store.

3.3 Contextual Access and Retrieval (RESTful APIs)

User needs are (a) *complex*; often, they cannot be expressed in a single query and exploration mechanisms are needed; (b) *not known in advance*; and (c) *comprising many factors* and requiring related information coming from different domains. Contextual retrieval requires understanding space, time, identity and links between entities. Annotations are used to capture the meaning of content in our RDF stores, by making explicit how entities are connected. The linkage is based on inference along linguistic relations in thesauri, taxonomies (e.g., *skos:broader/narrower* in IPSV, hyper(hypo)nyms in WordNet) and any kind of semantic relationships, such DBpedia *redirects*, *dcterm:subject*, *owl:sameAs*, etc. In this sense, two disparate datasets about diverse topics, like *Ambulance Call Outs* and the *Register of Fats and Oils Licenses* may both be relevant in the context of a user correlating the location of ambulances call outs and hospitals, because the latter contains the locations of establishments for which a license has been granted, where establishments can be filtered by type (hospitals, restaurants, etc.). User search and exploration needs while interacting with the data are captured and translated into structured queries. The retrieved information can be visualized on tables, maps, charts or as a ranked list of search results and saved into views, which can be exported in JSON or RDF. The following contextual APIs are exposed through REST services and integrated into a web based UI².

Catalogue-Based Dataset and Entity Explorations. Datasets can be explored according to the virtualized repositories where they belong (e.g., for Scenario 1 we have two repositories for customer data and open data) or by following any given reference taxonomical models. In both scenarios, the IPSV hierarchy is selected as the reference model for thematic catalogue exploration because of its wide coverage of city related topics, and a subset of DBpedia, namely all entities of type “PopulatedPlace” and their *PartOf* taxonomy, is selected as the geographical model. For the Safety Net scenario, the domain-specific Social Care Taxonomy is also used

¹ IPSV and dbp correspond respectively to <http://id.eds.org.uk/subject> and <http://dbpedia.org/resource/>

² Videos showcasing DALI:
<https://www.youtube.com/playlist?list=PL0VD16H1q5INAARBVy4GtSURLN4EWmcqF>

to catalogue the data. Datasets are organized into a hierarchical view of subcategories in the reference model(s), allowing an easy and thematic browsing of the data. The alignment is done automatically when the entity representing the dataset type is annotated with the model (*dcterms:subject*). Thus a dataset may sit in more than one subcategory, if appropriate. To avoid users having to navigate through empty categories, only the part of the catalogue tree for which there are datasets is shown.

In our Minneapolis scenario, the user can explore all datasets under the IPSV term “Safety” and subcategory “Emergencies” to find the *Fire and Ambulance call outs* dataset sit under the subcategory “Fire and rescue services”, the term its been annotated with. The user can also view at a glance all known datasets under a given location, e.g., all datasets for USA, state of Minnesota, Minneapolis city. The user can click on a dataset to display the tabular data (generated from their representation in RDF), explore the annotations, or plot spatial entities in a map. By clicking on any of these entities in the map the user can also explore its properties and attributes.

Semantic Keyword Search and Structured Filtering. Full-text search based on LARQ is used to discover entities matching the keyword search. The domain knowledge ontologies and models, used to annotate the data, are also used to expand the query with lexically and semantically related words. For example, the *Crime Data* dataset is returned as a result for the keyword search *Fire*. This is because *Fire* is semantically related to *Arson*, a datatype property in the dataset. *Fire* is lexically annotated with, among others, the IPSV term for *Fire IPSV:613* that is related to the term *IPSV:612*, also known as *Arson*, through the property *SKOS:related*. Datasets are ranked by number of matches (classes, properties and instances), weighted by the average syntactic score given to each match. If no matches are found for compound terms, they are recursively split into their constituents, e.g., *hospital health centers* would get datasets with results for both *hospital and health centers*. Each partial term is also semantically expanded and results are ranked considering also which part of the compound is matched (e.g., a match to *health center* is ranked higher than one to only *centers*). As per user request the matches can be plotted on the map and their provenance (semantic relatedness to the keyword) displayed.

While keyword search is a popular paradigm to retrieve data, structured queries provide the expressivity to specify complex information needs. Keyword search can be combined with faceted and spatial explorations in an iterative process, where the user can enter keywords and further refine the query by applying faceted filtering on the results, or any other dataset of interest. For example, the search for *Fire* gives back several matches in the *Police CAD* dataset, namely various instance values of the object property *Problem* – *Fire Assault*, *Assist Fire Personnel*, etc. – for each retrieved police call entity. The user can select to plot in the map only the entities related to *Fire Assault*, and overlay them with crime locations, from the *Crime Data* dataset, with more than a given number of arson crimes (specified by the user). In this case, *Arson* is a numerical datatype property and thus an equal/greater/lower than operator is suggested by the system to create the facet (different facets are suggested according to the datatypes –numerical/ boolean- and for object properties with a set of possible values). Users can also overlay spatial entities from any dataset in a map

using common constructs such as bounding boxes. This is implemented by executing a single SPARQL query to filter all entities URIs (and labels) with $Wgs84:lat$ and $Wgs84:lon$ values within the bounding box geo points.

Related Dataset and Entity Search. As datasets are aligned with ontologies through annotations, these annotations can be used to identify other datasets closely related to a given one, based on **topic** (datasets share the same or linked topics), **content** (datasets with related properties or content, even in different topics) or **entities in common** (same entities described in different datasets). In our scenario, the user can look for all datasets related to the *Crime Data* dataset with statistics on different kinds of crime. The *Police CAD* dataset is obtained as it is annotated with semantically related topics (*dcterms:subject* property). The user can explore the relatedness graph showing how the two datasets are linked, e.g., through the IPSV term *Crime and Law enforcement* as shown in Fig. 3.A for the topic-based criteria. Also, these two datasets are content-based related because they have properties or entities sharing the same annotations, as shown in Fig. 3.B, the instance value *Theft* for the property *Problem* in *Police CAD* has an *owl:sameAs* link to the DBpedia term *Category:Theft*, which is a broader term of *Category:Robbery*, an annotation property in *Crime Data*. The relatedness graphs for each annotator source are obtained on demand through a SPARQL federated query to find if there is a path (directly) linking annotations from a given dataset to annotations in other datasets from the same background source (e.g., DBpedia). Properties extracted based on rules, such as LAT/LONG and contact details (as defined in schema.org) are not considered relevant to identify related datasets.

Related datasets are ranked by summing the relevance weights for the relatedness graphs (pv) obtained for each criterion. The weight is calculated according to how significant the entity-level matches are – i.e., for the content-based relatedness graph: how many annotations are matched (num_anns_common) out of the total for the input dataset ($total_anns_input$), as well as, considering the average confidence score (WSc) of each matched annotations to assign weights to the different criteria when combined. The following formula, used to calculate the score (Sc) of a related dataset with respect to an input, responds to the intuition that datasets are more similar if they share more labels/annotations and share labels/annotations with large weights. The most relevant datasets have the highest score: $Sc_dataset = \sum_{pvi} Avg(WSc_anns_common)^{num_anns_common} \cdot \frac{num_anns_common}{total_anns_input}$

4 Experiments

In the first part, we perform a user study to evaluate the usability of the services exposed by means of the user interface, through a set of tasks that require retrieving complex information to create the relevant views. For the second part, we are demonstrating the effectiveness of the semantic search, whether adding related datasets has potential to improve search results, and the semantic cataloguing. We quantify the improvement on performance with respect to a non-semantic baseline. It is not our purpose to evaluate each step of the process or component independently, but to eva-

luate the relevancy of the results searched over distributed city data lifted into a knowledge graph, and in the context of a user-task.

4.1 Contextual Exploration: Usability

Evaluation Set-up. To evaluate performance in a more comprehensive manner, we have simulated a scenario, where evaluators are asked to use the system in order to answer the given complex information needs (simulating the role of a knowledge engineer). To test the ability of our system to retrieve this information, users are given a brief demo of the system and told they can use all the functionality available (Section 3.3). We have asked 5 users (all IBM employees and IT experts but not knowledgeable about semantic standards or the datasets) to retrieve the answers to the tasks in Table 3, which may span across more than one dataset. Queries 1-5 are part of Scenario 1, while Queries 6-10 are part of Scenario 2. The questions were given by experts of the respective commercial products with extensive hands-on experience in the domain. The order of the queries presented to each user was randomized. We evaluate on:

- Average number of tasks for which users found satisfactory answers vs. the ones for which they gave up or report a wrong result.
- Time to get the answer. We started a timer once the user was given the question and stopped the timer when the user would give up or report an answer.
- Which explorations and features were used to get the answers and the number of failed attempts.

Results. The results are shown in Table 2. We counted the features the users used to answer each query: semantic search (SEM), catalogue exploration (CAT), displaying tabular data, matches or entity information (DIS), plotting entities in a map (MAP), selection and faceted filtering (FAC), drawing a bounding box to visualize all entities within (BOU), and looking for related datasets and relatedness graphs (REL). The queries that were answered faster are those for which only one dataset is required to find the answer (Q3 and Q10). All users were successful in all tasks, except for Q2, Q3 and Q9 for which three different users (one for each query) gave up. For Q2 users would often attempt to find the answers in the *Police CAD* dataset (one of them gave up when she could not find it there), while the answer is found by applying faceted filtering on the property *robberies* in the *Crime* dataset (also returned as a result from the keyword search *robbery*). In Q3, one user gave up before realizing he could apply more than one facet filter in the same dataset. For Q9 a user failed to find the *readmission rates by heart attack* property in *Medicare Hospital Outcome of Care Measures*, picking instead the general *expected readmission rate* specified in the *Medicare Hospital Readmission Reduction* dataset. Facet filtering can then be applied to get the entities with the minimum readmission rates value.

Table 1. Test questions with the minimum n° of relevant sources and navigation links (that is the minimum number of steps as determined by the authors to obtain the answers).

| n° sources | Question (min. navigation links to answer them) |
|-------------------|---|
| 2 | Q1: The sport stadium in Minneapolis near one of the most dangerous pedestrian areas in the USA (4) |
| 2 | Q2: Which cafes are near robbery crimes areas in Minneapolis (5) |
| 1 | Q3: Locations with more than 10 car thefts and 10 arson crimes (3) |
| 2 | Q4: All places holding both a liquor license and sidewalk permits (2) |
| 2 | Q5: All police disturbances near the Creekview center (5) |
| 3 | Q6: Community programs for the elderly population in Queens (5) |
| 2 | Q7: Readmission rates for hospitals with emergency services (4) |
| 1 | Q8: Home delivery meal services in the Bronx (2) |
| 2 | Q9: Non profit hospitals with the minimum mortality rates from heart attack (3) |
| 1 | Q10: After school programs for middle school kids (2) |

For most tasks, users started by using keyword search or catalogue exploration. When catalogue exploration fails, such as when looking for *Home Delivered Meals* in Q8, users will use keyword search to find information hidden in the datasets (in this case the entities in the *DFTA_Contracts* dataset, which value for the object property *Contract_Type* is *Home_Delivered_Meals*). Besides searching and catalogue exploration, plotting a dataset or search results on the map and displaying entity and tabular data were features used in all queries (the latest, often used just to figure out if the provenance of the given answer, or search result, is sufficient). Facets were used in almost all queries. The query with the second largest number of attempts, Q7, is because first the *boolean* property specifying if a hospital has emergency services is found in a different dataset to the readmission rates one; and second, it took a while for a few users to understand that they can plot in the map the entities in common for both datasets, by combining the faceted and co-reference filters. In general, all queries related to hospitals took longer in average because of the large amount of clinical data, both in terms of number of datasets about hospitals, and the number of properties within each dataset (more than 50 in some of the Medicare datasets). For these cases, search is more efficient than catalogue browsing.

The bounding box feature was rarely used, even if it is the faster way to answer queries such as Q1 (e.g., by drawing a bounding box near the most dangerous pedestrian area in Minneapolis), or even when the users knew where to look in the map. For example for Q5, once the Creekview center was found (in the dataset *Minneapolis parks and recreations*) only one user attempted to find reported disturbances by searching in the datasets with entities near by. The reason behind Q5 largest number of failed attempts, is because users searched for the answers in both the *Police CAD* and *Crime* datasets, while only the former has the answer for disturbances. Nonetheless, most users would prefer to use faceted search to filter by area (if the property exists, such as Queens for Q6), rather than a bounding box. Relatedness was also hardly used, even if it is a useful feature for queries such as Q9 (the query that took the longest in average) to find all other hospital datasets with related properties to the *Medicare Hospital Readmission Reduction* dataset, e.g., those describing different measurements on readmission rates. Users preferred to do a semantic search and explore all matches till they find the one that is most appropriate. Finally, all queries were answered in average in less than 4 minutes, although the deviation between us-

ers varied greatly (with a max. time of 6 minutes), and required less than 7 steps in average (links) to retrieve the answers to a query. In total, the average of the sum of all failed attempts per query for all 5 participants is 7.5.

Table 2. Results from left to right for each query: the average of the sum of all features used by the 5 participants per query; the average num. of links used by user to get an answer; the average of the sum of all failed attempts for all 5 participants per query; the num. of users which succeeded; the average time to answer the queries and the deviation.

| Features used (for all 5 users per query) | | | | | | | Avg. links (For user) | Failed attemps (for all 5 userse per query) | Success | Avg. time | Deviation | |
|---|-----|-----|-----|-----|-----|-----|--------------------------|--|---------|-----------|-----------|---------|
| SEM | CAT | DIS | MAP | FAC | BOU | REL | | | | | | |
| Avg | 8.6 | 4.1 | 9.4 | 7.6 | 4.9 | 0.5 | 0.8 | 6.7 | 7.5 | 4.7 | 0:02:37 | 0:03:04 |

4.2 Semantic Search: Performance

Evaluation Metrics. We compare the precision and improvement on recall of our semantic approach with respect to a syntactic baseline based on *Lucene* full-text searches, without semantic expansion. We measure **precision** (P), defined as the number of relevant datasets with respect to the number of datasets found, and **recall** (R), defined as the number of relevant datasets found with respect to the total of relevant datasets. Total recall cannot be measured (no gold standard to evaluate against), so we consider as total all the unique relevant datasets found using both approaches, and measure the improvement in recall of the semantic approach (R_s) with respect to

the baseline (R_B) as: $R_improvement = 1 - \frac{R_B}{R_s}$ $P = \frac{Total_relevant}{Total}$

Increasing recall often comes with a decrease in precision, which is affected by the quality of the annotations, noisy mappings and ambiguous lexically related words. Therefore, precision is also measured for the top N of results: **TOP-1**, **TOP-3** and **TOP-5**, in order to evaluate the efficiency of the ranking.

Evaluation Set-up 1: Semantic Search. We evaluate the impact of semantic query expansion on 20 keywords, obtained randomly from the logs generated after the evaluation in 4.1. We distinguish between *coverage*, *correctness* and *relevance*. Coverage is measured by counting the number of results. However, this does not indicate whether the results are relevant. To evaluate correctness and relevance we have engaged three of the previous five evaluators. Each of them has assigned a score with a discrete value in $\{0,1,2\}$ for each datasets retrieved as a result, where: 0 implies the proposed dataset is based on semantically incorrect assumptions, i.e., due to an ambiguous annotation; 1 implies the proposed dataset is based on semantically correct justifications, but it is not relevant; and 2 implies the proposed dataset is correct and relevant to complement the information of the original dataset. For users to judge relevancy they can relate to the tasks presented in Section 4.1. Given the three user evaluations, a result was considered correct if at least two evaluators were rating it with values higher than 0, and it was considered relevant if at least two evaluators were rating it with 2 and the remaining evaluation was not 0.

Set-up1 Results: as shown in Table 3. The semantic approach improves the average recall of the system by 33%, as it is able to find all the relevant datasets found using the baseline approach plus some additional ones, without much loss in precision, from 70% in the baseline to 69% using the semantic approach. Furthermore, as shown in Table 4 the semantic ranking increases the precision from the 0.7 average for the baseline to 0.9, 0.78 and 0.75 for the top-1, top-3 and top-5 results respectively. In sum, a semantic approach helps increase recall, while also increasing the precision for the top ranked results, i.e., those users are likely to check. The syntactic errors are mostly due to ambiguous terms, e.g., the keyword *Fire* was mapped to the *Fire Station* dataset, but also to the instance *Fire restaurant* in the *Fats and Oils Licenses* dataset. The users rated the first match as relevant (2) and the second as incorrect (0). While these syntactic mappings were also captured by the semantic component, the inaccurate mappings were ranked lower than the more accurate semantic matches. In addition, other relevant datasets were only found using the semantic approach, like the *Crime Stats* with the property *Arson*. In two cases (*pedestrian* and *liquor licenses*) the syntactic approach performed slightly better than the semantic. This is because all relevant datasets were syntactically matched, and the semantic extension retrieved inaccurate lexically related results – that although they scored lower they were part of the top-5 (as less the 5 datasets were found in total). The semantic approach improves over the baseline in particular when asking for schema elements (types, properties) or term combinations (e.g., *sport stadiums*) rather than instance labels. In the latter case, both approaches perform the same (returning the instances with matching labels).

Table 3. Comparison between baseline and semantic approaches

| Query | Baseline | | | Semantic Approach | | |
|----------|----------|--------------|----------|-------------------|--------------------|--|
| | Coverage | Prec./Recall | Coverage | Prec./Recall | Recall Improvement | |
| 20 total | 3.15 | 0.70 / 0.66 | 5.65 | 0.69 / 1 | 0.33 | |
| TOTAL | | | | | | |

Table 4. Comparison for the top ranked results

| Qi | Baseline | | | Semantic Approach | | | Relatedness-TOP3 | | |
|---------|----------|-------|-------|-------------------|-------|-------|------------------|------|------|
| | TOP-1 | TOP-3 | TOP-5 | TOP-1 | TOP-3 | TOP-5 | #Total | #New | #Rel |
| Average | 0.7 | 0.69 | 0.70 | 0.9 | 0.78 | 0.75 | 7.15 | 5.45 | 3.25 |

Evaluation Set-up 2: Semantic Relatedness. We evaluate whether our algorithms that retrieve highly related tables can also improve the result of searches (by pulling up datasets based on their relatedness to top ranked datasets). To measure improvement on recall: (1) we find all related datasets for the TOP-3 ranked results in each of the previous queries (#Total); (2) from all related datasets, we select only the ones that did not appear in the top-5 search results (#New); and (3) from all the #New datasets, we select the ones rated as relevant (#Rel);. As before, we randomly ordered the searches and asked the 3 users to rate the results between 0-2. A dataset selected as related to a given one based on semantically correct assumptions (e.g., common annotations) may not be relevant, because the content of the dataset is not specialized enough to give any extra information, or the commonalities are not significant. In the same way, a disparate dataset from a different topic may be relevant because it describes similar entities from different points of view relevant as part of an exploration task.

Set-up 2 Results: in Table 4, on the right, we show the number of unique related datasets added by looking at related datasets from the TOP-3 (#Total), that do not appear as part of the TOP-5 semantic search results (#New) and that the users rated as relevant (#Rel). In average 5.45 datasets out of 7.15 found were not in the original TOP-5 results. A significant portion of the new related datasets are considered relevant (3.25 obtained an average user rating of 2). Thus, relatedness can be used to suggest further relevant explorations from user searches by making “hidden semantic links” across entities explicit, based on different *relatedness* criteria. In this experiment only the related datasets with an average score over a 0.5 threshold were selected. As an example, *Crime Data* is one of the TOP-3 results for *Fire*, because of the annotation property *Arson*. A related dataset, not part of the top ranked search results, is *Law Enforcement*. This is because, both datasets share the same IPSV subject *Crime*. Ranked before *Law Enforcement* is *Police CAD*, as their respective annotations are linked through IPSV and DBpedia, as shown in the relatedness graphs in Fig. 3.A&B. The total score for a related dataset is calculated by considering the combined weight of all relatedness graphs across the two datasets. As such, like in semantic search, datasets with inaccurate matches from ambiguous annotations are often ranked in lower positions. Users judged the relevance of the results, assessing if the relation between two datasets is meaningful considering the tasks in Section 4.1. The Fleiss’ kappa agreement between users was moderate/substantial ($k=0.6$)[8].

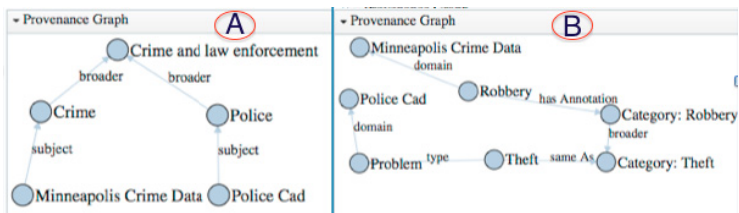


Fig. 3. Relatedness graphs for related datasets

Evaluation Set-up 3: Semantic Catalogue Search. IPSV is selected as the ontology of reference for browsing and searching datasets by category. For a total of 30 randomly chosen datasets, we ask the same 3 users to rate how well the dataset has been classified for each category (note a dataset may belong to more than one) with a value between [0-2], where 0 means the category is inaccurate, 1 that the dataset is correctly classified but it is not too helpful (e.g., not at the right level in the taxonomy), and 2 the category is relevant to find the dataset by browsing through the hierarchical tree. We measure precision as the number of datasets under a relevant category from the total.

Set-up3 Results: all selected 30 datasets are categorized (100% coverage), each dataset is categorized into an average of 4.33 categories (the maximum being 11 and the minimum 1), and 29 datasets out of 30 have at least one relevant category. If a minimum score is applied as threshold (experimentally selected to favor precision, i.e. less datasets placed under an incorrect category, with a little negative effect on recall), to ensure that the categories matched to a given term have certain quality, then we get

2.16 categories in average (with a maximum of 6 and a minimum of 1), and 28 datasets out of 30 have at least one relevant category. The precision for the relevant results in the non-filtering approach is 0.54; while in the score-based one is 0.78. The precision for the correct results is 0.84 and 0.93 respectively. The loss in recall is 0.76 on the score-based approach if we consider the non-filtering one as perfect recall (1). As an example, to specific categories may not be relevant if they are too specific, such as *Swimming pools* as the category for the dataset *Leisure Facilities*, or too general, like *Leisure and culture*. For this dataset, *Leisure center* was rated as the relevant category.

5 Related Work, Discussion on Usefulness and Future Lines

We address the timely issue of data consumption, exploration, search and linking in the context of cities, through a practical approach based on Linked Data. There are a number of unique challenges and opportunities for the IR and semantic communities in order to make heterogeneous city data searchable, and to address complex information needs that require analyzing the relationships between entities in context. In this light, we propose a lightweight and incremental information sharing approach directed towards leveraging the information spaces defined by the LOD datasets and city data of diverse ownership, to give meaning to the latter.

Various publishing platforms exist for automating the lifting of tabular data into semantic data, and interlinking datasets with existent LOD datasets [25][18]. In [18] Google Refine is used to allow expert users to clean and export tabular data into RDF through a reconciliation service extended with Linked Data sources. Following the tools and recommendations by the W3C used for automatically converting tabular data (mostly CSV) and relational tables into RDF[12][10][7], in [26] a set of tabular data from the Norwegian directorates FactPages is transformed into a LOD dataset. The DataLift project [25] goes a step forward by transforming the raw RDF extracted from the source format to well-formed RDF by mapping to selected ontologies. These approaches are based on the assumption that each row is an entity and columns are RDF properties for the first RDF conversion; In [25] the user is asked to input a set of vocabularies to describe the lifted data and the target RDF is then generated through a set of SPARQL construct queries. In QuerioCity[17] we proposed a platform to provide semantic context for city data and metadata by following a centralized and incremental graph-based approach. The focus is on data-view manipulation by different publishers while tracking provenance. However, the drawback is the significant added cost on indexing this data. Unlike previous approaches, DALI presents a light-weight approach that considers the distributed nature of RDF and it is able to ingest any customer or open data available, as long as it follows a tabular representation. The data is (1) virtualized into RDF, extracting spatial/temporal entities, datatypes and object properties for each entity (row), even if often entities are not linked (no foreign keys); (2) exposed and contextualized with any reference ontologies and models of choice in the Web of data; and (3) combined in arbitrary ways across data sources, through semantic services that support users, without knowledge of SPARQL, to refine explorations and federated queries.

Extracting structured data from tables on the Web and semantic search has also attracted interest from search engines [26][3]. In [26] columns in web tables are associated with types (automatically extracted from web pages), if the values in that column can be matched as values of the type. In [22] table rows containing entities of specific types derived from an ontology are automatically annotated. In [3] an approach is proposed for finding related tables on the Web based on: (1) Entity complement: union of entities with similar schemas; (2) Schema complement: joins of columns about the same entities. The BBC has annotated its world service radio archive with DBpedia topics. These associations, stored in a shared RDF store, are used to improve search and navigation within the archive [23]. Different from these works we propose an application that covers not only the annotating and semantic querying across a diverse set of heterogeneous, distributed enterprise and well-formed open tabular data for cities, but also the lifting of these data silos to Linked Data.

There are no tools that we can use for a meaningful comparison and we still are a long way from defining standard evaluation benchmarks to evaluate search methods for urban data platforms and comprising relevance judgment of similar datasets. From information platforms, such as DubLinked.ie, one can obtain the most common keyword searches and most downloaded datasets. More effort is needed to better capture the users' intention and experience while using the tool, in the context of complex real tasks that involve more complicated manipulation and combinations of datasets. Illustrated in the context of two real commercial use cases, our user-based evaluations are a first step to probe the feasibility and effectiveness of a knowledge-mining prototype to query entities distributed across datasets, without the need to ETL data. We also conducted a small-scale study with experts from the urban planning and the health domains to further validate the system. The experts were three city planners from Dublin City Council and three staff members from the Department of Nursing and Midwifery from Trinity College Dublin, evaluating scenario 1 and 2 respectively. We have used the widely cited methodology from Davis[4] on predicting how much people will use a new product. Given our focus on functional components, rather than user interfaces, and the higher importance of usefulness compared to ease of use (also reported in [4]), we report numbers on the former. In Table 5 we report combined numbers since there were no significant differences between the scenarios. Overall, users gave positive scores (avg. =5.85, on a scale of 1-7). Critically for the goals of our system, users gave the system a high score for allowing them to work more quickly.

Table 5. Questions and usefulness scores [1-7] for both scenarios.

| | | | | | |
|-----------------------------|-----|------------------------------|-----|--------------------------|-----|
| <i>Quality of work</i> | 6 | <i>Control over work</i> | 5.5 | <i>Work more quickly</i> | 6.5 |
| <i>Critical to my job</i> | 5.5 | <i>Increase productivity</i> | 6 | <i>Job performance</i> | 5.5 |
| <i>Accomplish more work</i> | 6 | <i>Effectiveness</i> | 5 | <i>Makes job easier</i> | 6 |
| <i>Useful</i> | 6.5 | | | | |

There is an abundance of research to be pursued: off-the-shelf co-reference tools can be used to make links more dense, as long as they do not depend on the availability of training data; incorporating social media data; exploring interactive ways to build NLP queries, exploiting user feedback to improve the machine-generated ranking, and supporting dataset discovery in the Web of Data using data-hubs such as CKAN.net.

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. *Web Semantics* 7(3), 154–165 (2009)
2. Cohen, W., Ravikummar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: *IJCAI Workshop on Information Integration* (2003)
3. Das Sarma, A., Fang, L., Gupta, N., et al: Finding related tables. In: *SIGMOD 2012*
4. Davis, F.D.: Perceived usefulness, ease of use, and user acceptance of information technology. *MIS quarterly* 13, 319–340 (1989)
5. Ding, L., Lebo, T., Erickson, J.S., DiFranzon, D., et al.: A portal for linked open government data ecosystems. *Web Semantics* 9(3) (2011)
6. Dublin Core: <http://dublincore.org/documents/dcmi-terms/>
7. Ermilov, I. Auer, S., Stadler, C. Csv2rdf: user-driven csv to rdf mass conversion framework. In: *ISEM 2013*
8. Fleiss, J.L., Cohen, J.: The equivalence of weighted kappa and the intraclass correlation coefficient as measure of reliability. *Educational and Psychological Measurement* 33, 613–619 (1973)
9. Miller, G.A.: WordNet: A Lexical Database for English. *Communications of the ACM* 38(11), 39–41 (1995)
10. Han, L., Finin, T.W., Parr, C.S., Sachs, J., Joshi, A.: RDF123: from spreadsheets to RDF. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 451–466. Springer, Heidelberg (2008)
11. <http://www.ibm.com/developerworks/industry/library/ind-intelligent-operations-center/>
12. <http://www.w3.org/2001/sw/rdb2rdf/> and <http://www.w3.org/TR/r2rml/>
13. IBM Curam: <http://www-03.ibm.com/software/products/en/social-programs>
14. IPSV: <http://doc.esd.org.uk/IPSV/2.00.html>
15. Kotoulas, S., Sedlazeck, W., Lopez, V., et al.: Linked data for citizen-centric care. In: *MIE 2014*
16. LARQ: <https://jena.apache.org/documentation/larq/>
17. Lopez, V., Kotoulas, S., Sbodio, M.L., Lloyd, R.: Guided exploration and integration of urban data. In: *Hypertext 2013*
18. Maali, F., Cyganiak, R., Peristeras, V.: A publishing pipeline for linked government data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012*. LNCS, vol. 7295, pp. 778–792. Springer, Heidelberg (2012)
19. NYC Open Data portal: <https://data.cityofnewyork.us/data>
20. Official datasets provided by Medicare and Medicaid services: <https://data.medicare.gov/>
21. OWL Time: <http://www.w3.org/TR/owl-time/>
22. Quercini, G., Reynaud, C.: Entity discovery and annotation in tables. In: *EDBT (2013)*
23. Raimond, Y., Ferne, T.: The BBC world service archive prototype. In: *ISWC Semantic Web Challenge (2013)*
24. Rodriguez-Muro, M., Rezk, M., Hardi, J., Slusnys, M., Bagosi, T., Calvanese, D.: Evaluating SPARQL-to-SQL translation in ontop. In: *ORE 2013*
25. Scharffe, F., Atemezing, G., Troncy, R., Gandon, F., et al.: Enabling linked-data publication with the datalift platform. In: *Workshop on Semantic Cities, AAAI 2012*
26. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the norwegian petroleum directorate’s factpages as semantic web data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) *ISWC 2013, Part II*. LNCS, vol. 8219, pp. 162–177. Springer, Heidelberg (2013)
27. Social Care taxonomy from the UK Social Care Institute for Excellence: <http://www.scie.org.uk>
28. VCARD: <http://www.w3.org/TR/vcard-rdf/>
29. WGS84 geo-coordinates: <http://www.w3.org/2003/01/geo/>

Knowledge Graphs

Building and Using a Knowledge Graph to Combat Human Trafficking

Pedro Szekely¹, Craig A. Knoblock¹(✉), Jason Slepicka¹, Andrew Philpot¹,
Amandeep Singh¹, Chengye Yin¹, Dipsy Kapoor¹, Prem Natarajan¹,
Daniel Marcu¹, Kevin Knight¹, David Stallard¹,
Subessware S. Karunamoorthy¹, Rajagopal Bojanapalli¹, Steven Minton²,
Brian Amanatullah², Todd Hughes³, Mike Tamayo³, David Flynt³,
Rachel Artiss³, Shih-Fu Chang⁴, Tao Chen⁴,
Gerald Hiebel⁵, and Lidia Ferreira⁶

¹ Information Sciences Institute, University of Southern California,
Marina Del Rey, CA, USA

{pszekely,knoblock}@isi.edu

² InferLink Corporation, El Segundo, CA, USA

³ Next Century Corporation, Columbia, MD, USA

⁴ Columbia University, New York, USA

⁵ Universitt Innsbruck, Innsbruck, Austria

⁶ Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Abstract. There is a huge amount of data spread across the web and stored in databases that we can use to build knowledge graphs. However, exploiting this data to build knowledge graphs is difficult due to the heterogeneity of the sources, scale of the amount of data, and noise in the data. In this paper we present an approach to building knowledge graphs by exploiting semantic technologies to reconcile the data continuously crawled from diverse sources, to scale to billions of triples extracted from the crawled content, and to support interactive queries on the data. We applied our approach, implemented in the DIG system, to the problem of combating human trafficking and deployed it to six law enforcement agencies and several non-governmental organizations to assist them with finding traffickers and helping victims.

Keywords: Linked data · Knowledge graphs · Entity linkage · Data integration · Information extraction

1 Introduction

Human trafficking is a form of modern slavery where people profit from the control and exploitation of others, forcing them to engage in commercial sex or to provide services against their will. The statistics of the problem are shocking. In 2014 the International Labor Organization on The Economics of Forced Labour¹

¹ <http://bit.ly/1oa2cR3>

reported that \$99 billion came from commercial sexual exploitation. Polaris² reports that in the United States 100,000 children are estimated to be involved in the sex trade each year, and that the total number of victims is likely much larger when estimates of both adults and minors as well as sex trafficking and labor trafficking are aggregated. Estimates indicate that traffickers control an average of six victims and derive \$150,000 from each victim per year. The sex trafficking industry is estimated to spend about \$30 million in online advertising each year. These advertisements appear in hundreds of web sites that advertise escort services, massage parlors, etc. The total number of such advertisements is unknown, but our database of escort ads crawled from the most popular sites contains over 50 million ads.

The objective of our work is to create generic technology to enable rapid construction of knowledge graphs for specific domains together with query, visualization and analysis capabilities that enable end-users to solve complex problems. The challenge is to exploit all available sources, including web pages, document collections, databases, delimited text files, structured data such as XML or JSON, images, and videos. This paper describes the technologies and their application to build a large knowledge graph for the human trafficking domain. We focus on the role of Semantic Web techniques to address the technical challenges, and we describe the challenges in using Semantic Web techniques given the scale of the data, the performance requirements of the application, and the social challenges of working within a large consortium of developers unfamiliar with Semantic Web technologies.

In the following sections we describe the challenges that we faced, present our approach to building a knowledge graph applied to human trafficking and how we addressed these challenges, describe how the system is being used in practice, and then present the related work, discussion, and future directions.

2 Challenges

The main source of data for the human trafficking domain is the web. There are hundreds of web sites containing millions of escort ads that sex providers use to attract clients. Figure 1 shows a screenshot of one web site containing the titles of escort ads. The actual ads contain photos of the provider and text that describe the services provided, prices and contact information. The ads use coded language, and often purposefully obfuscate the information to make it difficult for law enforcement to use search engines such as Google to look for information, as shown here:

```
YOU don't wanna miss out on ME :) Perfect lil booty Green eyes Long
curly black hair Im a Irish, Armenian and Filipino mixed princess :)
♥ Kim ♥ 7o7~7two7~7four77 ♥ HH 80 roses ♥ Hour 120 roses ♥ 15 mins
60 roses
```

The phone number is obfuscated, using unicode characters and letters to code the phone number. The rates are listed as roses instead of dollars, and abbreviations

² <http://www.polarisproject.org/index.php>

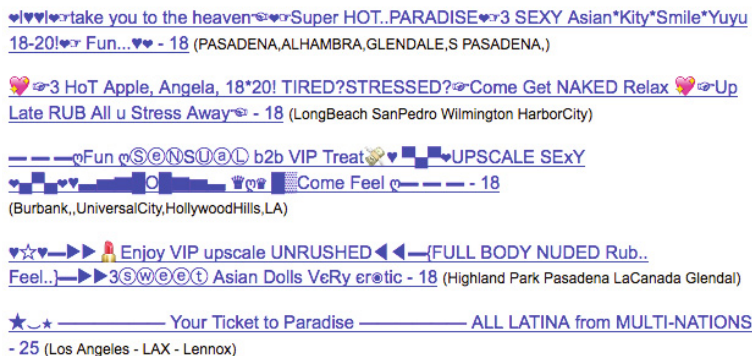


Fig. 1. Example page with an index of escort ads

such as HH stand for time periods (half hour). Capitalization is often arbitrary, which lowers the effectiveness of standard entity extractors.

In addition to escort ads, we use reference datasets such as Geonames, which provides geographic location information, and phone exchange databases, which provide information about the locations where phone numbers are registered.

Figure 2 shows a small example of the knowledge graph that we want to construct. The graph has nodes representing ads and data extracted from the ads, including images, phone numbers, locations and working names. The figure does not show a wide array of other attributes extracted from ads, including the title and text of the ad, physical attributes such as ethnicity, eye color, and hair type, and attributes such as rates. Note that the graph includes edges that represent the output of analytic processes such as Jaccard similarity among the text of ads and similarity of the images.

The main objective of our work is to construct a high quality knowledge graph and to provide a flexible and easy-to-use query interface to enable law enforcement agencies and non-governmental organizations (NGOs) to investigate leads and to assemble comprehensive evidence of trafficking for legal cases. A typical law enforcement scenario is to search the graph using the phone number of a suspected trafficker, retrieve all ads that mention the phone number, display the ads on a map or a timeline, find other ads that contain similar images or text, assemble a list of other phone numbers mentioned in the resulting set of ads, and then cross-reference the discovered numbers in law enforcement databases. A typical NGO scenario is to assist families in identifying lost children. In this case, strong attributes such as phone numbers are usually not available, so the search uses soft attributes such as physical characteristics, photos, likely locations, etc.

Building a knowledge graph to effectively support these types of scenarios requires addressing a number of challenges:

No Agreement on APIs or Schemas: Our team is part of a large consortium of over 15 organizations funded to develop “domain-specific search and indexing” technology and applying it to address the human trafficking challenge. Different

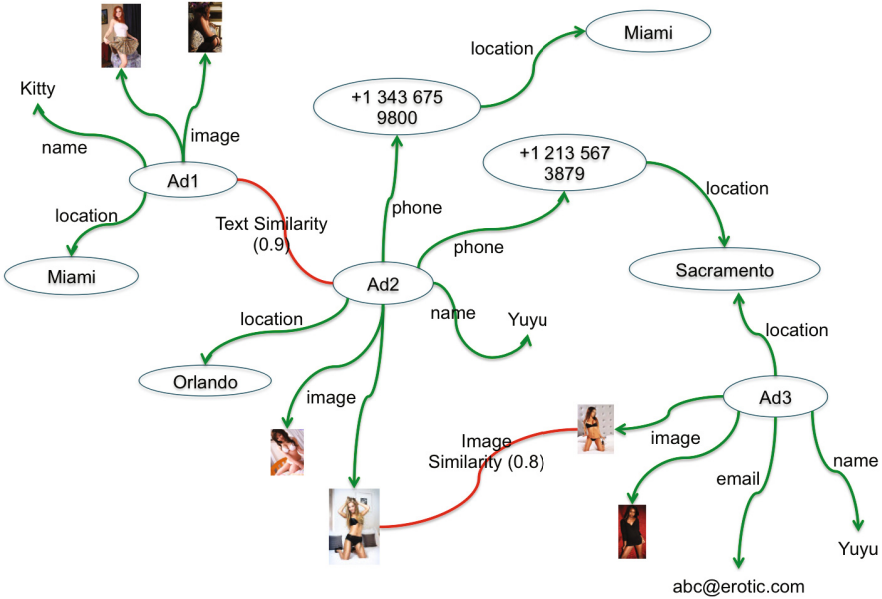


Fig. 2. Example knowledge graph

organizations focus on different aspects of the problem, crawling, extraction, knowledge-graph creation, query, analytics and visualization. Needless to say, it is not possible for a large group of organizations to agree on APIs or schemas for the inputs and outputs of the various components of a larger system. For example, several organizations are developing extraction technologies focusing on different aspects of the problem. The result is a collection of extraction tools that produce output in completely different formats and using completely different schemas. Some encode their extractions in relational databases, some produce text delimited files, and some produce JSON objects. All use different attribute names and structures to encode the information, and they produce literals in different formats (different formats for dates, phone numbers, etc., different units for physical characteristics, time periods, etc.) Our challenge as builders of the knowledge graph is to consume the output of the various extractor and analytic components produced by other organizations.

Provenance: A key requirement for law enforcement is to trace back from the knowledge graph to the original documents as they need to subpoena the raw documents from the web site providers. Furthermore, different extractors often extract the same attributes from pages (e.g., multiple phone number extractors) and sometimes they produce conflicting extractions. To produce a high quality knowledge graph, it is necessary to reason about the origin of the different extractors to determine which extractions should be added to the knowledge graph and which ones should be discarded. To address this concern, it is necessary to record provenance for every node and edge in the knowledge graph.

Scale: The consortium has already crawled 50 million web pages, and is continuously crawling the sources, producing on average 160,000 additional pages every day resulting in a knowledge graph with over 1.4 billion nodes. One challenge is to rebuild the complete knowledge graph from scratch in less than one day to incorporate improvements resulting from new versions of extractors and other software components. A second challenge is to incrementally extend the knowledge graph to incorporate the data from newly crawled pages.

Query Flexibility and Performance: Today’s users expect the ease of use and performance of search engines such as Google and web portals such as Amazon. A key challenge is to support efficient keyword search to produce a ranked list of matching nodes and to support efficient faceted browsing to enable users to quickly and easily filter the results.

Opposition to Semantic Web Technology: Most other organizations in the consortium are unfamiliar with Semantic Web technologies, and after initial discussions, clearly unwilling to learn or use these technologies. Our challenge is to seamlessly integrate the Semantic Web technologies we advocate with the “mainstream” technologies that other organizations are comfortable with (e.g., relational databases, NoSQL, Hadoop, JSON).

3 Building Knowledge Graphs

In this section we describe our overall approach to building knowledge graphs. We present the techniques using the human trafficking domain as an example, although the general approach can be applied to many other domains. Figure 3 shows the architecture of the overall system, called DIG (Domain-Insight Graphs). Each of the following subsections present the components of the DIG architecture in detail and describes how they were applied on the human trafficking data to build a knowledge graph for that domain.

3.1 Data Acquisition

Data acquisition requires finding relevant pages and extracting the required information from those pages. DIG uses Apache Nutch (nutch.apache.org) to

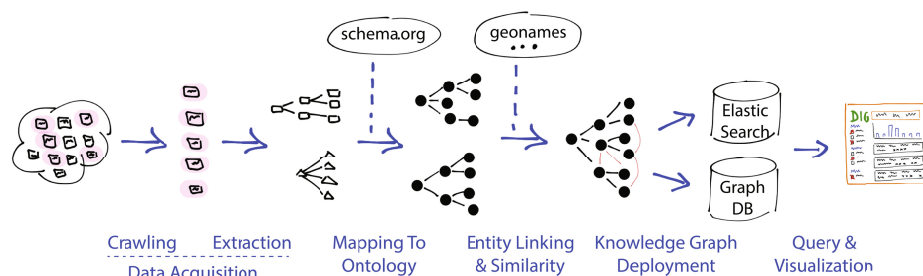


Fig. 3. Architecture for building a knowledge graph

support crawling at scale. Nutch offers a RESTful configuration interface that makes it easy to specify the URL patterns to be crawled, to monitor crawling progress, and to define revisit cycles to re-crawl periodically, downloading revisions to already crawled pages. To support focused crawling, we integrated a semi-structured content extractor into Nutch. The extractor can identify specific elements within a page, such as the list page with the ads shown in Figure 1, and direct Nutch to follow only those links.

After crawling, the next step is to extract features from the harvested data to produce a structured representation that can be used for indexing and linking in the next step of the DIG construction pipeline. Given the wide variety of pages and data on the web, it is infeasible to develop data extraction technology that works for any page and any type of data. DIG provides an open architecture that makes it easy to integrate a wide range of extraction technologies, so that data scientists can select the extraction technology most appropriate for the pages in their application domain. In addition to providing an open data extraction architecture, DIG also provides components for extracting data from both semi-structured pages and plain text.

The DIG semi-structured page extractor, called the *landmark extractor*, identifies elements in a page using landmarks defined with regular expressions. DIG provides a learning component that automatically infers rules for the landmark extractor from examples. To train an extractor, the data scientist provides a collection of pages and corresponding extractions (e.g., name, phone number and location from a set of pages that all come from the same site). Using a handful of examples, the learning component automatically creates a landmark extractor that extracts data from similar pages.

To support extraction from text, DIG offers a capability to enable data scientists to easily train extractors specialized to an application domain. In the human trafficking domain we need to extract data elements such as eye-color, hair type and color, and ethnicity from the escort advertisements. To train a new extractor for a text corpus, a data scientist highlights the desired data elements in a small number of sample sentences or short paragraphs selected from the corpus. For example, in the sentence “Perfect Green eyes Long curly black hair Im a Irish, Armenian and Filipino”, the data scientists highlights “Green eyes” and “Long curly black hair”. After the data scientist designates a text corpus and defines the examples, DIG automatically constructs thousands of tasks to acquire additional annotations using the Amazon Mechanical Turk crowd sourcing platform (www.mturk.com). The output of the Mechanical Turk tasks feeds a Conditional Random Field [6] that learns an extractor from the annotations provided.

3.2 DIG Ontology

The DIG ontology defines the terminology for representing the nodes and edges of the knowledge graph. A key decision in the design of our knowledge graph is the desire to record provenance data for each node and edge of the graph. In addition, we want to store the values of nodes in different representations,

including a string representation used for keyword search and display, and a structured representation used for structured query and analysis.

Consider, for example, the representation of a phone number (e.g., +1 343 675 9800) extracted from an ad (e.g., Ad2). Figure 2 depicts the ad and the phone number as nodes and uses a labeled edge to depict the relationship. The figure does not show the provenance and other metadata associated with each node and edge in the graph.

Unfortunately, RDF does not provide a convenient way to represent provenance and other metadata for each edge in the graph. The recommended way is to represent this information using reification. Reification is inconvenient as the metadata points to the edge it annotates, and retrieving it requires an additional query that requires a join.

In DIG we represent the graph edges as first class objects called *Features*. Each node in the graph can contain a collection of Feature objects, where each Feature represents an outgoing edge from the node. To represent the knowledge graph in RDF we introduce the following classes and properties:

```
FeatureCollection a owl:Class .
hasFeatureCollection a owl:ObjectProperty ;
  rdfs:range FeatureCollection .
Feature a owl:Class .
featureValue a owl:DatatypeProperty ;
  rdfs:domain Feature .
featureObject a owl:ObjectProperty ;
  rdfs:domain Feature .
```

An instance of `FeatureCollection` represents the collection of edges associated with a node in the graph; `hasFeatureCollection` associates such an instance with a node in the graph. An instance of `Feature` represents an edge in the graph. The `featureValue` represents the value of the Feature as a literal and corresponds to what normally would be a data property in an ontology. When the value of a `Feature` can also be represented as a structured object, `featureObject` represents the value of the feature as an RDF object. For each type of edge in the graph, the ontology also includes a property as illustrated in the following example for phone numbers:

```
phonenummer_feature a owl:ObjectProperty ;
  rdfs:domain FeatureCollection ;
  rdfs:range Feature .
```

The main benefit of our ontology is that all the information about a node or an edge in the graph can be conveniently accessed using property paths without the need to do separate queries to retrieve metadata or provenance. Consider the following examples:

1. `hasFeatureCollection / phonenummer_feature`
2. `hasFeatureCollection / phonenummer_feature / featureValue`
3. `hasFeatureCollection / phonenummer_feature / featureObject`
4. `hasFeatureCollection / phonenummer_feature / featureObject / countryCode`
5. `hasFeatureCollection / phonenummer_feature / prov:wasDerivedFrom`
6. `hasFeatureCollection / phonenummer_feature / prov:wasAttributedTo`

The first property path returns the **Feature** objects that hold the phone values as well as the provenance and other metadata associated with each phone number edge; the second one returns the phone numbers as literals, and the third returns the phone numbers as structured objects; the fourth returns the country codes of the phone numbers; the fifth returns the URIs of the original documents from which the phone numbers were extracted; and the sixth returns the URIs that identify the extraction software that produced the value. DIG uses the PROV ontology to record provenance.

3.3 Mapping Data to the DIG Ontology

The data extraction processes produce a variety of data in different formats. The next step towards construction of a knowledge graph is to convert all the extracted data as well as auxiliary structured sources to the DIG ontology. The data conversion process consists of two parts. We first define a mapping from the source schema to the ontology, and then we execute the mapping to convert the data into JSON-LD, a Linked Data representation in JSON.

To convert the data, we build on our previous work on Karma [5,10,11], which provides a semi-automatic approach to defining mappings from a data source to an ontology. Figure 4 is a screenshot of Karma showing the mapping of phone numbers extracted from escort ads to the DIG ontology. The challenge in using Karma for this task was creating a representation of the data that could be efficiently queried and converting the data into this representation at a massive scale. In the remainder of this section, we first describe how Karma is used to clean and model the data and then we describe the new capabilities to support creating large-scale knowledge graphs.

Karma provides an integrated environment to clean data while mapping it to an ontology. The user interface for cleaning the data is similar to a spreadsheet in

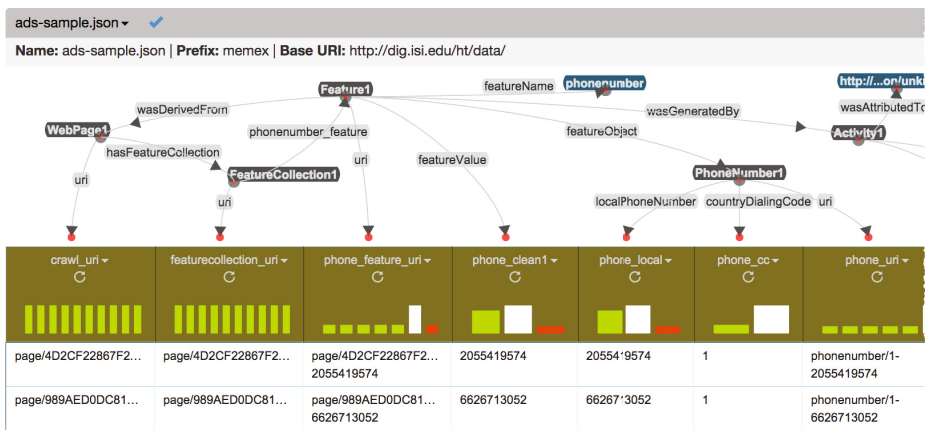


Fig. 4. Screenshot of Karma showing data extracted from escort ads and the associated model of this source for phone numbers

that users can define new attributes as functions of existing attributes. Karma uses Python as its formula language enabling users to define arbitrary data transformations succinctly. For example, the first column in Figure 4 (`crawl_uri`) defines the URI for a page as the SHA1 hash³ of the concatenation of the `url` and the `timestamp` attributes of the source. All other columns in the figure are also defined using Python scripts. Note that the `phone_clean1` column contains a normalized representation of the phone numbers.

The graph in the figure defines the mapping of the source to the DIG ontology as a model of the source in terms of the classes and properties defined in the ontology. In the model shown in the figure, the dark ovals represent classes and the links labeled with gray rectangles represent properties. The links between classes are object properties and the links between a class and an attribute of the source denote either data properties or specify that an attribute contains the URI for an instance of a class. The model shows that a `WebPage` has a `FeatureCollection`, which has a `phonenumbers_feature`. The phone Feature has a `featureValue` that points to the `phone_clean1` attribute and a `featureObject` property that points to a structured representation of the phone. Properties such as `wasDerivedFrom` and `wasGeneratedBy` record provenance. The URIs are important because several sources contain extractions from the same web pages, and the URIs are used to merge the converted data. For example, the `FeatureCollection` of each web page has a unique URI so that when the URI is reused in models the corresponding features are added to the same `FeatureCollection`.

One of Karma's unique capabilities is that it learns to define the mappings from sources to an ontology. Each time a user maps an attribute of a source to a class in the ontology, Karma uses the values of the attribute to learn the mapping [8]. When users define relationships between classes, Karma also learns so that later, when a user models a new source, Karma can automatically suggest the properties and classes to model the new source. For the human trafficking application we constructed 21 models with very similar structures. The learning component coupled with the easy-to-use graphical interface makes it possible to define a model of the complexity shown in Figure 4 in about 30 minutes. Karma proved to be effective to address the data heterogeneity challenge.

By default, Karma generates n-triples for every record and every node and link in a model. This serialization of RDF can be loaded in any RDF triple store. However, many of the big data tools, such as ElasticSearch (www.elastic.co), cannot read triples and require data about objects to be represented in JSON documents. We extended Karma to also generate JSON-LD and to allow developers to customize the organization of the JSON-LD documents. For example, if a developer specifies a root such as `WebPage1` in Figure 4, Karma will generate the JSON-LD shown in Table 1 for the first row of the source (URIs omitted for brevity).

The JSON-LD document is organized according to the structure of the model. Developers can customize the generation of the JSON-LD documents by specifying constraints to stop expansion at specified nodes (e.g., do not expand the

³ <https://en.wikipedia.org/wiki/SHA-1>

Table 1. Example JSON-LD generated by Karma

```

{
  "@context": "http://...",
  "a": "WebPage",
  "hasFeatureCollection": {
    "a": "FeatureCollection",
    "phoneNumber_feature": {
      "a": "Feature",
      "featureObject": {
        "a": "PhoneNumber",
        "localPhoneNumber": "6626713052" },
      "featureName": "phoneNumber",
      "wasGeneratedBy": {
        "wasAttributedTo": "http://dig.isi.edu/ht/data/soft...",
        "a": "Activity",
        "endedAtTime": "2014-04-02T17:55:23" },
      "wasDerivedFrom": "http://dig.isi.edu/ht/data/page/5C27...",
      "featureValue": "6626713052" }}}

```

Activity1 object) and constraints to include or exclude specific properties. If the developer does not specify any constraints, Karma uses all nodes connected to the root, breaking cycles arbitrarily. The approach is flexible, allowing developers to specify how much information around the root should be included in the JSON-LD documents. Furthermore, developers can generate JSON-LD documents organized around different roots. For example, developers can produce JSON-LD documents organized around phone numbers, and these would contain all web pages referring to a given phone number.

In addition to extending Karma to support JSON-LD, we also modified Karma so that it would run under Hadoop. After we have built a model for a given source, Karma can then apply this model to convert each source, with potentially millions of records, into JSON-LD, running the process on a cluster such as Amazon Web Services (AWS).

3.4 Computing Similarity

The next step in the processing is to identify potential links between similar data items. Due to the size of the datasets, this is a challenging problem. DIG provides capabilities to compute similarity for images and for text data. DIG's image similarity capability uses DeepSentiBank, a deep convolutional neural networks approach [3]. The approach extracts over 2,000 features from each image and computes compact hash codes (only 256 bits per image) that can be used to retrieve similar images. An important benefit of such similarity sensitive hash codes is that there is no need to train the similarity algorithms with images in the domain of interest. In our human trafficking application we used this approach with a database of 20 million images. The system precomputes the compact hash codes, which requires about 40 hours on a single machine, and is then able to find identical and near duplicate images for a new image over the entire 20 million images in

less than 2 seconds. For example, given a photo of a person, the system can find other photos of that person taken in similar settings (e.g., in the same room) or with similar clothing, even if the person is in a different pose.

DIG uses Minhash/LSH algorithms [7] to compute similarity on text data, as these algorithms can scale to large datasets containing hundreds of millions of documents. These algorithms work by computing random hashing functions on the tokens of a document, and can find pairs of similar items in a large dataset in $O(n * \log(n))$ time. Minhash/LSH computes an approximation of Jaccard similarity, defined as the ratio of tokens two documents have in common over the combined number of tokens in the two documents. The text similarity can be precomputed offline, which requires 12 hours on a single machine, and as new documents are added the similarity is incrementally evaluated.

To use these algorithms, DIG constructs a document for each data record and then runs the Minhash/LSH algorithms over the associated documents. DIG provides a library of tokenization methods to compute the tokens that form the document associated with a data record. If a data record contains sentences or larger texts, then the document can be formed using the words in the document, or word n -grams (sequences of several words). If the data records contain small values such as names of people or geographic locations, then the document can be formed using character n -grams (sequences of several characters). These n -grams are useful because they allow the algorithm to find similar items when they use slightly different spellings for words.

In the human trafficking application, we currently compute the similarity of ads based on the text of the ads. This type of similarity helps investigators find ads that are likely authored by the same person or organization. We are currently working to compute similarity on locations, phone numbers, names, etc. and then use the various similarity scores for performing entity resolution.

3.5 Resolving Entities

The next step in the DIG pipeline is to find the matching entities (often called entity resolution). Consider the entities shown in the knowledge graph in Figure 2. The entities are people, locations, phone numbers, etc. and each of these entities has one or more properties associated with them. The task in this step is to determine which data corresponds to the same entities. For example, we want to know which web ads refer to the same person or which geographic references actually refer to the same location. The output of this step is a set of explicit links between entities extracted from different sources.

DIG addresses two variations of the problem. The easier case is when there is an appropriate reference dataset that contains all the relevant entities. For example, GeoNames (geonames.org) is a comprehensive geographical database containing over 2.8 million populated places, so it can be used as a reference set for cities. In DIG, we use GeoNames as a reference set for populated places, so entity resolution for cities becomes the problem of mapping mentions of cities to the appropriate entity in GeoNames (e.g., mapping the string “Los Angeles, CA” to the record identifier for the city of Los Angeles in California).

To solve this variant of the entity resolution problem, a data scientist first uses Karma to map the reference dataset to the ontology being used. Then, the data scientist uses the similarity analysis discussed in the previous section to compute similarities between records in the reference dataset and other records in the knowledge base. The output of the similarity matching step is a small number of candidate entities for each entity mention in the knowledge base, typically less than 100. Next we define matching algorithms tuned to the entity type. Data scientists can define custom matching algorithms or use classifiers such as support vector machines (SVM) to define custom matching components to determine whether a mention should be matched with an entity. This matching step only needs to operate on the small number of similar candidates generated in the similarity matching, so this matching step can evaluate all candidates without affecting the overall scalability.

The second variant of the entity resolution problem addresses the case when there is no reference set for the entities of interest. For example, there is no reference set for the individuals described in the ads. For these cases it is necessary to infer the set of entities from the ads. DIG represents each entity as a set of features (e.g., a person can be represented by the phone number, the photos, the locations mentioned, and so on). This first step creates entities for each of the individuals in the ads. The second step eliminates the redundant entities using a clustering approach similar to Swoosh [1].

3.6 Generating the Graph

At this point, all datasets have been converted into JSON-LD using the domain ontology, and the links between similar records have been identified and evaluated. Next, records containing unique identifiers are merged. A key feature of the approach, is that Karma can use the URIs in the documents to merge JSON-LD documents generated using different models, denormalizing them and thereby precomputing joins. For example, Karma can merge the JSON-LD document shown in Figure 1 with similar documents generated for other features (e.g., location, email, etc.). The resulting merged document will have a single `FeatureCollection` object containing all the features generated from the various models. The merging occurs at all levels of the JSON documents.

In the human trafficking domain, the result is a connected knowledge graph where ads are connected via entities such as phone numbers and email addresses, as well as through text and image similarity. Consider the case of several phone extractors. When extractors agree, there will be a single `phonenumber_feature` with multiple `wasGeneratedBy` provenance objects, one for each extractor. When the extractors disagree, the `phonenumber_feature` will contain an array of two `Feature` objects, each with a single `wasGeneratedBy` provenance object.

The beauty of our overall approach is that we can use Karma models to generate alternative representations of the knowledge graph, each tuned to different uses of the data. This addresses one of the challenges discussed earlier, which is the opposition to Semantic Web technologies. By enabling developers to customize the root and contents of JSON-LD documents, and by customizing

the JSON-LD context file, Karma can generate JSON documents that mainstream developers desire. This approach simultaneously illustrates the benefit and addresses their reluctance to use Semantic Web technologies.

We take advantage of this flexibility, and we index the JSON-LD documents in Elasticsearch, a distributed search and analytics engine supporting full text search and structured search over extremely large collections of JSON documents. Elasticsearch, like Apache Solr (lucene.apache.org/solr), is built on Lucene, a widely used technology familiar to many developers. Elasticsearch offers great scalability and performance and is being used by very large, high-traffic web sites such as GitHub and LinkedIn. Unlike an RDF triple store, Elasticsearch has no support for joins, but this is not a problem because Karma denormalizes the data producing self-contained JSON-LD documents that contain all the information that the application wants to show users as a result of a query.

Karma can also store the data in AVRO format (avro.apache.org), a format compatible with the Hadoop processing stack. Storing the knowledge graph in AVRO format makes it possible to process the graph data using map/reduce on Hadoop, enabling scalable processing of all the data (e.g., to compute node similarity.)

3.7 Query and Visualization

Figure 5 shows a screenshot of the DIG query interface. The interface paradigm is similar to that of popular web sites such as Amazon (amazon.com). Users can search using keywords, and can filter results by clicking on check-boxes to constrain the results. For example, the figure shows that the user clicked on “latina”, to filter results to ads with the selected ethnicity. The user interface issues queries to the Elasticsearch index, responding to queries and updating all facets over the 1.4 billion node graph in under 2 seconds.

4 In Use

The application of DIG to combat human trafficking is in use today. The system has currently been deployed to six law enforcement agencies and several NGOs that are all using the system in various ways to fight human trafficking, such as by locating victims or researching organizations that are engaging in human trafficking. After evaluation of the current prototype is completed, a updated version of this application will be deployed to more than 200 government agencies that are interested in using DIG. Reports to date indicate that DIG tool has already been successfully used to identify several victims of human trafficking, but due to privacy concerns we cannot provide additional details.

All of the data used in the deployed application comes from publicly available web sites that contain advertisements for services. In the currently deployed application as of July 2015, there are 60 million ads with roughly 162,000 new ads per day (with new ads updated every hour). The number of objects (RDF

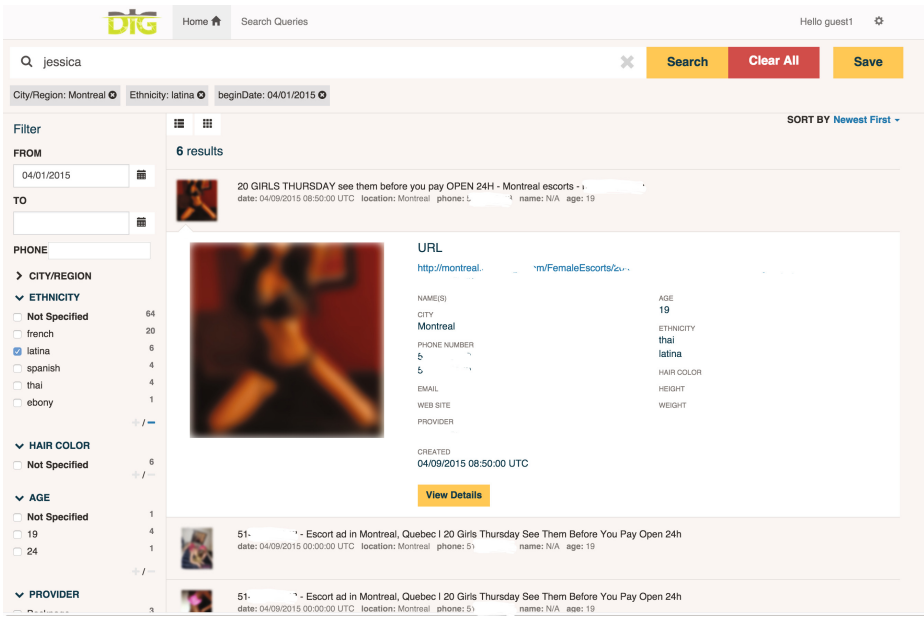


Fig. 5. Screenshot of DIG query interface showing results a query on the keyword “jessica”, filtered by city/region, ethnicity and date to focus on a small number of ads

subjects) is 1.4 billion and the number of feature objects is 222 million. These features are broken down in Table 2.

The DIG data processing pipeline and the Elasticsearch index that supports the query interface runs on a cluster with 23 nodes (384GB of RAM, 16 cores). The processing pipeline is implemented using Oozie workflows (oozie.apache.org). The times to rebuild the knowledge graph from scratch on a 23 node cluster is 27 hours and 15 minutes. Data files do not need to be redeployed to HDFS for re-processing, so the time to rebuild the graph is under 19

Table 2. Breakdown with number of each feature object in the datastore

| Feature | Count | Feature | Count | Feature | Count |
|--------------|------------|-------------|-----------|----------------|------------|
| Payment | 533,506 | Gender | 823,577 | # Tattoos | 277,597 |
| Email | 1,212,299 | Grooming | 181,544 | Username | 297,287 |
| In/Out Call | 92,564 | Hair Color | 760,312 | Phone Number | 46,410,902 |
| Age | 33,668,863 | Hair Length | 626,333 | Postal Address | 49,446,945 |
| Person Build | 1,071,288 | Hair Type | 614,043 | Provider Name | 52,925,078 |
| Bust | 602,235 | Height | 6,631,311 | Rate | 7,208,428 |
| Cup Size | 434,762 | Hips Type | 39,375 | Website | 811,218 |
| Ethnicity | 12,790,179 | Zip Code | 101,749 | Eye Color | 581,263 |

hours, satisfying our requirement to be able to rebuild the knowledge graph in under 24 hours.

5 Related Work

There is a variety of related work on building knowledge graphs. The Linked Open Data can be viewed as a large, heterogeneous knowledge graph. However, the data it contains has not been mapped to a single domain ontology, there is only limited linking, and the quality of the data is highly variable. Nevertheless, there are many useful and high quality sources that do form the basis of a knowledge graph, including DBpedia, Geonames, and the New York Times. These are heavily curated and carefully linked and provide coverage for very specific types of data. Each source in the Linked Open Data is created using different methods and tools and it results in a highly variable knowledge graph that requires considerable additional effort to use to build new applications.

Several commercial efforts that are building knowledge graphs, including the Google Knowledge Graph⁴ and the Microsoft Satori Knowledge Repository⁵. These graphs provide general knowledge about people places, organizations, and things with the purpose of improving search results. Since these systems are used to improve the search query results, the knowledge contained in these systems is general and spans many different domains. In contrast to the general Google and Microsoft knowledge graphs, our goal is to build comprehensive domain-specific graphs that can then be used for analysis in a specific domain.

The Linked Data Integration Framework (LDIF) [9] also focuses on building domain-specific knowledge graphs. Like DIG, it provides a data access module, a data translation module [2], and an identity resolution module [4]. LDIF also addresses scalability, processing data in a cluster using Hadoop. The most significant difference is that LDIF focuses on existing RDF and structured sources while DIG aggregates data from both structured and unstructured sources, including text documents, web pages, databases, and photographs. DIG also provides a highly extensible architecture for integrating new capabilities.

6 Conclusion

In this paper we described the DIG system⁶ and discussed how it can be used to build a knowledge graph for combating human trafficking. The role of Semantic Web technologies is central to the success of the system. We represent all the data in a common ontology, define URIs for all entities, link to external Semantic Web resources (e.g. Geonames), and publish data in RDF using multiple serializations for different back-end databases. DIG is not limited to human trafficking and has

⁴ https://en.wikipedia.org/wiki/Knowledge_Graph

⁵ <https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/>

⁶ Available under an Apache Version 2 License (dig.isi.edu).

already been applied in other problems domains including illicit weapons, counterfeit electronics, identifying patent trolls, and understanding research trends in both material science and autonomous systems.

In future work, we plan to refine the tools and technology to make it easier and faster to build new applications. We will also investigate techniques to leverage ontological axioms to enable richer queries and facets in the user interface. For example, subclass relationships (e.g., Escort sub-class-of Person) could be used to produce facets that enable users to narrow results to specific subclasses of objects. This will require techniques to efficiently forward-chain the inferences and explicitly represent them in the knowledge graph.

Acknowledgments. This research is supported in part by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract number FA8750-14-C-0240, and in part by the National Science Foundation under Grant No. 1117913. Cloud computing resources were provided in part by Microsoft under a Microsoft Azure for Research Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, NSF, or the U.S. Government.

References

1. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widom, J.: Swoosh: A generic approach to entity resolution. *The VLDB Journal* **18**(1), 255–276 (2009). <http://dx.doi.org/10.1007/s00778-008-0098-x>
2. Bizer, C., Schultz, A.: The r2r framework: publishing and discovering mappings on the web. In: *Workshop on Consuming Open Linked Data (COLD)* (2010)
3. Chen, T., Borth, D., Darrell, T., Chang, S.: DeepSentibank: visual sentiment concept classification with deep convolutional neural networks. *CoRR abs/1410.8586* (2014). <http://arxiv.org/abs/1410.8586>
4. Jentzsch, A., Isele, R., Bizer, C.: Silk: generating RDF links while publishing or consuming linked data. In: *9th International Semantic Web Conference* (2010)
5. Knoblock, C.A., Szekely, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyani, M., Mallick, P.: Semi-automatically mapping structured sources into the semantic web. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) *ESWC 2012. LNCS*, vol. 7295, pp. 375–390. Springer, Heidelberg (2012)
6. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the 18th International Conference on Machine Learning (ICML)* (2001)
7. Leskovec, J., Rajaraman, A., Ullman, J.D.: *Mining of massive datasets*. Cambridge University Press (2014)
8. Ramnandan, S.K., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning semantic labels to data sources. In: Gandon, F., Sabou, M., Sack, H., d’Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) *ESWC 2015. LNCS*, vol. 9088, pp. 403–417. Springer, Heidelberg (2015)
9. Schultz, A., Matteini, A., Isele, R., Mendes, P.N., Bizer, C., Becker, C.: LDIF: A framework for large-scale linked data integration. In: *21st International World Wide Web Conference (WWW 2012). Developers Track* (2012)

10. Taheriyani, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: A graph-based approach to learn semantic descriptions of data sources. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 607–623. Springer, Heidelberg (2013)
11. Taheriyani, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: A scalable approach to learn semantic models of structured sources. In: Proceedings of the 8th IEEE International Conference on Semantic Computing (ICSC 2014) (2014)

Data Processing, IoT, Sensors

Semantic-Guided Feature Selection for Industrial Automation Systems

Martin Ringsquandl¹(✉), Steffen Lamparter², Sebastian Brandt²,
Thomas Hubauer², and Raffaello Lepratti³

¹ Ludwig-Maximilians University, Munich, Germany
martin.ringsquandl.ext@siemens.com

² Siemens AG Corporate Technology, Munich, Germany
{steffen.lamparter,sebastian-philipp.brandt,thomas.hubauer}@siemens.com

³ Siemens AG Digital Factory, Nuremberg, Germany
raffaello.lepratti@siemens.com

Abstract. Modern industrial automation systems incorporate a variety of interconnected sensors and actuators that contribute to the generation of vast amounts of data. Although valuable insights for plant operators and engineers can be gained from such data sets, they often remain undiscovered due to the problem of applying machine learning algorithms in high-dimensional feature spaces. Feature selection is concerned with obtaining subsets of the original data, e.g. by eliminating highly correlated features, in order to speed up processing time and increase model performance with less inclination to overfitting. In terms of high-dimensional data produced by automation systems, lots of dependencies between sensor measurements are already known to domain experts. By providing access to semantic data models for industrial data acquisition systems, we enable the explicit incorporation of such domain knowledge. In contrast to conventional techniques, this semantic feature selection approach can be carried out without looking at the actual data and facilitates an intuitive understanding of the learned models. In this paper we introduce two semantic-guided feature selection approaches for different data scenarios in industrial automation systems. We evaluate both approaches in a manufacturing use case and show competitive or even superior performance compared to conventional techniques.

Keywords: Semantic data models · Feature selection · Automation systems · Machine learning

1 Introduction

Processing and mining of large data sets in modern industrial automation systems is a major challenge due to the vast amount of measurements generated by several types of field devices (e.g. sensors, controllers, actuators). Deployment of machine learning models requires upfront feature selection in order to obtain a reduced feature set, thereby speeding up processing time and preventing overfitting, while still preserving inherent characteristics of the original data. Even

in the age of massively distributed data processing, feature selection remains one of the main problems in automation, since it is a highly domain expertise-intensive task [7]. On the other hand, data generated by engineered systems exhibits many structural dependencies that domain experts are well aware of. This holds especially for industrial automation systems that are systematically planned and simulated before going into production. For example, for a given electric motor, it is documented how torque, speed and power measurements relate to each other. Thus, there is no need to compute correlations between them for asserting statistical dependence.

These computations are common in most of today’s feature selection techniques, therefore they exhibit some major disadvantages when applied in high-dimensional data as observed in today’s automation systems [13]. By accessing huge proportions of the original data, they quickly become computationally expensive, plus they are prone to losing valuable information, especially when transforming the feature space to lower dimensions so that the remaining variables can no longer be intuitively interpreted. Motivated by the commonly faced difficulties of *a*) processing vast amounts of data and *b*) integrating domain knowledge into learning models, the semantic guidance approaches of this paper were developed in order to facilitate what remains the most expertise intensive task – feature selection.

In general, there are two different types of high-dimensional data that need to be considered in separation. The first case is present if we are given a huge number of both features and instances. In this scenario, we argue for an approach, in analogy to the notion of the usage of *OWL 2 QL* for ontology-based data access (OBDA), that makes it possible to perform feature selection on *TBox* level rather than instance (data) level [10]. The other case is given when there are fewer instances than features, also called *sparse* data (e.g. rare events such as device failures). For this type of data, embedded feature selection techniques like Lasso have shown to be very effective [5]. Therefore, we also introduce an embedded feature selection approach that leverages from engineering background knowledge in semantic data models. The subsequent sections will describe both approaches and their application in more detail.

2 Application: Manufacturing Use Case

As an application scenario, consider multiple assembly lines that are part of a car door production facility. The core of each assembly line is responsible for welding the window frame and inner door panel, which happens in a semi-automated fashion, as the actual welding is done by a human worker. The overall system consists of an automated frame loading station that is responsible for putting window frames on a conveyor kit. These kits are then routed through the core assembly process by an electrically-operated conveyor. After the products have been assembled, they must go through a final quality control that verifies integrity of certain product characteristics. If quality conforms to specification, the product is sent to an outgoing packaging station.

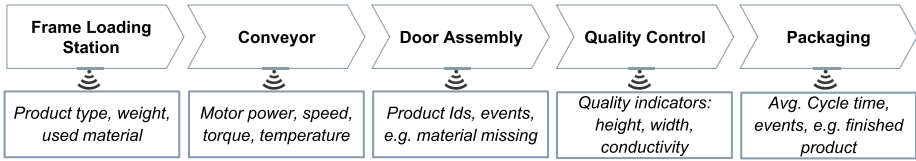


Fig. 1. Door assembly process and associated measurements

Plant operators are responsible for planning and scheduling of operations based on incoming orders. For this task, the operators have to assess uncertainties such as varying cycle times of each produced piece. Since production processes are of stochastic nature, it is non-trivial to get a solid estimate of the time when a certain product will be finished. In this case, decision support can be given by training advanced regression models that help to provide more robust and up-to-date time estimates. During production, all of the devices (e.g. light-barrier sensors, power meters, etc.) generate task-specific measurement data as shown in Figure 1. Today, data collection is agnostic of any machine learning tasks, as it is merely concerned with high-throughput historic data storage. As far as data analytics is concerned, any (sub-)set of the measurements taken could possibly be relevant for the time estimate regression task. A kind of brute-force approach would be to use all present data and try to train, for example, an ordinary least squares regression (OLS) model. However, this approach has some major shortcomings, since the OLS will include irrelevant and redundant information for fitting its coefficients and is very likely to overfit to particular patterns in the training data, therefore it is not going to generalize well in a live production scenario.

Consider the two regression models on the left-hand side of Figure 2 that try to predict cycle times. In this small example, employing five different predictor variables causes the model to overfit, while after p-value-based feature selection we obtain a more smoothed fit using only `Conveyor1Time`. On the right-hand side it can be seen how constraining the five predictors by a regularization penalty reduces their coefficients until they are effectively set to zero. For example, `LoadingProductWeight` quickly gets eliminated due to its small effect on the regression task. Since it is known that product weight is part of the overall weight feature, it could have been removed beforehand without any computation. Throughout this paper, we will relate to this learning problem of forecasting product-specific cycle times in an automated manufacturing system given a huge number of different sensor measurements as a running example.

3 Preliminaries

In this section, we first introduce the notion of semantic data models in the manufacturing domain in 3.1 and how their graph representation is used to measure structural similarity of feature entities. This is followed by a description of the general problem of embedded feature selection in linear models in 3.2.

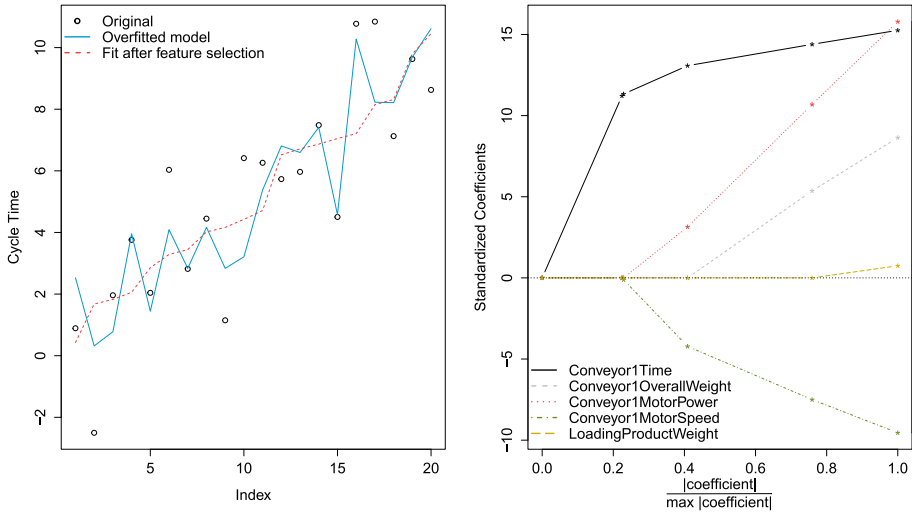


Fig. 2. Visualization of multiple linear regression model. Left: Overfitted model using all predictor variables vs. model after feature selection. Right: Coefficient values under decreasing regularization penalty

3.1 Semantic Representation of Manufacturing Data

Instead of having yet another information model language, we argue for the usage of the well-established Semantic Web standards to create, link, and share information models of automation systems in the manufacturing domain. For the purposes of feature selection, we augmented and tailored the Semantic Sensor Network (SSN) ontology¹ to meet the requirements of describing features in automation data, as can be seen in Figure 3. Here, the **Feature** concept is modeled as subclass of `ssn : InformationEntity`. Resorting to SSN is also beneficial for manufacturing systems, since devices and processes can naturally be integrated into its schema. Further details of this *feature ontology* are given in section 4.1. The graph representation of RDF-based² ontologies is a suitable property that we want to exploit for the description of dependencies between machine learning features. In formal terms, an RDF graph can be defined as a multi-graph.

Definition 1 (RDF Graph). *An RDF graph is a multi-graph $G = \langle V, E \rangle$ where each edge $e_i \in E$ is defined as triple (s, p, o) : $s, o \in V$ and p is the edge’s label.*

This formal definition allows us to specify the degree of similarity between feature entities in the graph by means of common structural patterns. Graph kernel

¹ <http://www.w3.org/2005/Incubator/ssn/ssnx/ssn>

² <http://www.w3.org/RDF/>

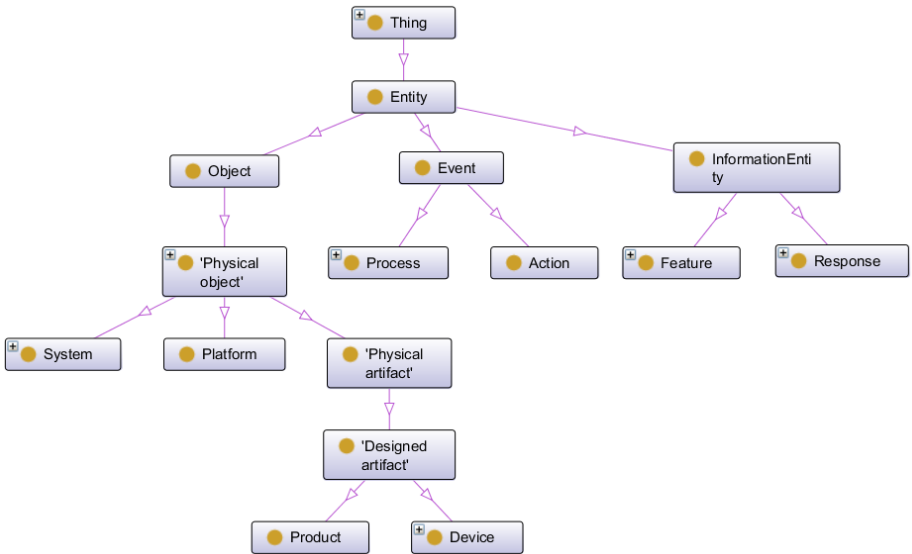


Fig. 3. Taxonomy of manufacturing feature ontology

functions have been shown to work well for capturing such patterns. The emerging field of machine learning in Linked Data has brought up a number of graph kernel functions particularly designed for RDF graph data.

Definition 2 (RDF Graph Kernel). *A graph kernel function is any function $\kappa : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$ s.t. for all $G_i, G_j \in \mathcal{G}$ satisfies $\kappa(G_i, G_j) = \langle \phi(G_i), \phi(G_j) \rangle$ is a valid kernel, where \mathcal{G} is the space of RDF graphs and ϕ is a mapping to some inner product space.*

Given n entities, the graph kernel can be denoted as kernel matrix:

$$\kappa = \begin{bmatrix} \kappa(G_1, G_1) & \kappa(G_1, G_2) & \dots & \kappa(G_1, G_n) \\ \kappa(G_2, G_1) & \kappa(G_2, G_2) & \dots & \kappa(G_2, G_n) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(G_n, G_1) & \kappa(G_n, G_2) & \dots & \kappa(G_n, G_n) \end{bmatrix}$$

In order to get pairwise similarities between all feature entities in our feature ontology, we can resort to one of the state-of-the-art graph kernels [8]. The idea of graph kernels is that every entity can be represented as the graph that is spanned by its adjacent entities up to a certain depth d . Then, similarity between two graphs is given by some metric, e.g. size of the intersection graph or pairwise isomorphisms like in the popular family of Weisfeiler-Lehman graph kernels [3]. In Figure 4 a simplified example of the graph spanned by feature `Conveyor1Speed` is shown at different levels of depth d . Data properties of entities like RDF literals (formatted in *italic style*) are usually considered to belong

to their respective entity and therefore they do not span a new depth level. Clearly, quality of similarity calculations depends on the amount of knowledge put into the ontology creation process. Nevertheless, we expect that already a small number of annotations can support feature selection.

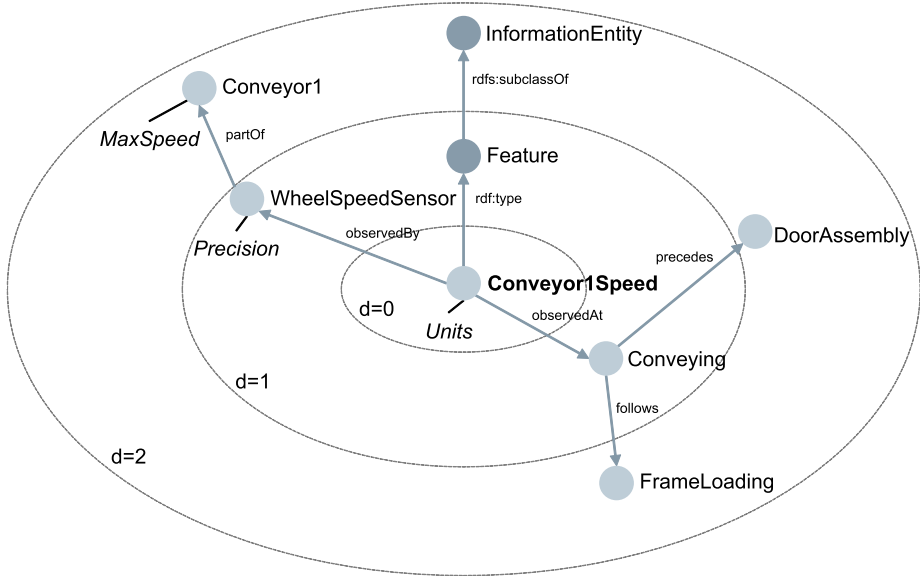


Fig. 4. Neighborhood graph of feature *Conveyor1Speed* at different depth values

Before describing how to exploit this notion of similarity between features in the training procedure of the manufacturing machine learning models, a general introduction to learning linear models is given.

3.2 Linear Model Embedded Feature Selection

Linear models are still one of the most popular machine learning models, especially in domains, where the emphasis lies on insights gained from looking at the model's coefficients. For example, the coefficients of the cycle time estimator regression can be interpreted for decision support in order to take action and reduce their influence on the overall cycle time. In case of sparse data, embedded feature selection techniques have shown to be very effective compared to other conventional feature selection. In the subsequent, we will introduce some standard formal notation of generalized linear models and their sparsity-inducing feature selection ability.

Given a training set $\{x_i, y_i\}_{i=1}^n$ where $x_i \in \mathcal{R}^p$ is a p -dimensional feature vector and $y_i \in \mathcal{R}$ is the response, i.e. for regression or classification. We consider

learning a linear model $h : R^p \rightarrow R$ with $h(\mathbf{w}) = \mathbf{w}^T \mathbf{x}$, where \mathbf{w} is a parameter vector. The general form of the regularized optimization problem is:

$$\operatorname{argmin}_{\mathbf{w}} l(y, h(\mathbf{w})) + \lambda \Omega(\mathbf{w}) \quad (1)$$

Here, $l(\cdot)$ denotes the loss function and $\Omega(\cdot)$ is the regularization term, also called penalty. The value of λ controls how much weight is given to the penalty, which is used to prevent overfitting of large parameter values. Setting $l(\cdot)$ to the square loss and $\Omega(\cdot)$ to the ℓ_1 -regularization results in the standard Lasso model:

$$\hat{\mathbf{w}}_{Lasso} = \operatorname{argmin}_{\mathbf{w}} (y - h(\mathbf{w}))^2 + \lambda \|\mathbf{w}\|_1 \quad (2)$$

The ℓ_1 -norm can be used for embedded feature selection by increasing the amount of shrinkage (λ) in the Lasso model, which effectively sets non-influencing components of \mathbf{w} to zero.

Due to their embedded feature selection ability, Lasso models have gained increasing attention for learning in sparse data sets, where the number of features is high, but many of them are irrelevant to the learning task [12]. Furthermore, in some applications, we want to include prior domain knowledge about relationships between features, for example if we know that motor speed and torque are depending on each other, they should also have similar influence (i.e. parameter weight) on the response variable. When features are represented in a graph structure, this quality is often called the smoothness property. The notion is as follows: If we specify relationships between features as *undirected* graph $G = \langle V, E \rangle$, the graph Lasso (Glasso) can be defined as

$$\begin{aligned} \hat{\mathbf{w}}_{GLasso} &= \operatorname{argmin}_{\mathbf{w}} l(y, h(\mathbf{w})) + \lambda(\alpha \|\mathbf{w}\|_1 + (1 - \alpha) \sum_{i,j \in E} (w_i - w_j)^2) \\ &= \operatorname{argmin}_{\mathbf{w}} l(y, h(\mathbf{w})) + \lambda(\alpha \|\mathbf{w}\|_1 + (1 - \alpha) \mathbf{w}^T L \mathbf{w}), \end{aligned} \quad (3)$$

where the second regularization term encourages connected features in the graph to have similar weights (smoothness). It can be preferably weighted against ℓ_1 by decreasing the α parameter. A more convenient formulation of the sum over squared weight differences is given by $\mathbf{w}^T L \mathbf{w}$, where L is the Laplacian matrix of the graph.

4 Approach

This section presents the main technical discussion of the developed semantic-guided feature selection approach. First, an introduction to concepts and axioms in the use case feature ontology is given, followed by a description of the semantic feature selection procedure. Ultimately, we present a custom linear model designed for embedded feature selection in RDF graphs.

4.1 Feature Ontology

There is a wide variety of modeling standards for manufacturing data that are used to facilitate interoperability of different systems, for example the exchange of material information between warehouse management and manufacturing execution systems. Recent developments of the OPC UA³ standard are concerned with a unified information model of field device descriptions, PLC programs, interfaces to enterprise levels such as ERP systems, and many more. Although these legacy data models describe several facets (e.g. device topologies, sensor measurements) of manufacturing systems, they are almost solely used for data exchange without taking advantage of their contained semantics.

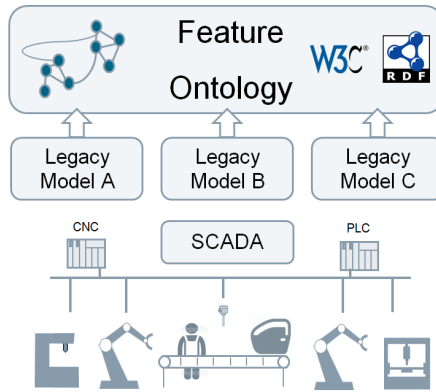


Fig. 5. Feature ontology on top of automation system legacy models

Instead of having another custom information model, our approach makes use of Semantic Web technologies that integrate existing semantics of legacy models into a unified ontology as shown in Figure 5. On top of the automation system and its supervisory control and data acquisition (SCADA), a feature ontology is deployed that represents domain concepts and relations between devices, events and information entities, such as taken sensor measurements. In this context feature means any piece of information that could be used as input to a learning algorithm. From an ontology engineering perspective, this is rather an ad-hoc modeling approach without strong axiomatization.

The result of what we call *Semantic Feature Selection*, i.e. inference about feature dependencies on a semantic level (rather than data level), is represented as RDF graph that contains a task-specific, reduced feature set that is tailored for consumption by machine learning models. These models are then able to perform preprocessing, training, and evaluation on the reduced feature set. One of the goals in development of the feature ontology was to keep the complexity of reasoning as small as possible so that modeling can be done with one of the *OWL 2* profiles that allow scalable reasoning as the number of features in

³ <https://opcfoundation.org/about/opc-technologies/opc-ua/>

the automation system grows. As discussed in section 3.1, the feature ontology references some concepts defined within the SSN ontology. In addition to that, we introduce some further relations concerning the connection between processes, devices and measurements in the manufacturing domain. Most importantly, we allow generic relations `dependsOn` and `independentOf` between features that subsume specific relations in existing engineering models, in case no further information is given.

Table 1. Main relations of the feature ontology

| Relation | Description |
|---|--|
| <code>derivedFrom</code> \sqsubseteq <code>directlyDependsOn</code> | <i>Connects an event or measurement that is derived from an original source, e.g. threshold overshoot events like 'temperature too high'</i> |
| <code>follows</code> \equiv <code>precedes</code> ⁻¹ | <i>Processes or Events that happen in a time-dependent order, e.g. packaging follows the assembly process</i> |
| <code>partOf</code> | <i>Partonomy describing device topologies, e.g. temperature sensor is part of a motor</i> |
| <code>directlyDependsOn</code> \sqsubseteq <code>dependsOn</code> | <i>A measurement is directly influencing another without further information, e.g. cycle time directly depends on failure events</i> |
| <code>physicalRelation</code> \sqsubseteq <code>directlyDependsOn</code> | <i>Measurements are connected by inherent physical laws, e.g. current and voltage</i> |
| <code>apartFrom</code> | <i>Events that happen at different locations, e.g. two assembly lines operating at separated shop floors</i> |
| <code>independentOf</code> | <i>A measurement is known to have no (direct) influence on the other, e.g. product ID does not influence conveyor motor temperature</i> |
| <code>observedBy</code> | <i>Measurement is sampled by a specific sensing device</i> |
| <code>observedAt</code> | <i>A measurement sampled during a specific process</i> |

Table 1 gives an overview of important relations that bear semantics for feature selection purposes. The independence statement is of statistic nature, therefore it is also a symmetric relation based on the fundamental probability theorem $P(X|Y) = P(X) \Leftrightarrow P(Y|X) = P(Y)$.

The $\mathcal{R}\text{Box}$ of the feature ontology further specifies some axioms that propagate dependencies through the feature space, as described in Table 2.

4.2 Feature Selection Procedure

Consider that we are given a learning problem of the form $y = f(x)$, where y is part of the semantic model, such that $\mathcal{O} \models \text{Response}(y)$, then the feature space of x can be reduced by excluding everything that is known to be independent of y . Furthermore, a good choice of features has to include anything that is known

Table 2. \mathcal{R} Box axioms of the feature ontology

| Axiom | Description |
|---|---|
| $\text{symmetric}(\text{independentOf}),$ $\text{symmetric}(\text{apartFrom})$ | <i>Independence and separation of processes are defined to be symmetric</i> |
| $\text{dependsOn} \cdot \text{observedBy} \cdot \text{observedBy}^{-1} \sqsubseteq$ dependsOn | <i>Dependencies propagate between measurements observed by the same sensing device</i> |
| $\text{independentOf} \cdot \text{dependsOn} \sqsubseteq$ independentOf | <i>If x is independent of y it is also independent of anything that y depends on</i> |
| $\text{observedAt} \cdot \text{apartFrom} \cdot \text{observedAt}^{-1} \sqsubseteq$ independentOf | <i>Assert independence of measurements taken at physically separated processes</i> |
| $\text{transitive}(\text{partOf}), \text{transitive}(\text{follows})$ | <i>Process flows and device topologies are transitive by nature</i> |

to directly influence the behavior of the response variable. Hence, the \mathcal{T} Box is accordingly augmented with the following axioms:

$$\begin{aligned} \mathcal{T} = & \\ & \exists \text{independentOf}.\text{Response} \sqsubseteq \text{ExcludedFeature} \\ & \exists \text{directlyDependsOn}^{-1}.\text{Response} \sqsubseteq \text{MandatoryFeature} \\ & \text{ExcludedFeature} \sqcap \text{MandatoryFeature} \sqsubseteq \perp \end{aligned}$$

Overlap in excluded and mandatory features results in an inconsistent feature ontology that is most likely due to a feature modeling mistake, since there should not be independence and direct dependence for two information entities at the same time.

Algorithm 1 summarizes this schema-level feature selection procedure. First, the assertion of y as an individual of the class **Response** must be given. Further classification of individuals is done by a standard *OWL 2* reasoner (e.g. Hermit⁴). In case of an inconsistency, i.e. disjointness of mandatory and excluded features can not be satisfied, the procedure exits. Otherwise, the sets of excluded features and mandatory features are collected, respectively. Finally, the algorithm returns the set of mandatory features and the set of optional features for the learning task. Note that this can be done without looking at the data, but by relying on engineering domain knowledge. After applying *Semantic Feature Selection* learning tasks such as cycle time forecasting can be employed with the reduced set of mandatory and optional features.

The main advantages of our approach in comparison with common feature selection procedures are summarized in Table 3.

⁴ <http://hermit-reasoner.com/>

Algorithm 1. Semantic Feature Selection

```

Input : Learning problem  $y$ , feature ontology  $\mathcal{O}$ , feature space  $\mathcal{F}$ 
Output : Mandatory features  $\mathcal{S}_m$ , optional subset  $\mathcal{S}_o$ 
 $\mathcal{A} \leftarrow \text{Response}(y)$  ▷ Instantiate response variable
 $\mathcal{S}_o \leftarrow \mathcal{F}$ 
 $\mathcal{S}_m \leftarrow \emptyset$ 
if  $\mathcal{O} \models \text{Dis}(\text{ExcludedFeature}, \text{MandatoryFeature}) \sqsubseteq \perp$  then
    return ▷ Unsatisfiable disjointness, inconsistent ontology
end if
for  $x_i \in \{x \mid \mathcal{O} \models \text{ExcludedFeature}(x)\}$  do
     $\mathcal{S}_o \leftarrow \mathcal{S}_o \setminus x_i$ 
end for
for  $x_i \in \{x \mid \mathcal{O} \models \text{MandatoryFeature}(x)\}$  do
     $\mathcal{S}_m \leftarrow x_i$ 
end for
return  $\mathcal{S}_o, \mathcal{S}_m$ 

```

Table 3. Advantages of semantic feature selection

| Criterion | Today | Our approach |
|------------------------|--|--|
| Complexity | Grows with dimensionality and size of data sets | Grows only with dimensionality (i.e. new feature entities) |
| Re-usage | Need to be re-executed for every incoming instance | Needs only re-execution if new features are added |
| Intepretability | Reducing feature spaces often loses intuitive interpretability | Explicitly focuses on facilitated human interpretation |

4.3 Graph Kernel Lasso

In some cases, data sets of automation systems are sparse. For example, if we want to forecast cycle times of rarely produced products, there will only be very few instances available for training. For better learning model performance it would be beneficial to further consider the semantics of the feature space during model training. In order to tackle this problem, we present a technique that integrates semantic dependencies into linear model learning and simultaneous feature selection.

In contrast to the standard graph Lasso defined in (3), where a relation between two features is either present or not, i.e. the graph’s adjacency matrix $A_{i,j} \in \{0, 1\}$, we want to use a more enhanced notion of dependencies that also takes semantics into account. In our application, we use RDF-graph kernels to capture similarities between entities in the manufacturing feature ontology. In reference to (3), we therefore define a graph kernel-weighted regularization term that encourages smoothness between similar entities in the RDF graph of the feature ontology.

$$\Omega(\mathbf{w}) = \alpha \|\mathbf{w}\|_1 + (1 - \alpha) \sum_{i,j \in V} \kappa(G_i, G_j)(w_i - w_j)^2 \quad (4)$$

where G_i and G_j are the spanned graphs of entities i, j in the vertex set of the whole RDF graph V and κ is some RDF graph kernel. It is easy to see that the second regularization term can be expressed as weighted Laplacian L_κ .

$$\sum_{i,j \in V} \kappa(G_i, G_j)(w_i - w_j)^2 = \mathbf{w}^T L_\kappa \mathbf{w} \quad (5)$$

with $L_\kappa = \text{diag}(r_1, r_2, \dots, r_p) - \kappa$, and r_i denoting the i th row sum of the kernel matrix κ . We refer to this model as the graph kernel Lasso (*GraKeLasso*). Note that if the kernel matrix is set to the identity matrix, this model is equivalent to the ElasticNet [14].

As mentioned above, the regularization penalty induces a smoothing, or grouping in a sense, that features that are similar with respect to the feature ontology graph have similar parameter values. Intuitively, if two features are closely related, e.g. torque and speed measurements of a conveyor motor, they should have a similar influence (i.e. signal) on the response variable. Additionally, if one feature turns out to be irrelevant, all its closely related features are also very likely to be irrelevant. The graph kernel Lasso model enforces both of these properties.

For our use case application, we can resort to a wide variety of graph kernels, such as the implementations of the *mustard* framework⁵.

A visual representation of the feature ontology graph kernels from our use case data set is given in Figure 6(a) for the full feature space on the left-hand side and (b) for the reduced feature space after semantic feature selection on the right-hand side. Every feature is represented by a segment on a circle the width of the chords connecting two features depicts the strength of similarity between the two. It can be seen that the full feature space contains some very dominant feature similarities, while the reduced feature space exhibits a more uniform distribution.

5 Evaluation

To show the value of semantic guidance in feature selection and the custom Lasso model, we evaluated the performance of forecasting cycle times, as sketched in the manufacturing use case scenario. The regression models are trained on different data sets generated by a discrete-event simulation model that conforms to the manufacturing process in section 2.

We compare five different regression models for the cycle time estimation task: Lasso, ElasticNet, Graph Lasso, *GraKeLasso*, and OLS. For *GraKeLasso*, we used a subtree-based variant of the Weisfeiler-Lehman graph kernel, whereas Graph Lasso incorporates only information about feature individuals connected via `dependsOn`, i.e. a simple dependency graph.

⁵ <https://github.com/Data2Semantics/mustard>

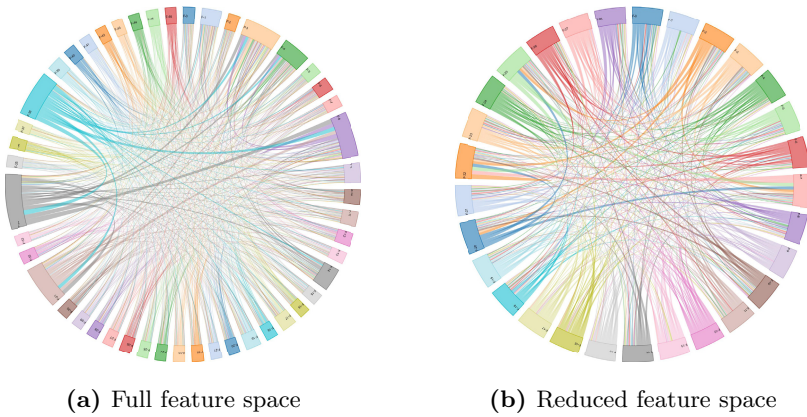


Fig. 6. Visualization of pairwise graph kernel weights between features. (a) Full feature space, (b) Reduced feature space after semantic feature selection

Set Up. Our simulation set up comprises of a source that generates two different product types at a specified time interval, each of which has a different distribution for weight and size. These products are sent to two separated assembly lines, where first a loading station measures product qualities using a balance and a barcode scanner. For the conveyor we monitor its electric motor (power, torque, speed, temperature) and some induced failure events. The simulated quality control again measures sizes of the products. Finally, the packaging station samples the cycle times we want to forecast. Each station further observes its current workload and operating timestamp (seen as soft sensors).

Overall, the final feature ontology consists of 6 processes (one additional for the separated assembly), 18 sensing devices that sample 47 different measurements, i.e. feature individuals. In addition to that there are 13 concrete instantiations of relations between two features.

Table 4. Original and reduced variants of product cycle time data set

| Data Set | n | p | Reduction | OLS CV RMSE |
|-----------------------------|------|----|---------------|-------------------------------|
| Cycle time full | 2000 | 47 | - | $\approx 1.36 \times 10^{11}$ |
| Cycle time semantic reduced | 2000 | 29 | 38.3 % | 0.08 |
| Cycle time p-value reduced | 2000 | 18 | 61.7 % | 0.06 |
| Cycle time sparse | 40 | 47 | - | 9.49 |

Data Sets & Results. Starting from the original cycle time data set, we obtain three additional variants for evaluation purposes. Table 4 depicts their individual

Table 5. Embedded model performances on 10-fold cross validation

| Data Set | Model | Reduction | CV RMSE |
|-------------------|-------------------|---------------|-------------|
| Cycle time full | Lasso | 47.4 % | 0.42 |
| | ElasticNet | 34.4 % | 0.57 |
| | Graph Lasso | 4.5 % | 0.32 |
| | <i>GraKeLasso</i> | 4.9 % | 0.32 |
| Cycle time sparse | Lasso | 51.3 % | 0.48 |
| | ElasticNet | 8.7 % | 0.46 |
| | Graph Lasso | 8.9 % | 0.54 |
| | <i>GraKeLasso</i> | 6.8 % | 0.43 |

characteristics. The full data set consists of 47 dimensions and 2000 instances, while in the sparse case, the number of instances is kept to 40 so sparseness is preserved. After applying the reasoning procedure presented in Algorithm 1, the number of features p reduces to 29 – approximately 38 % reduction. On the other hand, a common p-value based selection reduces dimensionality to 18 (at 0.05 significance threshold). This means that there are many linear dependencies in the original data set which can be eliminated by pairwise correlation. The semantic approach does not eliminate dependencies by correlation, but excludes features that are inferred to be independent of the response variable by means of the feature ontology. For each of the data sets an OLS model is trained and evaluated with respect to the coefficient of variation of the root-mean-square error (CV RMSE) in cycle time seconds. It can be seen that best performance is given for the p-value reduced data set, however, the semantic reduced data set shows competitive results.

The embedded feature selection models are evaluated in a similar setting. Model performances shown in Table 5 correspond to the overall best value determined by a grid search over λ , whereas $\alpha \in]0, 1]$ was set to the best of an inner cross validation, respectively. Final performance results are again averaged over 10-fold cross validation. The reduction column also reports on each of the embedded model’s average feature selection capabilities, i.e. number of zero valued coefficients.

Discussion. Overall, our results indicate two main insights. First, compared to p-value based selection, which does a better job at reducing dimensionality, semantic feature selection shows competitive performance in a sense that it keeps the needed features in its original form without any data-intensive computations. Interestingly, after upfront feature selection the ordinary least squares model outperforms all the other approaches for this setting and yields the overall lowest error. Second, our *GraKeLasso* shows best performance on the full and the sparse data set, because it can take similarities of the whole feature space into account. In summary, it can be seen that both of our approaches effectively decrease prediction errors and show competitive or even superior performance compared to conventional techniques. Due to this limited simulation scenario, we could not

show that a combination of both approaches is beneficial. Further evaluations on large-scale systems, when upfront feature selection alone does not suffice, are necessary to investigate this.

6 Related Work

Due to the plethora of research concerned with feature selection, we will only present related works that are closely connected to the one in this paper. For a general overview of the field, we refer to the survey paper [4].

Coming from a semantic perspective on feature selection, the technique introduced by [6] describes a fuzzy approach to capture implicit semantics of data sets in order to reduce their dimensionality. They apply fuzzification on the degree of which features are dependent based on their co-occurring consistency with the decision variable. However, this technique does not rely on any explicit semantics defined in the data model and also needs preferably access to the full data set. In the field of biomedical machine learning applications, considering domain knowledge in feature selection has been studied and shown that feature spaces for association rules can be greatly reduced when medical domain knowledge about concepts is introduced, see [1]. The authors introduce dependencies between medical concepts, such as diseases and treatments, in order to learn more compact association rules. Another kind of related work that has been studied with increasing interest is the family of Lasso regularization models. Algorithms like the GOSCAR have been applied to consider dependency knowledge between genes in DNA sequences as penalty for group regularization in graph Lasso models. These models have shown to increase classification accuracy in several studies, e.g. [11]. Similarly, for image classification semantic dependencies between labeled images have been integrated into a Lasso approach [2].

In summary, including semantics into feature selection has been the concern of very few research studies outside of text and image processing domains, where semantics are mostly given through natural language. However, there are certain knowledge-based approaches that argue for the dynamic adaption of features to account for changes in the data generation process [9].

7 Conclusion and Future Work

In this work, we introduced the application of semantic feature selection for machine learning models in modern industrial automation systems. Only few works have been concerned with the usage of explicit semantics, in order to facilitate feature selection, which still remains one of the main issues. Especially in the manufacturing domain, there are many known dependencies between measurements that are cut out to guide this process. In this paper, we show how a small amount of semantic relations can be used to significantly reduce the size of feature spaces for exemplary learning problems in manufacturing systems and still yield good performance. Furthermore, we presented an embedded feature

selection for linear models that captures feature similarities within RDF graphs and outperforms conventional approaches when applied to sparse data sets.

In future work, we plan to implement the developed approach in a real-life automation system. A particular promising direction seems to be the conjunction of this approach within OBDA systems, where ontologies are used to retrieve instance data. By deploying machine learning on top of OBDA, a coupling of our approach and ontology-based queries could reveal some synergy effects.

References

1. Blake, C., Pratt, W.: Better rules, fewer features: a semantic approach to selecting features from text. In: Proc. of IEEE Int. Conf. on Data Mining, pp. 1–8 (2001)
2. Chen, X., Yuan, X., Yan, S., Tang, J., Rui, Y., Chua, T.S.: Towards multi-semantic image annotation with graph regularized exclusive group lasso. In: Proc. of 19th ACM Int. Conf. on Multimedia - MM 2011, pp. 263–272 (2011)
3. de Vries, G.K.D.: A fast approximation of the Weisfeiler-Lehman graph kernel for RDF data. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part I. LNCS, vol. 8188, pp. 606–621. Springer, Heidelberg (2013)
4. Guyon, I.: An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res. (JMLR)* **3**, 1157–1182 (2003)
5. Jenatton, R., Audibert, J.Y., Bach, F.: Structured Variable Selection with Sparsity-Inducing Norms. *J. Mach. Learn. Res. (JMLR)* **12**, 2777–2824 (2011)
6. Jensen, R., Shen, Q.: Semantics-preserving dimensionality reduction: Rough and fuzzy-rough-based approaches. *IEEE Transactions on Knowledge and Data Engineering* **16**(12), 1457–1471 (2004)
7. Jirkovsky, V., Obitko, M., Novak, P., Kadera, P.: Big data analysis for sensor time-series in automation. In: Proc. of Emerging Technology and Factory Automation (ETFA), pp. 1–8 (2014)
8. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for RDF data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 134–148. Springer, Heidelberg (2012)
9. Ringsquandl, M., Lamparter, S., Lepratti, R.: Context-aware analytics in MOM applications. In: Workshop Notes of the 6th International Workshop on Acquisition, Representation and Reasoning about Context with Logic (2014)
10. Rodríguez-Muro, M., Kontchakov, R., Zakharyashev, M.: Ontology-based data access: ontop of databases. In: Proc. of the 12th Int. Sem. Web Conf. (2013)
11. Yang, S., Yuan, L., Lai, Y.c., Shen, X., Wonka, P., Ye, J.: Feature grouping and selection over an undirected graph. In: Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (KDD), pp. 922–930 (2012)
12. Ye, J., Liu, J.: Sparse Methods for Biomedical Data. *SIGKDD explorations* **14**(1), 4–15 (2012)
13. Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proc. of the 20th Int. Conf. on Mach. Learn., pp. 1–8 (2003)
14. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society* **67**, 301–320 (2005)

A Semantic Processing Framework for IoT-Enabled Communication Systems

Muhammad Intizar Ali¹ (✉), Naomi Ono¹, Mahedi Kaysar¹, Keith Griffin²,
and Alessandra Mileo¹

¹ INSIGHT Centre for Data Analytics, National University of Ireland,
Galway, Ireland

{ali.intizar,naomi.ono,mahedi.kaysar,alessandra.mileo}@insight-centre.org

² Cisco Systems, Galway, Ireland

kegriffi@cisco.com

Abstract. Enterprise Collaboration Systems are designed in such a way to maximise the efficiency of communication and collaboration within the enterprise. With users becoming mobile, the Internet of Things can play a crucial role in this process, but is far from being seamlessly integrated in modern online communications. In this paper, we showcase the use of a solution that goes beyond today's ad-hoc integration and processing of heterogeneous data sources for static and streaming data, providing more flexible and efficient processing techniques that can bridge the gap between IoT and online Enterprise Communication Systems. We document the technologies used for sensor deployment, sensor data acquisition based on the OpenIoT framework, and stream federation. Our main contributions are the following, i) we present a conceptual architecture of IoT-enabled Communication Systems, that builds upon existing frameworks for semantic data acquisition, and tools to enable continuous processing, discovery and federation of dynamic data sources based on Linked Data; ii) we present a semantic information model for representing and linking IoT data, social data and personal data by re-using and extending the existing standard semantic models; iii) we evaluate the performance of virtualisation of IoT sources based on OpenIoT in our testbed and show the impact of transmission, annotation and data storage, as well as initial results on scalability of RDF stream query processing in such a framework, providing guidelines and directions for optimisation.

Keywords: IoT · RDF stream processing · Stream federation · Communication systems · OpenIoT · Linked data

1 Introduction

Enterprise communication systems currently and historically have been primarily aimed at person to person communication. Users of such systems typically

This research is sponsored by Science Foundation Ireland (SFI) grant No. SFI/12/RC/2289 and Cisco Systems.

interact with an endpoint such as a phone, video system or unified communications software client capable of multi-modal communications. Communication modes typically consist of instant messaging, voice, video and voicemail to allow individuals or groups to communicate in real time. Such systems have not historically enabled open machine to machine or machine to person communication. The emergence of Internet of Things (IoT) provides the potential to enable communication between sensory devices and communication systems using open interfaces, but this potential is under investigated and few solutions have existed in isolation. As a result, the flexible integration of a large amount of multi-modal data streams from diverse application domains is still one of the key challenges in developing IoT-enabled communication systems.

The lack of interoperability results into the inability for such systems to integrate information from external sources in an easy and cost-effective way. This issue becomes more evident if we consider advances in the IoT space, which demands dynamic and flexible exchange of information between IoT sources. To overcome these interoperability issues in communication systems and across smart enterprise applications towards IoT-enabled solutions, we developed a Linked Data infrastructure for networking, managing and analysing streaming information. In order to ensure high reusability, we leveraged existing semantic models for the annotation of sensor data (e.g. SSN), social web (e.g. FOAF) and personal information (e.g. PIMO), and extended the ontological model to incorporate personal, business and online communication concepts.

In order to set the basis for our evaluation, we identified a usecase scenario in the enterprise communication space, to illustrate the potentials of IoT-enabled Communication Systems. We then designed and developed the processing pipeline from IoT sources to stream processing and reasoning, which is seamlessly integrated in our framework. Our main contributions in this paper include:

- design of a Linked Data framework for IoT-enabled smart enterprise applications that connects physical to virtual sensors and enables scalable stream processing and reasoning;
- interoperable integration of various IoT sources (corresponding to capabilities) in the context of an open source online communication system;
- demonstration of the effectiveness of our proposed framework based on a concrete instance of OpenIoT and Apache Open Meetings;
- experimental validation of the performance and scalability of our IoT-enabled infrastructure and lessons learned.

The remainder of this paper is organised as follows: Section 2 presents our scenario and state of the art, Section 3 details our IoT-enabled Linked Data infrastructure and its software components, which we evaluate in Section 4 before we conclude with some remarks and lessons learned in Section 5.

2 Motivation and State of the Art

Sensor technologies and sensory devices are nowadays part of our everyday lives. The Internet of Things (IoT) not only provides an infrastructure for sensor deployment, but also a mechanism for better communication among connected sensors. Data generated by these sensors is huge in size and continuously produced at a high rate. This requires mechanisms for continuous analysis in real-time in order to build better applications and services. Data streams produced by various sensors can be classified into three different categories, namely, (i) *Physical (static) Sensors*, (ii) *Mobile & Wearable Sensors*, and (iii) *Virtual Sensors & Social Media Streams*.

Among the above three categories, mobile sensors are harder to integrate within enterprise communication systems. This is not only due to technical integration issues and interoperability, but also due to their dynamic nature and constantly changing context. Mobility and location-based sensory input, for example, result into a higher level of unpredictability and lower level of control over the distributed infrastructure that characterises enterprise communication systems. These challenges are matched by new opportunities for IoT-enabled collaboration and communication systems to be designed in order to sense the context of a mobile user and take decisions according to dynamic sensory input. In the domain of enterprise communication systems, mobile users have the potential to produce a lot of dynamic sensory input that can be used for the next generation of mobile enterprise collaboration, with great potentials for better user experience. In this paper we propose a framework and a set of software component for IoT-enabled online meeting management that combine existing technologies in a scalable infrastructure.

2.1 Motivating Scenario: IoT-Enabled Meeting Management System

Alice is hosting an online meeting for her company *FictionDynamic*. The meeting is planned to hold in Meeting Room B at 11:00 am. Bob and Charlie attending the meeting while they are on the move, thus their availability and ability to participate to the meeting in various ways is dynamically changing. The IoT-enabled Meeting Management System (IoT-MMS) enables i) automatic on-the-fly semantic enrichment of IoT information related to the meeting attendees, ii) communication of such richer information to the participants via their IoT-MMS client through a panel showing IoT values and related user capabilities (e.g. ability to hear properly, share a screen, type, talk), iii) use of such rich information to improve user experience and optimise meeting management on-the-fly. The integration of a web-based MMS with sensory input and enterprise data such as attendees details, calendars and agenda items makes it possible to characterise and manage the following aspects in a flexible and interoperable way:

- updating (enabling or disabling) users capabilities based on IoT input (via sensors abstraction and interpretation, semantic integration and stream query processing);

- managing agenda items, including users involved and capability requirements via business logic rules;
- dynamically verifying privacy-constraints on agenda items based on location and context.

In Sections 3 and 4, we illustrate the design and implementation of our IoT-MMS framework enabling characterisation and management of the above mentioned aspects.

Enabling and disabling user capabilities has the potential of improving user experience: acting on microphones and speakers of attendees based on their participation and the level of noise can avoid unpleasant feedback loops, and guidance for the meeting host on changing capabilities of attendees on the move would promote more effective management of online meetings. In the same way as capabilities are enabled or disabled, additional functionalities can be characterised by adding specific semantic queries and action triggers. For example, the IoT-MMS can support the meeting host in dynamically re-assigning agenda slots to participants, based on users involved and their changing capabilities. Also, queries over the attendees calendars and presence status [10] for availability would make it possible to suggest alternative meeting slots if key attendees become unavailable or if their capabilities become compromised.

2.2 State of the Art

Internet of Things (IoT) research in recent years has focused on modelling domain knowledge of sensor networks and services [3, 4, 12, 16]. The Semantic Sensor Network (SSN) ontology is one of the most significant efforts in the development of an information model for sensory data [6]. The SSN Ontology provides a vocabulary for expressive representation of the sensors, their observations and knowledge of the surrounding environment¹. SSN is being widely adopted by many IoT-based applications for the representation of sensor data. SSN ontology defines only a high-level scheme of sensor systems, therefore SSN alone cannot represent an information model for a richer IoT infrastructure and needs to be aligned with the existing ontologies or with new concepts from application domains. Consider our scenario in Section 2.1, the SSN ontology needs to be aligned with existing semantic models for the representation of the meeting/calendar information and personal/social data.

Data acquisition from distributed heterogeneous sensors is another essential aspect of IoT-enabled applications. The Global Sensor Network (GSN) middleware facilitates flexible integration and discovery of sensor networks and sensor data [1], enabling fast deployment and addition of new IoT platforms by supporting dynamic adaptation². X-GSN [5] is an extension of GSN and therefore supports all virtual sensors and wrapper developed for the GSN middleware. X-GSN is deployed as a web server which continuously listens for sensor data over

¹ <http://www.w3.org/2005/Incubator/ssn/ssnx/ssn>

² <http://sourceforge.net/projects/gsn/>

a pre-configured port (default port = 22001), and it contains various wrappers built as subclasses of the GSN wrappers, each acting as a thread in the GSN.

OpenIoT [2] is an open source middleware for collecting information from sensor clouds. OpenIoT can collect and process data from virtually any sensor in the world, including physical devices, sensor processing algorithms and social media processing algorithms (<http://openiot.eu>). OpenIoT combines and enhances results from leading edge middleware projects, such as the Global Sensor Networks - GSN and the Linked Sensor Middleware³ (LSM) [1,9].

However, IoT-enabled applications not only require to gather sensor data from distributed sensors network, but also demand to provide adaptive applications which can query data streams generated by sensors and can take smart decisions accordingly. Furthermore, IoT-enabled applications need to provide robustness because of the autonomous and distributed nature of the underlying architecture. OpenIoT in its current state does not support stream query processing over data streams generated by various sensors, hence lacking the ability to facilitate realtime decisions. We used the OpenIoT framework for sensor data acquisition and semantic annotation, creating additional wrappers that are needed for streaming IoT data, and we extended it by introducing stream query processing [8] and stream reasoning capabilities based on rules [11].

3 IoT-enabled Communication Systems

In this section, we introduce the layered architecture of the IoT-enabled Communication System and briefly describe each of the layers involved in the processing pipeline.

3.1 Application Architecture

Figure 1 illustrates our conceptual architecture for IoT-Enabled Communication System. OpenIoT acts as a core component for data acquisition and semantic annotation of the data produced by various sensors. We extended the functionalities of the OpenIoT platform by introducing HTTP Listener wrapper for capturing streaming data, and semantic querying and reasoning layer, which allows IoT-enabled communication systems to include semantically annotated data streams produced by sensors as an additional source information. IoT-enabled Communication Systems can perform real-time continuous queries over data streams and consume the results of these queries to take context-aware and user-centric decisions in real-time. As shown in Figure 1, there are three main layers involved in our IoT-enabled Enterprise Communication System architecture, namely (i) *Data Acquisition and Semantic Annotation Layer*, (ii) *Stream Processing and Reasoning Layer*, and (iii) *Application Layer*. Below, we further elaborate on each of these layers and their components.

³ <http://lsm.deri.ie>

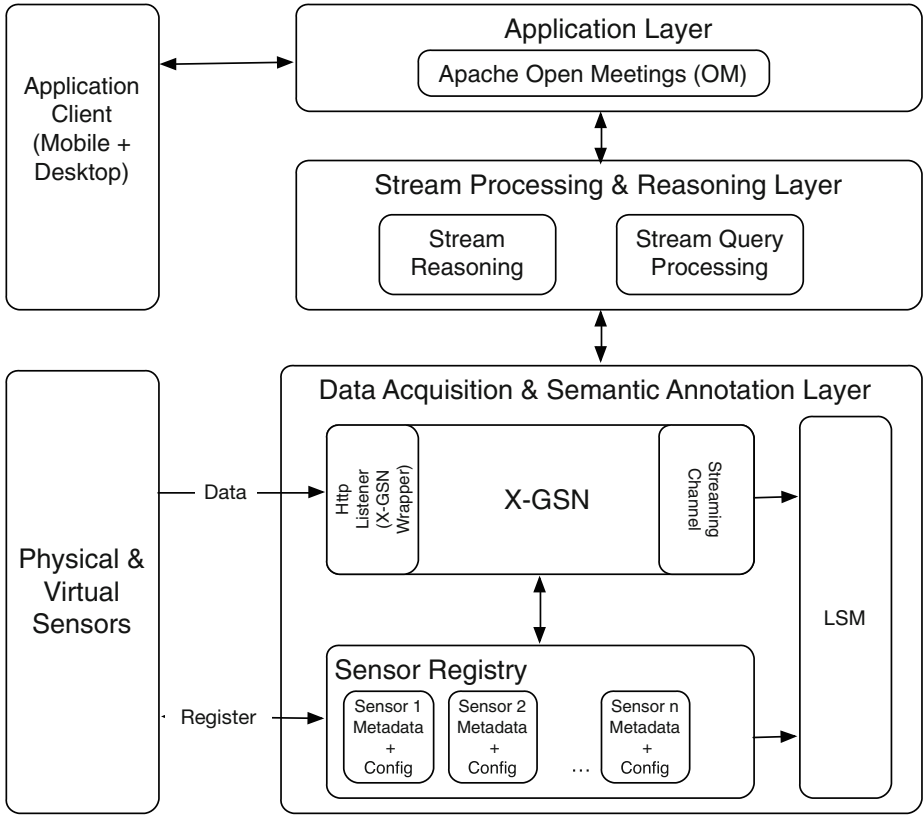


Fig. 1. IoT-Enabled Communication System Architecture

3.2 Data Acquisition and Semantic Annotation Layer

This layer is mainly responsible for acquiring sensor data from mobile devices and performing semantic annotation of the acquired data using our information model. We briefly discuss each of the components and their functionalities.

Data Acquisition

Our proposed architecture can acquire data from any type of sensor, whether it is physical sensor deployed at a fixed location, a mobile sensor or even a virtual sensor representing virtual data streams (e.g. social media data streams). However, considering the IoT-MMS scenario presented in Section 2.1, we focus on data acquisition from mobile sensors only.

Mobile Application for Data Acquisition: In order to receive data from various mobile sensors, we developed an android base application which can continuously sense the information from a mobile device. Once, the application is launched, a registered user can choose the sensors for which he/she wants to

share the data. Data produced by the selected sensors is continuously sent to the OpenIoT server.

Sensor Registration: A sensor is considered as a basic entity in the OpenIoT platform. Each and every sensor participating within the framework should be registered in the OpenIoT Platform before sending any observation. Sensors are uniquely identified within the OpenIoT platform by assigning a unique id. Mobile devices are registered as platforms (ssn:platform⁴), which can host multiple sensors. During the sensor registration process, some meta information (e.g. type of sensor, owner of the device, sensor observation unit etc.) is acquired. Sensors can be either registered individually and associated to an individual user or they can be registered in bulk if multiple sensors have the same meta information attached to them.

Sensor Observations Transmission: Whenever a sensor is successfully registered in the OpenIoT platform and the mobile application for data acquisition is launched using any mobile device, all the selected sensors on that particular device start transmitting their observations to the OpenIoT platform. Our processing pipeline makes it possible to select and de-select sensors dynamically without re-launching the application. We developed an X-GSN wrapper for mobile data acquisition, which is deployed over the X-GSN Server. As shown in Figure 1, the *Http Listener* is an integral part of the *X-GSN Wrapper* which continuously listens for the sensor observations. As soon as any observation is received, this layer starts processing the data accordingly using meta information of that particular sensor from which the observation is acquired. *X-GSN* also includes a *Streaming Channel* component which publishes semantically annotated RDF streams.

Semantic Annotation

We reused and integrated different semantic models for the representation of all acquired information in our IoT-MMS scenario, including sensor metadata, sensor observation, meeting/event data, meeting attendees and their capabilities. Linked Data representation allows for easy integration of semantic information collected from heterogeneous data streams as well as integration with static knowledge to perform querying and reasoning.

Semantic Annotation of Sensor Data Streams: We used the SSN ontology for representing sensors, their observations, and their platform (mobile device). The OpenIoT platform carries out annotation of the virtualised data streams that have been provided by the X-GSN data wrappers. We used an information model to explicitly define semantics of sensory data such as sensor types, models, methods of operation and common measurement definitions. As a result, sensor capabilities can be defined in accordance with existing conditions and be integrated as Linked Data. This helps bridging the gap between real-time information generated from various independent data sources and a huge amount of

⁴ [http://purl.oclc.org/NET/ssnx/ssn/\\$#\\$Platform](http://purl.oclc.org/NET/ssnx/ssn/$#$Platform)

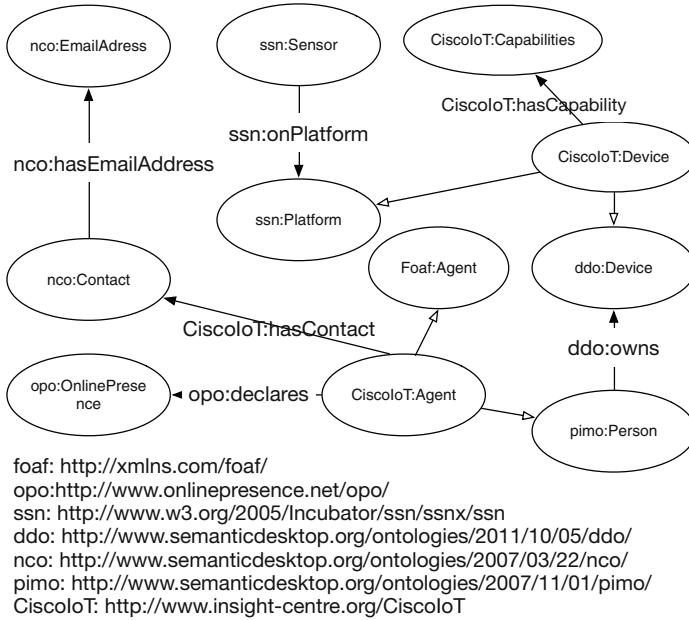


Fig. 2. Information Model - Device and Contact

interconnected information already available over the Web. For example, sensors and their data can be linked to geographic data (e.g. correlated natural phenomena), user-generated data (e.g. Meeting Data), and some implicit information (e.g. user profiles, calendar) through our semantic driven approach.

Semantic Annotation of Application Users and Mobile Devices: SSN is a de-facto standard for semantic annotation of sensor data streams. However, it still lacks the information to associate data generated from the sensors with any particular owner or user of that particular sensor. Keeping the usecase scenario of the IoT-MMS in mind, we represent the mobile client user as an owner of sensors embedded in the particular mobile device the user has used to log-in to the IoT-MMS mobile client. We used the NEPOMUK Contact Ontology (nco) [7] to represent a user and his/her contact information, while we used Digital.Me Device Ontology (ddo) to associate a device with any particular user [15]. As described earlier, multiple sensors embedded within a single mobile device can be easily represented using ssn:platform concept. Figure 2, depicts the information model for integration/linkage of sensor data with the contact information of the user as well as the device hosting that particular sensor. Each device can have multiple capabilities (e.g. noise, light, proximity) depending on the available sensors embedded within the device.

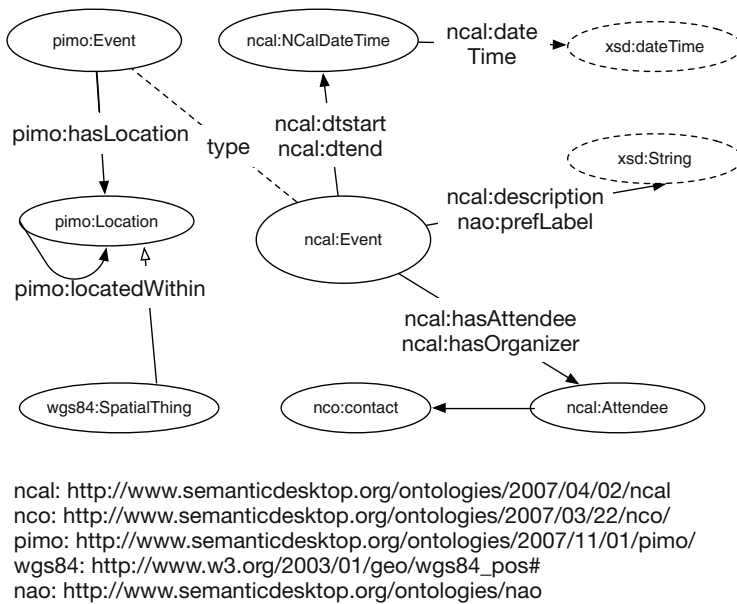


Fig. 3. Information Model - Meeting Management

Semantic Annotation of Meeting Data: We used the NEPOMUK⁵ and related semantic desktop ontologies⁶ for semantic representation of meetings, their description, organiser, list of attendees, location, starting time and duration [14]. NEPOMUK Calendar Ontology (nca) is used for semantic annotation of the meetings created by any user of the IoT-MMS. Figure 3 gives an overview of the information model for the semantic annotation of meeting data.

3.3 Stream Processing and Reasoning Layer

One of the most important factors for IoT-enabled applications is their ability to detect events within minimal time delay. The *Stream Query Processing* component -shown in Figure 1- enables to continuously query sensor data streams and detect events in realtime, while the *Stream Reasoning* component contains application logic to make smart decisions customised to the particular requirements and context of the user.

Stream Query Processing: We integrated the CQELS (Continuous Query Evaluation over Linked Streams) query engine for the execution of continuous queries over semantically annotated data streams of mobile sensors [8]. CQELS is a state of the art stream query processing engine for RDF data streams, which allows to register queries over sensor data streams. Once a query is registered, CQELS continuously monitors sensor data streams and produces a stream of

⁵ <http://nepomuk.semanticdesktop.org>

⁶ <http://www.dime-project.eu>

results matching the query patterns. Listing 1, shows a CQELS query to monitor the noise level of a certain user of the IoT-MMS.

Stream Reasoning: This components consumes the stream generated as a result of the CQELS queries and facilitates smart decisions by associating patterns of events to actions. This is modelled using event-condition-action (ECA) rules in AnsProlog, where the events are triggers for actions to be executed. For example, results of the CQELS query in Listing 1 can be used by the stream reasoning component to trigger a rule that, based on the noise level, mutes a single or multiple users whenever noise level surpasses the specified threshold, and different thresholds can be dynamically selected based on indoor or outdoor user location. Similarly, rules can be used to suggest changes to the agenda when the associated attendee is late or temporarily disconnected, or to warn attendees on certain privacy threats when they are in a public place like an airport lounge or a train.

```

prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
prefix lsm: <http://lsm.deri.ie/ont/lsm.owl#>
select ?noiseValue
WHERE {
STREAM <http://lsm.deri.ie/resource/1409752298064700000> [RANGE 5s]
{
?ob_5 rdf:type ssn:Observation.
?value_5 ssn:observedProperty <http://lsm.deri.ie/resource/1409752298163783000>.
?ob_5 ssn:featureOfInterest ?foi.
?value_5 lsm:isObservedPropertyOf ?ob_5.
?value_5 lsm:value ?noiseValue.}
}

```

Listing 1. A Sample CQELS Query to Monitor Noise Level

3.4 Application Layer

This layer represents the class of enterprise applications that can benefit from IoT intelligence which we showcase using our IoT-enabled Communication System based on Apache OpenMeetings. We extended the OpenMeetings server to generate semantically annotated data and to communicate with the reasoning component of our framework by continuously observing the status of relevant sensors generated by the stream processing layer and take appropriate actions at the application layer.

In what follows, we describe the process flow of our IoT-enabled OpenMeetings extension and illustrate the concepts and implementation of our online Meeting Management solution.

OpenMeetings (OM) is an open source software used for web conferencing, collaborative white board drawing, document editing etc. It uses OpenLaszlo RIA framework for client side generation and Red5 streaming server for remoting and streaming. In a physical conference room it can use Real-Time Messaging Protocol (RTMP) for high performance transmission of video, audio and data between Flash and the server. RTMP is a TCP based protocol which keeps the persistence connection and allows low latency communication.

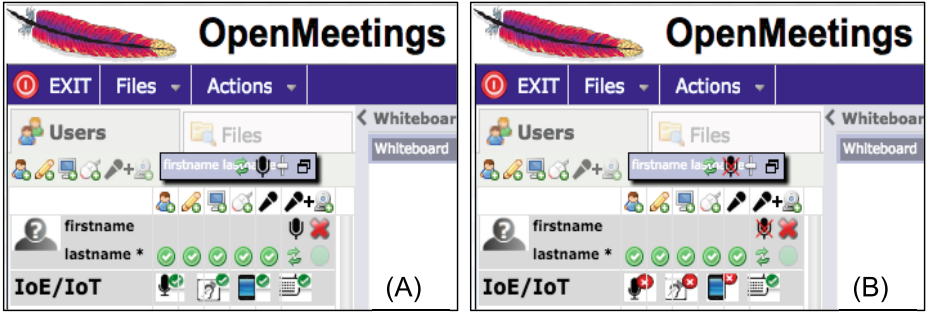


Fig. 4. Snapshots of IoT-enabled OpenMeetings Client

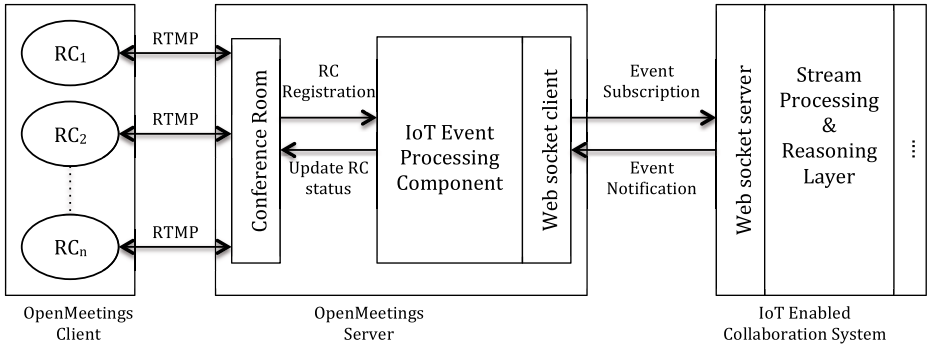


Fig. 5. Process Flow of IoT-enabled OpenMeetings System

Meeting Management in online communication and collaboration systems like OpenMeetings is enhanced with IoT input using our framework. In order to do that, the status of sensors registered on the IoT platform is monitored by the *Stream Reasoning* component, which identifies their status and determines appropriate actions based on rules. The sensors we considered include noise, proximity, light and location, while the actions are related to changing the status of a set of capabilities. Capabilities illustrate real-time availability of the participants to perform certain actions including talking, listening, reading a display or typing, and they are represented in the application control panel as a new set of IoT-related icons. Based on thresholds on the value of readings from specific IoT sources, the status of these icons is automatically updated from green to red or vice versa, indicating whether a participant can perform the corresponding activity or not. This provides the meeting host with updated information on the capability of the attendees, and can further be used to act on specific actuators (e.g. muting a microphone). For example, the ability to read the screen is not active if a user is connected via phone and answers a phone call. Figure 4(A) shows a client with all capabilities active, while Figure 4(B) shows the configuration for a mobile user surrounded by a lot of noise and

talking on the phone. Location can also be used to trigger privacy related rules (e.g. prompt a warning, if a user is in a call with a customer in public spaces like airport lounges).

The IoT-enabled OpenMeetings Process Flow is illustrated in Figure 5. When a remote client (RC) connects to an online conference in OpenMeetings, the server creates a Real Time Messaging Protocol (RTMP) connection and it registers as a web-socket client endpoint. Semantic information related to the client and the IoT sources is retrieved, and semantic queries are automatically generated to monitor updates of sensory input from that client. The server also subscribes to these queries, and when sensory updates are detected, the *Stream Reasoning* component processes them by consuming events as they are produced and applies the rules to determine which action should be executed in the client application, returning results as a JSON object to the corresponding web socket clients. Based on these results, the web socket client calls a remote method of its remote client and changes the status of the relevant IoT icons accordingly, prompting a warning message if required.

4 Evaluation

We evaluated our proposed architecture mainly by measuring performance and scalability of OpenIoT and query processing within our framework. We believe this is key for the applicability of our approach, since it demonstrates that semantic technologies embedded in OpenIoT can be used in this practical system without hindering feasibility and user experience, and enabling enhanced IoT-intelligence capabilities and business logic to be deployed by leveraging semantic representations. In the current paper we focus on the system and the software tools. Next steps would aim at a full in-use application in an industry setting. As a result, we aim at a realistic set-up and study that will provide more in-depth evaluation of usability and user experience.

Performance and scalability are two critical aspects for applications which are designed to adapt and react to changes in near real-time. In this section, we present the results of our feasibility tests conducted to evaluate the performance and scalability of our proposed solution. With this evaluation we also aim at demonstrating how semantic technologies in IoT can be applied to real scenarios and create a new market for IoT-enabled solutions like in the collaboration and communication systems space, highlight what are the main drawbacks and limitations of state-of-the-art technologies such as X-GSN and OpenIoT in this setting, and provide some suggestions on what key aspects should be tackled by the research community to make the technology deployable on a larger scale.

Experimental Setup (Testbed). We deployed our OpenIoT Server over a machine running Debian GNU/Linux 6.0.10, with 8-cores of 2.13 GHz processor and 64 GB RAM. Apache OpenMeetings server is installed over a machine with Ubuntu 12.04.5 LTS, 1 core of 2.30GHz and 1 GB RAM, while Android App for sensor data transmission was installed over a mobile device running on Android

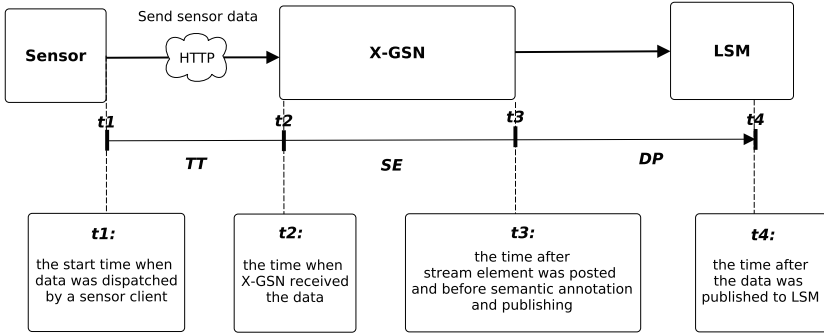


Fig. 6. Different Points for Processing Time Measurements

OS v4.3 (Jelly Bean), with Dual-core 1.5 GHz Krait and 1GB RAM, supporting Android SDK version 8 to the SDK version 21.

Performance. In order to evaluate the performance of our proposed solution, we observed the processing time required by various components of our infrastructure with varying size of the sensory stream. We identified various time points of observation to examine the average time delay for each time point of measurement. Figure 6, illustrates four time points of measurement, where:

- t_1 is the start time when data generated by a sensor is sent.
- t_2 is the time when X-GSN receives the data, hence we refer to $TT = t_2 - t_1$ as to the the time required by the network to send the data from the sensor to the server (transmission time).
- t_3 indicates the time when the raw data has been processed and stream elements have been created with time stamp allocation to each sensor observation, hence $SE = t_3 - t_2$ is the time needed to create a semantically annotated stream element.
- t_4 is the time when semantically annotated stream elements are successfully published to LSM, hence $DP = t_4 - t_3$ is the time required for publishing the semantically annotated triples into LSM.

Figure 7 depicts the average processing time required to perform the three main steps of the OpenIoT data processing pipeline, namely, (i) *Transmission Time (TT)*, (ii) *Stream Element Creation Time (SE)*, and (iii) *Data Publishing Time (DP)*. We specified an average delay of 0.5 seconds before sending each sensor observation to avoid overloading the system by concurrent requests (the impact of concurrent requests are investigated in the scalability analysis). All execution times shown in Figure 7 are the average of 5 executions. It is evident from the results that there is no significant delay in the *Stream Element Creation* and *Data Publishing* times, despite the increase in number of sensors from 10 to 10000. Similarly, it is no surprise to see the increase in the *Transmission Time* corresponding to an increase in the network traffic.

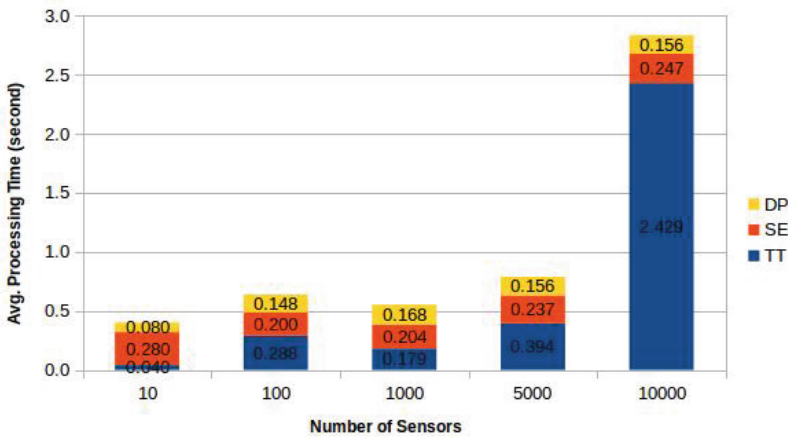


Fig. 7. Average Processing Time with Varying Number of Sensors

Scalability. We conducted our scalability test of the OpenIoT framework by sending concurrent requests with increasing number of users and observed the throughput (ability to deal with the concurrent users/requests per second) for each of the three phases of the OpenIoT data processing pipeline, namely (i) *Data Acquisition*, i.e. the ability of OpenIoT to receive data from sensors, (ii) *Stream Element Creation*, i.e. the ability of OpenIoT to process raw data and assign stream time stamps to each observation, and (iii) *Stream Data Publication*, i.e. the ability of OpenIoT to semantically annotate and publish/store the semantically annotated data within the LSM framework.

We used Apache JMeter ⁷ for conducting the scalability tests, which is a well known tool to perform stress tests over distributed web applications. We observed the throughput of OpenIoT with increasing number of concurrent requests (10,100,1000,5000 and 10000) sent by Apache JMeter with a ramp-up time of (5,50,500,2500 and 5000) accordingly. We allowed the execution time of 10 minutes after the completion of ramp-up time. As shown by our results in Figure 8, the throughput for the *Data Acquisition* phase remains higher than 200 requests/sec when the input size is 100 concurrent sensor requests or below, and it is reduced to around 76 requests/sec with concurrent requests sent from 10000 sensors. Similar throughput was achieved for *Stream Elements Creation* phase. However, the throughput of the *Stream Data Publication* phase, which is the ability of OpenIoT to publish the semantically annotated sensor data streams either to a streaming channel or storing within a data store, seems to be a bottleneck. Increase in the throughput of the *Stream Data Publication* phase from 1000 sensor inputs onwards, as shown in Figure 8, is a false positive. In fact, a deeper investigation into the results of Apache JMeter logs revealed that

⁷ <http://jmeter.apache.org/>

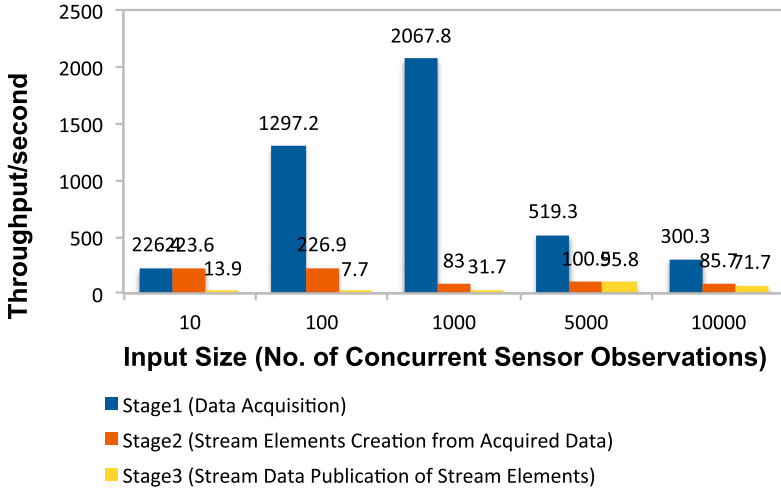


Fig. 8. Throughput of the Various Components of OpenIoT

this higher throughput was achieved because of significant increase in error rate caused by the fact that the LSM server starts refusing connection requests when the number of concurrent users increases beyond 1000. Further investigation is required in this respect to perform experiments where the noise generated by refused connections is filtered out.

5 Discussion and Future Deployment

In this paper, we showcase the applicability of semantic technologies in the IoT space for enterprise communication on the move. We focused on the advantages and feasibility of using the OpenIoT framework (extended with continuous query processing and IoT intelligence) in the Apache OpenMeetings collaboration and communication systems. We characterised requirements that can produce scalable solutions and issues to be investigated more carefully.

As discussed in our introduction and scenario description, semantic-based solutions for IoT in this space can facilitate the deployment of interoperable and flexible IoT-enabled enterprise applications. The ability to semantically integrate and query static and dynamic data makes it easier and more cost-effective to add new external sources and design the business logic (IoT-intelligence) promoting a new market of innovative services that provide significant advantages over ad-hoc IoT deployment. Semantics also helps integrating semantic knowledge about a user independently of the applications producing it (e.g. mobile app for producing sensory input, desktop client for online meeting services, calendar for meeting schedule, etc.) as long as there is a semantic information model that relates the different pieces of knowledge.

Performance. Results are very positive regarding the use of semantics since there is very little and linear impact of semantic-related processing in our IoT-enabled OpenMeetings. The real bottleneck seems to be network traffic, which suggests increasing bandwidth or clever management of queues in order to improve transmission delays.

Scalability. Test results suggests that the actual acquisition and generation of semantic streams can manage up to 200 readings per second if the total concurrent requests by users is not much greater than 100. This could be reasonable in a medium enterprise by constraining the number of concurrent meetings (or participants) that can be scheduled on the OpenMeetings server. If we want the processing pipeline to go as far as the IoT-intelligence goes, we can deal with a much lower throughput of a few (concurrent) sensory input per second, due to the time required to publish the acquired annotated sensor data to the stream processing and reasoning layer.

The X-GSN to LSM communication is performed per-observation by default, this is quite slow and can be very costly when there are a lot of concurrent X-GSN threads (e.g. concurrent sensory input) to be handled. It is worth mentioning that servlet's threads on the server side have been managed without using any optimisation queue, therefore there is easy margin for improvement. Hence, in order to reduce the impact of the bottleneck to publish annotated sensor streams to an application channel via LSM, a possible solution is to either use a cluster version of JBoss hosting X-GSN server, or to configure a queue in the default implementation of X-GSN that makes it possible to buffer the observations.

Deployment Plan. While acting on the application side is entirely dependent on the type of application, acting on the X-GSN implementation is related to improving current semantic solutions and we have already triggered the discussion to list it as a potential improvement within the OpenIoT developers community. Regarding feasibility for continuous query evaluation and reasoning, we are currently evaluating an initial testbed by mocking up 25 simultaneous meetings using our IoT-MMS. Each meeting consists of 10 attendees including organiser, while 4 type of sensor observations (noise level, proximity, location and light) were monitored for all mobile users. Initial results show that our IoT-MMS is capable of generating simultaneous queries over the proposed test without any substantial performance issues, with similar results as the ones published for the specific stream processing engine evaluation [13]. Following this simulation, we will be setting up a deployment for in-use evaluation of our IoT-MMS system within our partner industry in the next few months, following integration of our solution in a proprietary online collaboration system. This will make it possible to evaluate usability and performances of specific business logic related to online meeting management events and action triggers, and conduct a proper in-use evaluation not only with respect to scalability and performance but also usability and impact.

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proc. of VLDB, pp. 1199–1202. VLDB Endowment (2006)
2. Ait, N.K., Al, J.E.B., Al, A.G.: D2. 3 openiot detailed architecture and proof-of-concept specifications (2013)
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. *Computer networks* **54**(15), 2787–2805 (2010)
4. Barnaghi, P., Wang, W., Henson, C., Taylor, K.: Semantics for the internet of things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS)* **8**(1), 1–21 (2012)
5. Calbimonte, J.P., Sarni, S., Eberle, J., Aberer, K.: Xgsn: an open-source semantic sensing middleware for the web of things. In: 7th International SSN Workshop (2014)
6. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The {SSN} ontology of the {W3C} semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web* **17**, 25–32 (2012)
7. Groza, T., Handschuh, S., Moeller, K.: The nepomuk project-on the way to the social semantic desktop (2007)
8. 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)
9. Le-Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. *Web Semantics: Science, Services and Agents on the World Wide Web* **16**, 42–51 (2012)
10. Mehmood, Q., Ali, M.I., Fagan, O., Friel, O., Mileo, A.: Semantically inter-linked notification system for ubiquitous presence management. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) ODBASE 2013. LNCS, vol. 8185, pp. 588–605. Springer, Heidelberg (2013)
11. Mileo, A., Abdelrahman, A., Policarpio, S., Hauswirth, M.: StreamRule: a nonmonotonic stream reasoning system for the semantic web. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 247–252. Springer, Heidelberg (2013)
12. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: Vision, applications and research challenges. *Ad Hoc Networks* **10**(7), 1497–1516 (2012)
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. Sauermann, L., Grimnes, G.A.A., Kiesel, M., Fluit, C., Maus, H., Heim, D., Nadeem, D., Horak, B., Dengel, A.R.: Semantic desktop 2.0: the gnowsis experience. 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. 887–900. Springer, Heidelberg (2006)
15. Scerri, S., Rivera, I., Debattista, J., Thiel, S., Cortis, K., Attard, J., Knecht, C., Schuller, A., Hermann, F.: di.me: Ontologies for a pervasive information system. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 499–504. Springer, Heidelberg (2014)
16. Wang, W., De, S., Toenjes, R., Reetz, E., Moessner, K.: A comprehensive ontology for knowledge representation in the internet of things. In: IEEE TrustCom, pp. 1793–1798. IEEE (2012)

SPARQL and Querying Linked Data

LSQ: The Linked SPARQL Queries Dataset

Muhammad Saleem¹(✉), Muhammad Intizar Ali², Aidan Hogan³,
Qaiser Mehmood², and Axel-Cyrille Ngonga Ngomo¹

¹ Universität Leipzig, IFI/AKSW, PO 100920, 04009 Leipzig, Germany
saleem@informatik.uni-leipzig.de

² Insight Center for Data Analytics, National University of Ireland, Galway, Ireland

³ Department of Computer Science, Universidad de Chile, Santiago, Chile

Abstract. We present LSQ: a Linked Dataset describing SPARQL queries extracted from the logs of public SPARQL endpoints. We argue that LSQ has a variety of uses for the SPARQL research community, be it for example to generate custom benchmarks or conduct analyses of SPARQL adoption. We introduce the LSQ data model used to describe SPARQL query executions as RDF. We then provide details on the four SPARQL endpoint logs that we have RDFised thus far. The resulting dataset contains 73 million triples describing 5.7 million query executions.

1 Introduction

Although there are now hundreds of public SPARQL endpoints available on the Web – collectively exposing billions of facts and receiving millions of queries per month – current works suggest that in terms of SPARQL technology, there is still considerable room for improvement [1, 2, 9]. Many of these endpoints suffer from availability and performance issues [2]. In addition, the recent recommendation of SPARQL 1.1 [6] brings new challenges. Tackling these challenges could benefit from more data about how users are currently interacting with SPARQL endpoints and which queries they are sending. Such knowledge may help to focus research on optimising those queries or query features that are most often used.

Although query logs are available for public SPARQL endpoints through initiatives like USEWOD [4], the datasets are only accessible after having signed legal agreements, which limits re-use. Likewise, the format of the raw logs is ad-hoc in nature, depending on their source. We thus introduce the Linked SPARQL Queries Dataset (LSQ): a public, openly accessible dataset of SPARQL queries extracted from endpoint logs.¹ The current version that we describe in this paper consists of 73.2 million triples collected from four query logs, which we have gathered from the maintainers of public endpoints and for which we have gotten permission to make the logs public. We foresee a number of potential use cases for such a dataset:

UC1 Custom Benchmarks. The LSQ dataset can be used to generate realistic benchmarks by selecting queries matching ad-hoc desiderata [12].

¹ The LSQ dataset is available from <http://aksw.github.io/LSQ/>.

UC2 SPARQL Adoption. The data can be used by researchers to conduct analyses of features used in real-world SPARQL queries [3,10,11].

UC3 Caching. Works on caching [8,14] could benefit from a dataset of real-world queries by, e.g., analysing real-world sequences of queries.

UC4 Usability. Analysis of user behaviour – e.g., errors made, how they refine queries, etc. – could guide the design of better interfaces.

UC5 Meta-Querying. One could find out what are the queries that people are asking about a resource of interest, be it a product, person, city, etc.

These use cases not only require details about queries, but also query executions, agents, result sizes, etc. We now describe the LSQ data model, whose goal is to comprehensively capture all such aspects of query logs.

2 RDF Data Model

Our goal is to create a Linked Dataset describing the SPARQL queries issued to various public SPARQL endpoints. In Figure 1, we provide an overview of the core of the schema for the LSQ data-model. Listing 1 provides a comprehensive example output for a query. The main aspects of the dataset are now presented.²

Queries in the data are typed as a subclass of `sq:Query` (e.g., `lsqv:Select`, `lsqv:Ask`, etc.). We create query instances for each log whereby a query is linked to a single endpoint from whose log it was extracted. Hence, if the same query with the same syntax is issued to the same endpoint multiple times, it is represented with a single instance of `sq:Query`, linked to multiple instances of `lsqv:Execution` for each time the query was run. Each such execution instance provides a time (`dct:issued`) and a unique agent IRI computed from a cryptographically-hashed and salted I.P. address (`lsqv:agent`).

To help make the dataset as general as possible, we attach a complete SPIN representation of the query to each query instance [7]. Given that the SPIN representation may involve an arbitrary level of nesting using a variety of predicates, to make querying LSQ more convenient and efficient, we provide shortcut triples to indicate the SPARQL query features used in the query. These triples link query instances (with the predicate `lsqv:usesFeature`) to instances of `sd:Feature`. We enumerate a comprehensive list of such feature instances in our vocabulary, including `lsqv:Filter`, `lsqv:Optional`, `lsqv:SubQuery`, etc. We also provide shortcuts to the IRIs and literals mentioned in a query so consumers can easily find all queries about a given resource.

In addition to the query structure, we also provide generic *structural statistics* [1] about the static query including the number of Basic Graph Patterns (`lsqv:bgps`) and the number of triple patterns (`lsqv:triplePatterns`). We also provide *data-driven statistics* [1] (incl. the number of results returned and the query runtime) about the execution of the query. Since such data are not typically provided by the logs, we generate these statistics by running the query

² More details are available in the technical report at <http://goo.gl/LZehl1>.

Listing 1. An example LSQ representation of an SWDF query

```

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix lsqr: <http://lsq.aksw.org/res/> .
@prefix lsqrd: <http://lsq.aksw.org/res/SWDF-> .
@prefix lsqv: <http://lsq.aksw.org/vocab#> .
@prefix sp: <http://spinrdf.org/sp#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# QUERY INSTANCE META-DATA
lsqrd:q483 lsqv:endpoint <http://data.semanticweb.org/sparql> ;
  sp:text ""SELECT DISTINCT ?prop
  WHERE {
    ?obj rdf:type swdf:SessionEvent .
    ?obj ?prop ?targetObj .
    FILTER (isLiteral(?targetObj)) }
  LIMIT 150"" .

# STRUCTURAL META-DATA
lsqrd:q483 lsqv:bgps 1 ; lsqv:triplePatterns 2 ; lsqv:joinVertices 1 ;
  lsqv:meanJoinVerticesDegree 2.0 ;
  lsqv:usesFeature lsqv:Filter , lsqv:Distinct , lsqv:Limit ;
  lsqv:mentionsSubject "?obj" ;
  lsqv:mentionsPredicate "?prop" , rdf:type ;
  lsqv:mentionsObject "?targetObj" , swdf:SessionEvent ;
  lsqv:joinVertex lsqr:q483-obj .
lsqr:q483-obj lsqv:joinVertexDegree 2 ; rdf:type lsqv:Star .

# DATA-SENSITIVE META-DATA
lsqrd:q483 lsqv:resultSize 16 ; lsqv:runTimeMs 6 ;
  lsqv:meanTriplePatternSelectivity 0.5007155695730322 ;

# QUERY EXECUTION META-DATA
lsqrd:q483 lsqv:execution lsqrd:q483-e1 , lsqrd:q483-e2 , lsqrd:q483-e3 , lsqrd:q483-e4 .
lsqrd:q483-e1 lsqv:agent lsqr:A-WlxKEOQQRlhCUBdGRx1QGVRbQRNsN2YUWF5W ;
  dct:issued "2014-05-22T17:08:17+01:00"^^xsd:dateTimeStamp .
lsqrd:q483-e2 lsqv:agent lsqr:A-WlxKEOQQRlhCUBdGRx1QGVRdRBNsN2YUW1pS ;
  dct:issued "2014-05-20T14:34:35+01:00"^^xsd:dateTimeStamp .
lsqrd:q483-e3 lsqv:agent lsqr:A-WlxKEOQQRlhCUBdGRx1QGVRdRBNsN2YUW1pS ;
  dct:issued "2014-05-20T14:28:37+01:00"^^xsd:dateTimeStamp .
lsqrd:q483-e4 lsqv:agent lsqr:A-WlxKEOQQRlhCUBdGRx1QGVRdRBNsN2YUW1pS ;
  dct:issued "2014-05-20T14:24:13+01:00"^^xsd:dateTimeStamp .

# SPIN REPRESENTATION
lsqrd:q483 a sp:Select ;
  sp:distinct true ; sp:limit "150"^^xsd:long ;
  sp:resultVariables ( [ sp:varName "prop"^^xsd:string ] ) ;
  sp:where (
    [ sp:subject [ sp:varName "obj"^^xsd:string ] ;
      sp:predicate rdf:type ;
      sp:object <http://data.semanticweb.org/ns/swc/ontology#SessionEvent>
    ]
    [ sp:subject [ sp:varName "obj"^^xsd:string ] ;
      sp:predicate [ sp:varName "prop"^^xsd:string ] ;
      sp:object [ sp:varName "targetObj"^^xsd:string ]
    ]
  )
  [ a sp:Filter ;
    sp:expression [ a sp:isLiteral ; sp:arg1 [ sp:varName "targetObj"^^xsd:string ] ]
  ]
) .

```

Table 1. High-level analysis of the queries and query executions in the LSQ dataset for each log (QE = Query Executions, UQ = Unique Queries, PE = Parse Errors, RE = Runtime Error, ZR = Zero Results, SEL = SELECT, CON = CONSTRUCT, DES = DESCRIBE; percentages are with respect to UQ)

| DATASET | QE № | UQ № | PE № | RE № | ZR № | SEL % | CON % | DES % | ASK % |
|---------|-----------|-----------|---------|---------|---------|----------|----------|----------|----------|
| DBpedia | 1,728,041 | 1,208,789 | 426,425 | 69,523 | 176,257 | 94.6 | 0.9 | 0.1 | 4.4 |
| LGD | 1,656,254 | 311,126 | 13,546 | 50,059 | 143,574 | 89.3 | 2.3 | 0.0 | 8.4 |
| SWDF | 1,411,483 | 99,165 | 13,645 | 475 | 25,674 | 68.8 | 0.0 | 31.1 | 0.1 |
| BM | 879,426 | 129,989 | 100,916 | 0 | 29,073 | 100 | 0.0 | 0.0 | 0.0 |
| Overall | 5,675,204 | 1,749,069 | 554,532 | 120,057 | 374,578 | 91.6 | 1.2 | 2.3 | 4.9 |

Table 2. Percentage of unique queries containing different types of joins (a query may contain multiple join types).

| DATASET | STAR % | PATH % | HYBRID % | SINK % | NO JOIN % |
|---------|-----------|-----------|-------------|-----------|--------------|
| DBpedia | 38.58 | 8.60 | 6.79 | 6.31 | 61.23 |
| LGD | 28.18 | 9.46 | 7.57 | 1.24 | 72.00 |
| SWDF | 10.70 | 11.25 | 4.01 | 0.93 | 84.25 |
| BM | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 |
| Overall | 33.05 | 8.79 | 6.62 | 4.51 | 66.51 |

log contains a noticeably high ratio of parse errors (77.63%), compared with DBpedia (35.27%), SWDF (13.75%), or LGD (4.35%).³ Conversely, while LGD is the lowest in terms of parse errors, it generates the highest ratio of runtime errors (16.08%), followed by DBpedia (5.54%), SWDF (0.05%), and BM (0%). Often these are timeouts, which will, in practice, occur more frequently for larger datasets.

Table 2 shows the popularity of join types as defined previously in [13]. The idea is to count individual join variables within a BGP as individual joins and type them depending on how they connect triple patterns. We say that a join vertex has an “outgoing link” if it appears as a subject of a triple pattern, and that it has an “incoming link” if it appears as predicate or object. STAR has multiple outgoing links but no incoming links. PATH has precisely one incoming and one outgoing link. HYBRID has at least one incoming and outgoing link and three or more links. SINK has multiple incoming links but no outgoing links. From Table 2, we see that most queries are STAR (33.1%) or contain no join (66.5%); again we see the uniformity of BM queries suggesting the influence of one agent.

³ We suspect that for BM, one automated agent is asking a high volume of simple potentially “invalid” queries to the endpoint; unfortunately the BM log did not include agent data, so we can neither confirm nor refute this possibility.

Table 3. Comparison of the mean values of different query features across all query logs (RS = Result Size, TPs = Triple Patterns, JVs = Join Vertices, MJVD = Mean Join Vertex Degree, MTPS = Mean Triple Pattern Selectivity)

| DATASET | RS | BGPs | TPs | JVs | MJVD | MTPS | RUNTIME (ms) |
|---------|--------|------|------|------|------|-------|--------------|
| DBpedia | 87.57 | 1.81 | 2.22 | 0.40 | 0.78 | 0.002 | 20.26 |
| LGD | 161.90 | 1.75 | 2.16 | 0.37 | 0.75 | 0.030 | 32.28 |
| SWDF | 19.65 | 2.57 | 2.94 | 0.26 | 0.35 | 0.025 | 11.98 |
| BM | 0.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.000 | 6.78 |
| Overall | 122.45 | 1.74 | 2.04 | 0.24 | 0.45 | 0.013 | 26.40 |

Table 4. Percentage of queries using various specific SPARQL features

| DATASET | UNION | OPTIONAL | DISTINCT | FILTER | REGEX | SERVICE | SUB-QUERY |
|---------|-------|----------|----------|--------|-------|---------|-----------|
| DBpedia | 4.42 | 36.20 | 18.44 | 23.47 | 2.90 | 0.0005 | 0.00 |
| LGD | 9.65 | 25.10 | 22.25 | 31.10 | 1.25 | 0.0000 | 0.01 |
| SWDF | 32.71 | 25.32 | 45.40 | 0.95 | 0.06 | 0.0012 | 0.02 |
| BM | 0.00 | 0.00 | 100.00 | 0.00 | 0.00 | 0.0000 | 0.00 |
| Overall | 7.64 | 31.78 | 23.30 | 23.19 | 2.22 | 0.0004 | 0.01 |

Table 3 shows the mean values for various query features across all query logs. These features are often considered, e.g., when designing SPARQL benchmarks [1, 5]. The SWDF queries are generally more complex, on average, in terms of the number of BGPs and total number of triple patterns. However, they contain fewer joins among triple patterns and the join vertex degree is also quite low (e.g., 0.35 for SWDF vs. 0.78 for DBpedia). We also see that slower runtimes correspond with larger dataset sizes. We again see that the BM queries often return zero results, suggesting again a high volume of simple, synthetic queries.

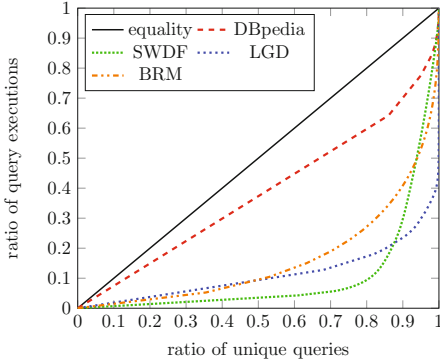
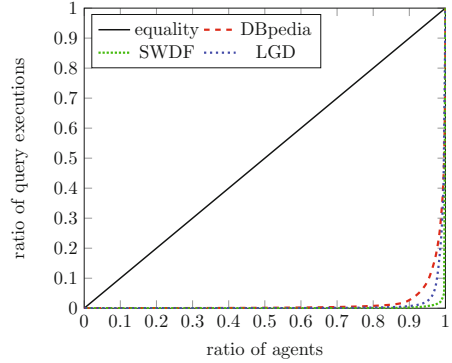
Tables 4 and 5 show the percentage use of (groups of) different SPARQL features [3]; a query is counted in a group if it uses one such feature. We found that the SPARQL 1.1 features are rarely used; however, in the case of DBpedia and LGD, this may be due to the age of the logs. The most widely used feature is `OPTIONAL` (31.78%), followed by `DISTINCT` (23.3%) and `FILTER` (23.19%). Solution modifiers (i.e., `LIMIT`, `OFFSET`, `ORDER BY`) are also quite often used (18.11%).

Execution and Agent Analysis: Thus far we have analysed unique queries. We now look at (a) whether the same queries tend to be executed many times and (b) how many agents are responsible for how many executions.

With respect to the number of times a given query is executed, if we take the total number of query executions (5,675,204) and the total number of unique queries (1,749,069) from Table 1, we can see that a given (syntactically identical) query is executed on average about 3.2 times in the scope of the logs defined.

Table 5. Percentage of queries using various classes of features

| DATASET | SOLUTION MOD. | AGGREGATES | (-)EXISTS | BINDING | GRAPH |
|---------|---------------|------------|-----------|---------|-------|
| DBpedia | 1.036 | 0.001 | 0.001 | 0.000 | 0.002 |
| LGD | 60.443 | 0.007 | 0.000 | 0.000 | 0.000 |
| SWDF | 33.265 | 2.405 | 0.001 | 0.008 | 0.001 |
| BM | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Overall | 18.117 | 0.174 | 0.001 | 0.001 | 0.001 |


Fig. 2. Lorenz curve for distribution of executions per unique query

Fig. 3. Lorenz curve for distribution of executions per unique agent

To compare this distribution for the four logs, Figure 2 provides a Lorenz curve, which shows what (maximal) ratio of unique queries account for what (minimal) ratio of query executions. For example, we see that for SWDF, 80% of the unique queries account for about 10% of the overall executions, or equivalently that the top 20% most frequently executed queries account for 90% of all executions. On the other hand, the executions for DBpedia are much more evenly spread. For LGD, the sharp ascent of the curve suggests that a handful of unique queries are run a great many times and form the overall majority of executions.

Regarding unique agents, DBpedia had 3,041, LGD had 725 and SWDF had 274; we did not have agent data for BM. Figure 3 presents the Lorenz curve of how executions are distributed amongst agents, in which we can see a heavy skew; for example, 90% of the agents with fewest executions are cumulatively responsible for fewer than 3% of the total executions (2.7% for DBpedia, 0.7% for LGD, and 0.2% for SWDF). From this curve, we posit that most queries encountered in these logs are from a few high-volume, automated agents; this should potentially be taken into account by users of the LSQ dataset (again, our goal is to provide the queries from real-world logs “as is”).

4 Conclusions and Future Directions

In this paper we presented LSQ, which is (to the best of our knowledge) the first public Linked Dataset describing SPARQL queries issued to endpoints. We introduced various use cases for LSQ, detailed our data model, and analysed the results of RDFising logs from four endpoints. The current version of LSQ contains 73 million triples describing 5.7 million query executions.

We are currently collecting logs from other SPARQL endpoints (e.g., Bioportal, Strabon) that will be added into LSQ. We likewise hope to update and extend logs from current endpoints (esp. DBpedia). We will also link the dataset with the benchmark generation framework FEASIBLE to ease the development of benchmarks customised towards specific software applications or algorithms. The Linked Dataset, a SPARQL endpoint, and complete dumps are all available on the LSQ homepage – <http://aksw.github.io/LSQ/> – along with pointers to code, a VoID description, example LSQ queries, and various other dataset assets.


Acknowledgments. This work was supported in part by a research grant from the German Ministry for Finances and Energy under the SAKE project (Grant No. 01MD15006E), by Science Foundation Ireland (SFI) under Grant No. SFI/12/RC/2289, by the Millennium Nucleus Center for Semantic Web Research under Grant No. NC120004 and by Fondecyt Grant No. 11140900.

References

1. Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 197–212. Springer, Heidelberg (2014)
2. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: SPARQL web-querying infrastructure: ready for action? In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 277–293. Springer, Heidelberg (2013)
3. Arias, M., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An empirical study of real-world SPARQL queries. CoRR (2011)
4. Berendt, B., Hollink, L., Hollink, V., Luczak-Rösch, M., Möller, K., Vallet, D.: Usage analysis and the web of data. SIGIR Forum **45**(1), 63–69 (2011)
5. Görlitz, O., Thimm, M., Staab, S.: SPLODGE: systematic generation of SPARQL benchmark queries for linked open 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 I. LNCS, vol. 7649, pp. 116–132. Springer, Heidelberg (2012)
6. Harris, S., Seaborne, A., Prud’hommeaux, E. (eds.): SPARQL 1.1 Query Language. W3C Recommendation, March 21, 2013
7. Knublauch, H., Hendler, J.A., Idehen, K. (eds.): SPIN - Overview and Motivation. W3C Member Submission, February 22, 2011

8. Lampo, T., Vidal, M.-E., Danilow, J., Ruckhaus, E.: To cache or not to cache: the effects of warming cache in complex SPARQL queries. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part II. LNCS, vol. 7045, pp. 716–733. Springer, Heidelberg (2011)
9. 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)
10. Picalausa, F., Vansummeren, S.: What are real SPARQL queries like?. In: SWIM (2011)
11. Rietveld, L., Hoekstra, R.: Man vs. machine: differences in SPARQL queries. In: USEWOD (2014)
12. Saleem, M., Mehmood, Q., Ngomo, A.-C.N.: FEASIBLE: a featured-based SPARQL benchmark generation framework. In: ISWC (2015) (to appear)
13. Saleem, M., Ngonga Ngomo, A.-C.: HiBISCuS: hypergraph-based source selection for SPARQL endpoint federation. In: Presutti, V., d’Amato, C., Gandon, F., d’Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 176–191. Springer, Heidelberg (2014)
14. Williams, G.T., Weaver, J.: Enabling fine-grained HTTP caching of SPARQL query results. 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. 762–777. Springer, Heidelberg (2011)

Automatic Curation of Clinical Trials Data in LinkedCT

Okkie Hassanzadeh^{1,2} and Renée J. Miller¹

¹ Department of Computer Science, University of Toronto, Toronto, Canada
{okkie,miller}@cs.toronto.edu

² IBM T.J. Watson Research Center, New York, USA

Abstract. The Linked Clinical Trials (LinkedCT) project started back in 2008 with the goal of providing a Linked Data source of clinical trials. The source of the data is from the XML data published on ClinicalTrials.gov, which is an international registry of clinical studies. Since the initial release, the LinkedCT project has gone through some major changes to both improve the quality of the data and its freshness. The result is a high-quality Linked Data source of clinical studies that is updated daily, currently containing over 195,000 trials, 4.6 million entities, and 42 million triples. In this paper, we present a detailed description of the system along with a brief outline of technical challenges involved in curating the raw XML data into high-quality Linked Data. We also present usage statistics and a number of interesting use cases developed by external parties. We share the lessons learned in the design and implementation of the current system, along with an outline of our future plans for the project which include making the system open-source and making the data free for commercial use.

Keywords: Clinical trials · Linked data · Data curation

1 Introduction

The clinical research community and the healthcare industry have well recognized the need for timely and accurate publication of data related to all aspects of clinical studies, ranging from recruitment information and eligibility criteria to details of different phases and the achieved results [8, 15, 17, 19, 20]. ClinicalTrials.gov is currently the main mechanism of achieving this goal. Maintained by U.S. National Institutes of Health, it is the largest and most widely used registry of clinical studies with registered trials from almost every country in the

The data source is publicly available at <http://linkedct.org>. Data dumps available at <http://purl.org/net/linkedct/datadump>. Please note scheduled maintenance down times on our Twitter feed <https://twitter.com/linkedct>. Resource URIs validated as proper Linked Data by <http://validator.linkeddata.org/> (“All tests passed”). Part of LOD cloud. Registered on <http://datahub.io>.

R.J. Miller—Partially supported by NSERC BIN.

world. There has been a significant increase in the number of registered trials as a result of a mandate by FDA and requirement from various journals that a trial needs to be registered before it can start or the results can be published [24, 25]. There is also an ongoing effort in the community to increase the quality of the data, and require publication of the results after the completion of the registered trials.

The Linked Clinical Trials (LinkedCT) project started in 2008 with the goal of publishing the ClinicalTrials.gov data as high-quality (5-star [4]) Linked Data on the Web. Our inspiration for the project came from a simple experiment on matching patients with clinical trials as a part of the LinQuer project [10]. A simple task of retrieving all the trials on a certain condition along with all their attributes turned into a laborious Extract-Transform-Load process. The process involved studying the schema of the XML data, writing code to crawl the data and load them in IBM DB2 on a local server to be able to query the data using DB2's pureXML features. Our initial goal was to simply publish the result of this transformation as Linked Data using D2R server [5], with the main challenge being discovering links to external sources [11]. Initial user feedback and work done as part of the LODD project [13, 14] on developing use cases over the data made it apparent that the transformation process was not only laborious, but also error-prone. The errors along with the slow and static transformation process called for a new solution that replaces the manually designed transformation process with a mostly automated *curation* [7, 23] of the clinical trials data.

The following section describes some of the challenges faced in designing an automated data curation process and our solution using xCurator [23]. We then describe a number of interesting applications and usage scenarios of LinkedCT. We finish the paper with a number of future directions which includes making the data available free for commercial use, and making the platform open-source to facilitate development of applications hosted on LinkedCT.org.

2 Data Curation in LinkedCT

In this section, we describe the end-to-end curation process we have designed to construct an up-to-date high-quality Linked Data source out of the XML data published by ClinicalTrials.gov. Figure 1 shows the overall architecture of the system. In what follows, we describe different components of the system.

2.1 From XML to RDF Knowledge Graph

Although there are mapping tools and systems for transforming XML into RDF (e.g., XSPARQL [2]), a key challenge in building a high-quality linked knowledge base is construction of a target data model that accurately describes the entity types and their relationships that exist in the original data, and that also facilitates knowledge discovery and linkage to external sources. As stated earlier, our initial manually-designed transformation of the data into relational and RDF resulted in a number of quality issues reported by users and discovered through working on usage scenarios as a part of the LODD project [13, 14]. For example:

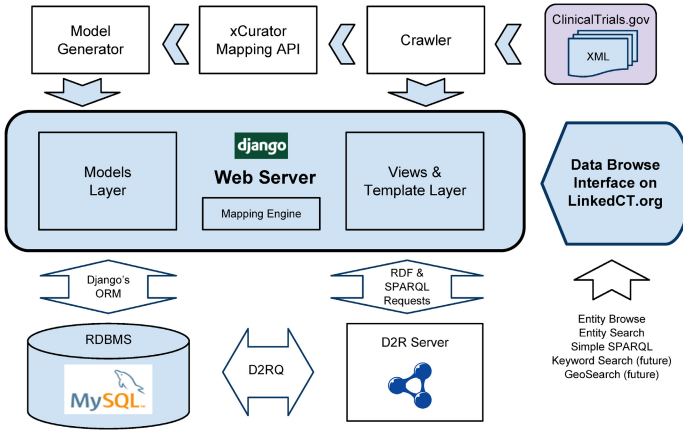


Fig. 1. LinkedCT Platform Architecture

- Users familiar with the original NIH data pointed out missing information in the form of either missing entity types (RDF classes) or missing attributes (RDF properties) from entities of a certain type. Such data can easily be missed in a manual mapping process.
- Use cases required a literal property (e.g., location country represented as a string-valued property) to be represented as an entity (e.g., Country being an RDF class). For example, linking data is typically done only over entities (with URIs that can be linked).
- Users found inconsistencies with the original XML, and we were unable to verify the reason (programming error vs. an update in the source) due to lack of provenance information or caching of the original NIH data.

Figure 2 shows a sample XML tree from the data that can help explain the reason behind some of the problems in a manual mapping design.

- Nodes with label `mesh_term` contain string values. Only a careful study by an expert can reveal they describe entities of type Drug.
- A simple approach of making a type (class) per each non-leaf node in the tree (which is similar to the common RDB2RDF approach of creating a class per each table) will result in entities of type `id_info` whereas this node is simply grouping a list of identifiers and is not representing an entity. Such extraneous types can make the data hard to query and understand.
- There are nodes in the tree such as `collaborator` and `lead_sponsor` that have different labels but represent the same type of entity, i.e., an agency.

The above challenges are addressed in xCurator [23], an end-to-end system for transformation of semi-structured data into a linked knowledge base. Our experience in LinkedCT has been one of the main use cases for evaluation of the accuracy of the mapping discovery in xCurator. The xCurator mapping generator uses a

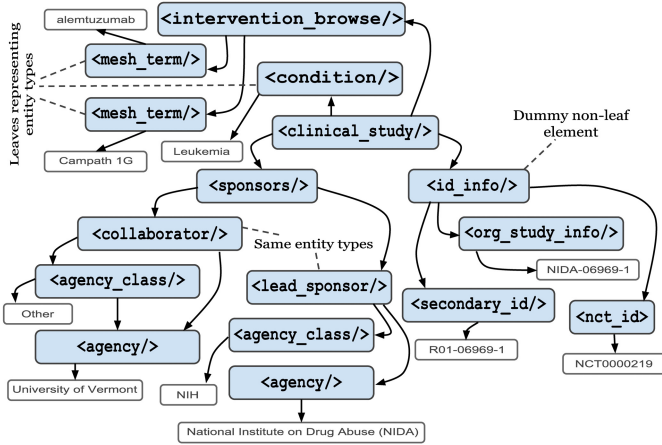


Fig. 2. Sample XML from ClinicalTrials.gov

number of heuristics based on statistical measures and other data properties over a large enough sample of instances to automatically construct a set of classes that refer to real-world entity types. A main criteria for identification of entity types in xCurator is the ability to link instances of the derived type with entities in external knowledge bases, which will in turn result in a higher quality linked data sources in terms of linkage to external sources. We refer the reader to Hassas et al. [23] for a detailed description of the mapping discovery and evaluation using LinkedCT data. The results clearly show the superiority of xCurator’s automatic mapping discovery to the initial manual mapping, even if only a small random subset of the data is used to generate the mappings.

2.2 Web Application Design

Although xCurator provides an end-to-end solution for mapping discovery and creation, along with publication of the resulting knowledge base in RDF following the Linked Data principles, we chose to use only the mapping generator component due to a number of reasons. First and foremost, xCurator is designed as a generic tool that can be used for any semistructured data whereas the strict focus on clinical trials data can help us better tune the system and algorithms to improve the outcome. Moreover, we have been able to tune LinkedCT’s implementation to make a relatively light-weight web application that unlike xCurator can run on modest hardware or virtual machines. As shown in Figure 1, the web application is extended with a *Crawler* module that continuously checks for new trials on ClinicalTrials.gov and also checks for updates in existing trials. The xCurator mapping generator component uses the crawled data in a one-time process to generate mappings. These mappings are translated into an intermediate Object-Relational Mapper (ORM) model definition used by the web application

which is implemented in the Django framework [1]. This is done in the *Model Generator* module which is a Python code generator that can directly be used in the web application.

In addition to the ORM models, the Django web application handles HTTP service requests with a set of templates that provide the HTML and RDF view for the data browse web interface accessible at <http://linkedct.org>. In addition, it provides various APIs called by the crawler for addition and update process. Linkage to external sources such as DrugBank, PubMed, GeoNames, and DBpedia are performed using pre-defined linkage rules embedded in the *mapping engine* module in the Django web application and called during addition and update procedures. The mapping engine also performs duplicate detection in a similar way using pre-defined rules. The rules are defined using the results of our previous study on the quality of various linkage techniques [10, 11].

2.3 Data Backend and SPARQL Endpoint

Ideally, the web application can work on top of a reliable RDF store for storage and querying. Unfortunately, there are no active projects on RDF support over Django Web Framework, and no non-commercial RDF stores capable of handling the very large number of updates and queries that LinkedCT needs. The alternative option is using a relational backend. We are currently using a MySQL database hosted on a secondary server. For an RDF view and SPARQL endpoint, we use the D2R server with D2RQ mappings [5] that are similarly generated automatically out of xCurator mappings by the model generator module. For scalability, we have to put a limit in D2R server configuration that limits the number of results returned and so the SPARQL endpoint is only useful for basic querying with small result sets. This makes it feasible to keep the web application lightweight despite the large load and large amount of data. Clearly, the limit on the SPARQL endpoint is far from ideal and one of the main shortcomings of our framework that we wish to address in the future as pointed out in Section 4. For applications requiring full SPARQL support, we make NTriples data dumps available regularly (once a month) and on demand.

3 Applications and Usage Statistics

Although ClinicalTrials.gov provides a relatively powerful “advanced search” feature, there is still a clear benefit in using LinkedCT even for basic data discovery and semantic search over the data. For example, using simple SPARQL queries or even on the Linked Data HTML browse interface on <http://linkedct.org>, one can quickly find a field named `is_fda_regulated` for entities of type Trial.¹ The ClinicalTrials.gov web pages and its advanced search do not include this field, and a keyword search for this field name yields no answer (likely because their

¹ See: http://data.linkedct.org/resource/trial/fields/is_fda_regulated/ - at the time of this writing, there are 58,122 trials with `is_fda_regulated` set to `Yes` and 110,889 trials set to `No`.

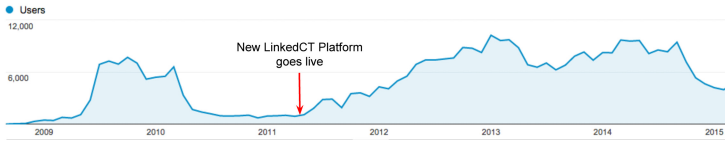


Fig. 3. Number of Unique LinkedCT.org Website Users

search is only over data, not metadata). Another benefit of publishing the data as Linked Data and providing a SPARQL endpoint is facilitating application building. One such application is the mobile faceted browsing application developed by Sonntag et al. [21] that uses LinkedCT and other interlinked sources to assist clinicians with various patient management activities.

Another basic advantage of publishing high-quality Linked Data is an implicit and important yet undervalued effect on the visibility of the data and search engine rankings. Figure 3 shows the number of unique visitors to LinkedCT.org website since the start of the project. The initial website receives a large number of visitors after its initial announcement and being indexed by Web search engines, but then the number goes down to under 1,000 by May 2011 when the new platform goes live. Our analysis of the access logs show that the main decrease is from search engine referrals. This completely changes after the new platform described in Section 2 goes live, which happens quietly without any public announcements of the new platform. Again, our analysis shows that a large portion of the increase is the result of search engine referrals, but this time the number remains high. This can be attributed to both the dynamic update of the trials in the new platform, and the higher quality and quantity of links to external sources. Again, achieving this without any effort on search engine optimization or public announcements on the project shows an interesting side outcome of following Linked Data principles.

Apart from the above-mentioned applications and basic advantages of publishing Linked Data, there are several very interesting healthcare applications that rely on LinkedCT as one of their primary sources. Examples include:

- Zaveri et al. [26] perform a very interesting study on research-disease disparity. The study shows that there is a large gap between the amount of resources spent on a disease (in terms of clinical trials and publications) and the disease fatality (death rate). LinkedCT is one of the three core data sets used in this study, a study which required links to external sources.
- The Linked Structured Product Labels (LinkedSPLs) project aims at publishing FDA’s drug label information as Linked Data on the Web [6]. A main use case in the project is discovering missing Adverse Drug Reactions (ADRs) through linkage to finished trials on LinkedCT and their associated PubMed articles.
- PANg project [3] that aims at discovering patterns in knowledge graphs, uses LinkedCT data to find clusters of strongly related studies, drugs, and diseases. An example of an interesting pattern is one that shows the drug

“Varenicline”, a drug used to treat nicotine addiction, has recently been linked to treating alcohol use disorders. At the time of this writing, this information is absent from the Wikipedia article on Varenicline.

4 Conclusion and Future Directions

Despite the relatively large user base and various applications built on top of LinkedCT, the project is far from complete. In Section 2, we presented an honest description of the current system architecture, including a few shortcomings of the platform such as a three-layer process for generating RDF, and a SPARQL endpoint that only allows simple queries with small output. A list of known issues is available on our issue tracker at <https://code.google.com/p/linkedct/issues>. Some of these shortcomings resulted in duplicate efforts in the community, for example Bio2RDF’s inclusion of ClinicalTrials.gov data in its latest release [9], which is based on a static one-time processing of the XML source as in our initial release [11], although the mapping seems to be of a high quality. As a result, we recently made LinkedCT’s Web server code open-source to not only build a community to maintain the project, but also to expand the features in the Web server and build in-house user-contributed applications. The code is available on GitHub at <https://github.com/oktie/linkedct>. We will also maintain a list of projects contributed by users and application scenarios, and will be open to new proposals. Examples of applications include a geographical search interface showing trials on a certain condition in a given proximity on Google Maps, and a fuzzy keyword search interface powered by SRCH2 (<http://srch2.com>).

Without a doubt, various healthcare applications that rely on LinkedCT data are critical to the success of the project. An important use case of the data is facilitating matching of patients with clinical trials. Previous work has shown promising results, but using custom transformations and the original XML data [12, 16, 18]. It would be interesting to see how LinkedCT can be used in such scenarios and with real patient data. Use cases requiring reasoning may also need an extension of the ontology or its mapping to an existing domain ontology such as Ontology of Clinical Research [22]. Another interesting study that becomes possible as a result of LinkedCT data is a longitudinal study over trials using the RDF data dumps that are published monthly since 2013. To further facilitate commercial applications and use cases developed by commercial entities, we have changed the data license from CC-BY-SA-NC to Open Data Commons Open Database License (ODbL) that allows non-restricted commercial use.

References

1. Django Web Framework. <https://www.djangoproject.com/>
2. 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)

3. Anderson, P., Thor, A., Benik, J., Raschid, L., Vidal, M.: PANg: finding patterns in annotation graphs. In: SIGMOD, pp. 677–680 (2012)
4. Berners-Lee, T.: Linked Data - Design Issues (2006). <http://www.w3.org/DesignIssues/LinkedData.html> (accessed April 27, 2015)
5. Bizer, C., Cyganiak, R.: D2R server - publishing relational databases on the semantic web. In: ISWC Posters and Demonstrations Track (2006)
6. Boyce, R.D., et al.: Dynamic Enhancement of Drug Product Labels to Support Drug Safety, Efficacy, and Effectiveness. *J. Biomedical Semantics* **4**, 5 (2013)
7. Buneman, P., Cheney, J., Tan, W.C., Vansummeren, S.: Curated databases. In: PODS, pp. 1–12 (2008)
8. Califf, R.M., Zarin, D.A., Kramer, J.M., Sherman, R.E., Aberle, L.H., Tasneem, A.: Characteristics of Clinical Trials Registered in ClinicalTrials.gov, 2007–2010. *JAMA* **307**(17), 1838–1847 (2012)
9. Dumontier, M., et al.: Bio2RDF release 3: a larger, more connected network of linked data for the life sciences. In: ISWC, pp. 401–404 (2014)
10. Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R.J., Wang, M.: A framework for semantic link discovery over relational data. In: CIKM, pp. 1027–1036 (2009)
11. Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R.J., Wang, M.: LinkedCT: A Linked Data Space for Clinical Trials. Technical Report CSRG-596, University of Toronto, August 2009
12. Huang, Z., ten Teije, A., van Harmelen, F.: SemanticCT: a semantically-enabled system for clinical trials. In: Riaño, D., Lenz, R., Miksch, S., Peleg, M., Reichert, M., ten Teije, A. (eds.) KGC 2013 and ProHealth 2013. LNCS, vol. 8268, pp. 11–25. Springer, Heidelberg (2013)
13. Jentzsch, A., Andersson, B., Hassanzadeh, O., Stephens, S., Bizer, C.: Enabling tailored therapeutics with linked data. In: Proceedings of the WWW 2009 Workshop on Linked Data on the Web (LDOW 2009) (2009)
14. Jentzsch, A., Zhao, J., Hassanzadeh, O., Cheung, K.-H., Samwal, M., Andersson, B.: Linking open drug data. In: I-SEMANTICS (2009)
15. Laine, C., et al.: Clinical Trial Registration: Looking Back and Moving Ahead. *New England Journal of Medicine* **356**(26), 2734–2736 (2007)
16. MacKellar, B., et al.: Patient-oriented clinical trials search through semantic integration of linked open data. In: IEEE ICCI*CC, pp. 218–225 (2013)
17. Novack, G.D.: Clinical Trial Registry-Update. *Ocular Surface* **7**(4), 212–4 (2009)
18. Patel, C., Cimino, J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching patient records to clinical trials using ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 816–829. Springer, Heidelberg (2007)
19. Prayle, A.P., et al.: Compliance with Mandatory Reporting of Clinical Trial Results on ClinicalTrials.gov: Cross Sectional Study. *BMJ* **344** (2012)
20. Ross, J.S., et al.: Publication of NIH Funded Trials Registered in ClinicalTrials.gov: Cross Sectional Analysis. *BMJ* **344** (2012)
21. Sonntag, D., Setz, J., Ahmed-Baker, M., Zillner, S.: Clinical trial and disease search with ad hoc interactive ontology alignments. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 674–686. Springer, Heidelberg (2012)
22. Tu, S.W., et al.: OCRE: an ontology of clinical research. In: 11th International Protege Conference (2009)

23. Yeganeh, S.H., Hassanzadeh, O., Miller, R.J.: Linking semistructured data on the web. In: WebDB (2011)
24. Zarin, D.A., et al.: Trial Registration at ClinicalTrials.gov between May and October 2005. *New England Journal of Medicine* **353**(26), 2779–2787 (2005)
25. Zarin, D.A., et al.: The ClinicalTrials.gov Results Database-Update and Key Issues. *New England Journal of Medicine* **364**(9), 852–860 (2011)
26. Zaveri, A., et al.: ReDD-observatory: using the web of data for evaluating the research-disease disparity. In: WI, pp. 178–185 (2011)

Linked Data

DBpedia Commons: Structured Multimedia Metadata from the Wikimedia Commons

Gaurav Vaidya¹✉, Dimitris Kontokostas², Magnus Knuth³, Jens Lehmann²,
and Sebastian Hellmann²

¹ University of Colorado Boulder, Colorado, USA
gaurav.vaidya@colorado.edu

² University of Leipzig, Computer Science, AKSW, Leipzig, Germany
{Kontokostas,Lehmann,Hellmann}@informatik.uni-leipzig.de

³ Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
magnus.knuth@hpi.de

Abstract. The Wikimedia Commons is an online repository of over twenty-five million freely usable audio, video and still image files, including scanned books, historically significant photographs, animal recordings, illustrative figures and maps. Being volunteer-contributed, these media files have different amounts of descriptive metadata with varying degrees of accuracy. The DBpedia Information Extraction Framework is capable of parsing unstructured text into semi-structured data from Wikipedia and transforming it into RDF for general use, but so far it has only been used to extract encyclopedia-like content. In this paper, we describe the creation of the DBpedia Commons (DBC) dataset, which was achieved by an extension of the Extraction Framework to support knowledge extraction from Wikimedia Commons as a media repository. To our knowledge, this is the first complete RDFization of the Wikimedia Commons and the largest media metadata RDF database in the LOD cloud.

Keywords: Wikimedia commons · DBpedia · Multimedia · RDF

1 Introduction

Wikipedia is the largest and most popular open-source encyclopedia project in the world, serving 20 billion page views to 400 million unique visitors each month¹. Wikipedia has over 200 language editions, from the English Wikipedia (4.6 million articles) to the Tumbuka Wikipedia (177 articles). Every article contains metadata such as its page title, its list of contributors, the categories it belongs to, and the other articles it links to. Articles may also contain structured data, such as the latitude and longitude of geographically situated articles. Since 2007, the DBpedia project has been extracting this metadata and structured data and making it publicly available as RDF [3].

¹ <http://reportcard.wmflabs.org/>

Until recently, this extracted data had almost no information on the media files used to illustrate articles. While some media files are stored within a particular language edition of Wikipedia, over twenty-five million of them are located in a centralized repository known as the Wikimedia Commons². The Wikimedia Commons acts as a media backend to all of Wikipedia; media files uploaded to it under an open-access license can be easily inserted into articles in any language. Metadata and structured data associated with the files are stored on the Wikimedia Commons' MediaWiki instance, in a format similar to that used by Wikipedia. This will likely be superseded by Wikidata, the Wikimedia Foundation's new structured data store, but this project is still under discussion³. We make this large and well maintained media resource accessible for semantic tools by extending the DBpedia Extraction Framework to read data from the Wikimedia Commons in addition to other Wikipedia language editions.

In this paper, we describe the dataset and the extraction process required to provide *DBpedia Commons* (DBC). We report on the extensions to the DBpedia Information Extraction Framework (DIEF) to support File pages, multiple languages on the same page, and proper Wikimedia Commons media URL construction. In addition we describe the ontological changes we made in the DBpedia ontology for annotating media files and the additional external vocabularies we chose for the media representation. To our knowledge, this is the first complete RDFization of the Wikimedia Commons and the largest media metadata RDF database in the LOD cloud.

2 Wikimedia Commons

The Wikimedia Commons follows many of the same conventions as Wikipedia itself: regular pages can contain textual content and embedded media files, pages may be placed in more than one category, and namespaces allow project and policy pages to be separated from content pages. Two main differences distinguish the Wikimedia Commons from Wikipedia: (a) Every Wikipedia edition is written entirely in a single language. The Wikimedia Commons is designed to be used by users of every language: where possible, page content is written in multiple languages so that it can be understood by all these users. (b) Most Wikipedia content is in its page content, i.e. its articles. Most Wikimedia Commons content is associated with individual files in the `File` namespace: thus, rather than describing a subject, as Wikipedia articles do, most Wikimedia Commons content describes a media file.

Our strategy for extracting data from Wikimedia Commons content therefore focused on extracting as much information as possible for each page from the `File` namespace. Since the DBpedia Extraction Framework can already extract content from MediaWiki archival dumps, we decided to modify it to support extracting content from archival dumps of the Wikimedia Commons⁴. Note that

² <http://commons.wikimedia.org/>

³ See <https://commons.wikimedia.org/wiki/Commons:Wikidata> and https://www.mediawiki.org/wiki/Multimedia/Structured_Data

⁴ Such dumps are created monthly at <http://dumps.wikimedia.org/commonswiki/>

this means the extraction framework never examines the media files directly; instead, it uses MediaWiki's dump format to infer statements about them.

3 Wikimedia Commons Extraction

We identified three kinds of data that we were interested in: (1) File Metadata, (2) Page Metadata, and (3) Content Metadata. File metadata describes the file that has been uploaded to the Wikimedia Commons, such as its encoding format, image dimensions and file size; these are stored in the backend database used by the MediaWiki software that runs the Wikipedia websites. Page metadata is stored for each MediaWiki page, including those that describe files. This includes the page title, the list of contributors and a history of changes. Finally, the content metadata is stored on the MediaWiki page itself: this includes a list of outgoing external and internal links, the list of categories the page belongs to as well as standard templates that allowed descriptions, sources, authority information and latitude and longitude of the subject of the page to be stored. This is often stored in a MediaWiki template, such as `{{Information}}`. After investigating the available file metadata⁵, we decided to focus on Page and Content Metadata, as File metadata would require parsing the database dumps separately, necessitating much new software development. Unfortunately, this means that we cannot currently provide the dimensions or size of Wikimedia Commons files.

The DBpedia Information Extraction Framework (DIEF) has support for reading MediaWiki XML exports. DIEF was modified to read monthly backups of the Wikimedia Commons. Many of the extractors used to extract page metadata from Wikipedia [2] functioned flawlessly on the Wikimedia Commons dump, extracting titles, categories, authors and other page and content metadata and transforming them into RDF with only minor changes. Four new File Extractors targeting Wikimedia Commons-specific information were developed (Section 3.1). The DBpedia mapping-based extractor was adapted to work on Wikimedia Commons media and creator pages (Section 3.2). We used this extractor to obtain licensing information through the mapping-based extraction.

IRI Scheme. By using the <http://commons.dbpedia.org> domain and following the existing naming strategy of DBpedia, the DBC resources are published under the <http://commons.dbpedia.org/resource/> namespace. For example, <http://commons.dbpedia.org/resource/File:DBpediaLogo.svg>.

3.1 Media Extractors

FileTypeExtractor. The `FileTypeExtractor` guesses the media MIME type by examining its file extension, and uses a preconfigured index to assign both the direct type and the transitive closure of the direct type using `rdf:type`

⁵ https://www.mediawiki.org/wiki/Manual:Image_table

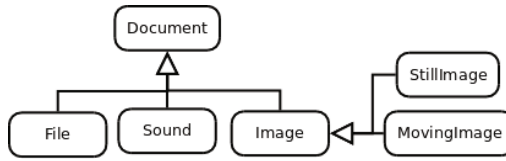


Fig. 1. Hierarchy of main Document classes

(cf. Figure 1 and Section 4). 354,873 files could not be identified by their file extension; an expansion of the preconfigured index will be necessary to include them. The direct type is also linked with `dct:type`. `dct:format` captures the MIME type according to RFC 6838⁶. The file extension is directly queryable with `dbo:fileExtension`. In addition, we provide `dbo:fileURL` for access to the final media URL and `dbo:thumbnail` and `foaf:depiction` for still images. This extractor also provides links to the image itself by using the special page `Special:FilePath`, which provides redirects to the image file. A sample output of this extractor is:

```

1 @prefix db-com: <http://commons.dbpedia.org/resource/File:>.
2 @prefix commons-path: <http://commons.wikimedia.org/wiki/Special:FilePath/>.
3 db-com:DBpediaLogo.svg a dbo:StillImage, dbo:Image, foaf:Image, dbo:File,
4   dbo:Document, foaf:Document, schema:CreativeWork, dbo:Work ;
5   dct:type dbo:StillImage
6   dct:format "image/svg+xml";
7   dbo:fileExtension "svg" ;
8   dbo:fileURL commons-path:DBpediaLogo.svg;
9   dbo:thumbnail commons-path:DBpediaLogo.svg?width=300;
10  foaf:depiction commons-path:DBpediaLogo.svg.

```

GalleryExtractor. The Wikimedia Commons and Wikipedia both support galleries that make it easy to display a series of related images in a compact format⁷. On Wikimedia Commons, this may be used to display a representative set of images about a single topic, such as the page for Colorado⁸. The GalleryExtractor identifies galleries embedded in pages, extracts the list of individual media items, and links them to the page resource with `dbo:galleryItem`.

```

1 db-com:Colorado dbo:galleryItem
2   db-com:2006_C0_Proof.png, db-com:Anasazi_Heritage_Center.jpg,
3   db-com:Bearlakeinspring2.jpg, db-com:Beol_court25.jpg .

```

ImageAnnotationExtraction. The Wikimedia Commons may contain additional annotations for parts of a still image using `{{ImageNote}}` and related templates. The annotations mark a rectangular region within the image and provide a description text in MediaWiki syntax, which may in turn contain hyperlinks to other resources. We extract this information using the W3C *Media*

⁶ <http://tools.ietf.org/html/rfc6838>

⁷ http://en.wikipedia.org/wiki/Help:Gallery_tag

⁸ <http://commons.wikimedia.org/wiki/Colorado>

Fragments [5] vocabulary. The annotated box is identified by a separate IRI that is linked to the original resource through `dbo:hasAnnotation`. As seen in the example below, the new IRI is based on the original resource by suffixing the the coordinates of the part of the image being annotated as well as its total width and height, extracted from the `{{ImageNote}}` template, which is necessary in case the original image needs to be scaled.

```

1 @prefix ann: <http://commons.wikimedia.org/wiki/Special:FilePath/Yes_concert.jpg?width
   =1514&height=1024#xywh=pixel:539,380,110,108>.
2 db-com:Yes_concert.jpg dbo:hasAnnotation ann: .
3 ann: "Jon Anderson"@en .

```

CommonsKMLExtractor. Keyhole Markup Language or KML⁹ is an XML format used to describe map overlays, allowing images of maps to be precisely georeferenced to a location on the planet. The CommonsKMLExtractor extracts the KML data from Wikimedia Commons and stores them as an `rdfs:XMLLiteral` value of the `dbo:hasKMLData` property.

```

1 db-com:Yellowstone_1871b.jpg dbo:hasKMLData ""
2 <?xml version=1.0 encoding=UTF-8?>
3 <kml xmlns="http://earth.google.com/kml/2.2">
4 <GroundOverlay> <!-- KML data --> </GroundOverlay></kml>""^rdfs:XMLLiteral .

```

3.2 Infobox to Ontology Mappings Using the Mapping Extractor

The DBpedia Information Extraction Framework (DIEF) has a sophisticated system for extracting infoboxes from Wikipedia articles. An ‘infobox’ is a special template that stores semi-structured data about the subject of an article. For example, `{{Infobox person}}` may record the birth date and location of the person, while `{{Infobox book}}` might record the ISBN and OCLC number of the book. The DBpedia Mapping Extractor allows contributors to the DBpedia Mappings Wiki¹⁰ to describe how template properties map to properties on the DBpedia ontology [3, Sec.2.4].

A similar set of templates provides information on the Wikimedia Commons; for example, the `{{Location}}` template stores the location that is the subject of a media file, such a building being photographed or a city being mapped. A new DBpedia mapping namespace for the Wikimedia Commons was created¹¹ and DIEF was refactored to extract templates from media file and creator pages and use DBpedia mappings to convert them to RDF statements.

License Extraction. Licenses are encoded in the Wikimedia Commons as templates, e.g. the template `{{cc-by-sa}}` present on a `File` page indicates that the media file has been licensed under the Creative Commons BY-SA license.

⁹ <https://developers.google.com/kml/documentation/>
¹⁰ <http://mappings.dbpedia.org/>
¹¹ <http://mappings.dbpedia.org/index.php/Mapping-commons>

Table 1. Description of the DBC datasets

| Title | Triples | Description |
|---------------------|-------------|---|
| Labels | 29,203,989 | Labels for resources |
| Provenance | 272,079,712 | Provenance information (pageIDs, revisionIDs) |
| SKOS | 94,701,942 | SKOS hierarchy based on the category hierarchy |
| Geo data | 18,400,376 | Geo coordinates for the media files |
| File Information | 414,118,159 | File metadata |
| Annotations | 721,609 | Image annotations |
| Galleries | 2,750,063 | Image galleries |
| Types | 111,718,049 | Resource types |
| KML | 151 | KML data |
| Mappings | 95,733,427 | Mapped infobox data |
| Infobox | 87,846,935 | Unmapped Infobox data |
| Interlanguage links | 4,032,943 | Links to other DBpedia editions |
| Internal links | 116,807,248 | Internal links to other Wikimedia Commons pages |
| External links | 17,319,980 | Links to external resources |
| Metrics | 58,407,978 | Article metadata (in/out degree, page size) |
| Templates | 77,220,130 | Template metadata and usage |

We used the Mapping Extractor described above to map each template to a URL describing the license, such as <https://creativecommons.org/licenses/by-sa/2.0/>. However, it is a common practice on the Wikimedia Commons to nest and embed multiple licenses together: for example, the template instruction `{{self|cc-by-sa-4.0}}` indicates that this file was created by the uploader (‘self’) who has licensed it under a Creative Commons CC-BY 4.0 license. Since nested or wrapped templates are not currently supported in DIEF, we added a pre-processing extraction step to unwrap license templates specifically to make all license mappings identifiable to the Mapping Extractor.

```
1 db-com:DBpediaLogo.svg dbo:license <http://creativecommons.org/publicdomain/mark/1.0/>
```

4 Dataset

A general overview of the datasets provided by DBC is provided in Table 1, where each row provides a summary of one or more similar datasets. A total of 1.4 billion RDF triples were inferred from the Wikimedia Commons dump prepared in January 2015, describing almost 30 million unique IRIs. A diagram for the new classes we introduced for Wikimedia Commons media files is depicted in Figure 1: `dbo:Document` has the subclasses `dbo:File`, `dbo:Sound`, and `dbo:Image`. A `dbo:Image` can be a `dbo:StillImage` (e.g. picture) or a `dbo:MovingImage` (e.g. video). DBC mostly consists of still images (Table 2) with JPEG as the most popular format (Table 4). Table 3 provides the most frequent properties in DBC while Table 5 lists the most common media licenses. One of the largest datasets are the *mappings* (95.7M triples) which is based on the infobox to ontology mappings (Section 3.2), and so include the license information. The authors, with contributions from the DBpedia community, invested significant effort to ensure that 90% of all occurrences of infobox templates and 78% of all template parameters on the Wikimedia Commons have either been mapped to an RDF entity in the DBpedia ontology or have been determined to have no structured data.¹²

¹² <http://mappings.dbpedia.org/server/statistics/commons/>, as of April 25., 2015

Table 2. Top classes

| Count | Class |
|------------|-----------------|
| 25,061,835 | dbo:StillImage |
| 611,288 | dbo:Artwork |
| 90,011 | dbo:Agent |
| 49,821 | dbo:MovingImage |
| 19,126 | dbo:Person |

Table 4. Top MIME types.

| Count | MIME type |
|------------|-----------------|
| 20,880,240 | image/jpeg |
| 1,457,652 | image/png |
| 878,073 | image/svg+xml |
| 455,947 | image/tiff |
| 246,149 | application/pdf |

Table 3. Top properties

| Count | Property |
|------------|-------------------|
| 73,438,813 | dct:subject |
| 43,209,414 | dbo:license |
| 29,201,812 | dce:language |
| 24,496,724 | dbo:fileURL |
| 24,496,706 | dbo:fileExtension |

Table 5. Top licenses

| Count | License |
|-----------|---------------|
| 7,433,235 | CC-by-sa v3.0 |
| 4,096,951 | CC-pd v1.0 |
| 3,704,043 | GNU-fdl v1.2 |
| 3,681,840 | GNU-fdl |
| 2,116,411 | CC-by-sa v2.0 |

Access and Sustainability. DBpedia Commons is part of the official DBpedia knowledge infrastructure and is published through the regular releases of DBpedia along with the rest of the DBpedia language editions. The first DBpedia release that included this dataset is *DBpedia 2014*¹³. DBpedia is a pioneer in adopting and creating best practices for Linked Data and RDF publishing. Thus, being incorporated into the DBpedia publishing workflow guarantees: (a) long-term availability through the DBpedia Association and the Leipzig Computer Center long-term hosting platform and (b) a shared codebase with the DBpedia Information Extraction Framework. Besides the stable dump availability we created <http://commons.dbpedia.org> for the provision of a Linked Data interface [4], a SPARQL Endpoint and more frequent dataset updates. The dataset is registered in DataHub¹⁴ and provides machine readable metadata as void¹⁵ and DataID¹⁶ [1]. Since the project is now part of the official DBpedia Information Extraction Framework, our dataset reuses the existing user and developer support infrastructure, e.g. the general discussion and developer list as well as the DBpedia issue tracker for submitting bugs.

5 Use Cases

In the following, we provide several existing or possible use cases of the DBC dataset.

Galleries, Libraries, Archives and Museums. Collectively known as GLAMs, such institutions hold large repositories of documents, photographs, recordings and artifacts. Several have made large contributions of media to the Wikimedia Commons, such as the [128,000 images donated](#) by the National Archives and Records Administration of the United States, containing rich metadata stored using the `{{NARA-image-full}}` template.

¹³ <http://downloads.dbpedia.org/2014/commons>

¹⁴ <http://datahub.io/dataset/dbpedia-commons>

¹⁵ <http://commons.dbpedia.org/void.ttl>

¹⁶ <http://dbpedia.s16a.org/commons.dbpedia.org/20150110/dataid.ttl>

By [mapping parameters in this template](#) to properties in the DBpedia Ontology, we were able to quickly obtain descriptions, authors, notes and local identifiers in RDF for this media¹⁷. DBC provides a community-edited source of structured data in RDF that can exist in parallel with any structured data being published directly by a GLAM.

Image Recognition Algorithms. Over 96,000 images on the Wikimedia Commons have embedded annotations¹⁸. By making the coordinates of these annotations available through our Image annotations dataset, we provide a training dataset that can be used to teach machine-learning algorithms to identify annotations that may be of interest to Wikimedia Commons editors.

License Extraction. Media uploaded to the Wikimedia Commons must either be in the public domain or licensed under an open-access licenses, but individual language Wikipedias may allow users to upload unlicensed images as long as they have a fair-use rationale for them. This means that not all media files embedded in Wikipedia articles may be freely reused. Furthermore, different open-access licenses have different requirements for re-use: some allow any re-use as long as the original creator is cited, while others require any derivative works to carry the same license as the original. Since licenses on the Wikimedia Commons are encoded by licensing template, we were able to use the Mapping Extractor (Section 3.2) to provide license URLs for several million media files. This allows licensing conditions for many Wikimedia Commons media files to be determined automatically. In particular, the German National Library contacted some of the authors specifically for this metadata: they included Wikimedia Commons images in their Linked Data interface and were interested in displaying the license information directly there using our license dataset. This integration is not yet deployed.

6 Conclusions and Future Work

We introduced DBpedia Commons, to our knowledge the first large-scale knowledge extraction from Wikimedia Commons. We present the adaptations and additions made to the DBpedia Information Extraction Framework to facilitate the correct extraction of media files and their metadata, including license information. The dataset contains 1.4 billion RDF triples that provide file metadata, provenance, descriptions, and license information.

Acknowledgments. This work was funded by the [Google Summer of Code 2014 program](#) and by grants from the EU's 7th & H2020 Programmes for projects ALIGNED (GA 644055) and GeoKnow (GA 318159).

¹⁷ See http://commons.dbpedia.org/resource/File:Douglas_MacArthur_lands_Leytel.jpg for an example.

¹⁸ https://commons.wikimedia.org/wiki/Category:Images_with_annotations

References

1. Brümmer, M., Baron, C., Ermilov, I., Freudenberg, M., Kontokostas, D., Hellmann, S.: DataID: towards semantically rich metadata for complex datasets. In: Proc. of the 10th International Conference on Semantic Systems, pp. 84–91. ACM (2014)
2. Kontokostas, D., Bratsas, C., Auer, S., Hellmann, S., Antoniou, I., Metakides, G.: Internationalization of Linked Data: The case of the Greek DBpedia edition. *Web Semantics: Science, Services and Agents on the WWW* **15**, 51–61 (2012)
3. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal* (2014)
4. Lukovnikov, D., Stadler, C., Kontokostas, D., Hellmann, S., Lehmann, J.: DBpedia viewer - an integrative interface for DBpedia leveraging the DBpedia service ecosystem. In: Proc. of the Linked Data on the Web 2014 Workshop (2014)
5. Troncy, R., Mannens, E., Pfeiffer, S., Deursen, D.V.: Media fragments URI. W3C Recommendation, September 2012. <http://www.w3.org/TR/media-frag/>

Archiving and Publishing Scientific Data

Provenance-Centered Dataset of Drug-Drug Interactions

Juan M. Banda¹(✉), Tobias Kuhn^{2,3}, Nigam H. Shah¹, and Michel Dumontier¹

¹ Stanford University - Center for Biomedical Informatics Research,
1265 Welch Road, Stanford, CA 94305, USA

{jmbanda,nigam,michel.dumontier}@stanford.edu

² Department of Humanities, Social and Political Sciences, ETH Zurich,
Zürich, Switzerland

tokuhn@ethz.ch

³ Department of Computer Science, VU University Amsterdam,
Amsterdam, Netherlands

Abstract. Over the years several studies have demonstrated the ability to identify potential drug-drug interactions via data mining from the literature (MEDLINE), electronic health records, public databases (Drugbank), etc. While each one of these approaches is properly statistically validated, they do not take into consideration the overlap between them as one of their decision making variables. In this paper we present LInked Drug-Drug Interactions (LIDDI), a public nanopublication-based RDF dataset with trusty URIs that encompasses some of the most cited prediction methods and sources to provide researchers a resource for leveraging the work of others into their prediction methods. As one of the main issues to overcome the usage of external resources is their mappings between drug names and identifiers used, we also provide the set of mappings we curated to be able to compare the multiple sources we aggregate in our dataset.

Keywords: Drug-drug interactions · Nanopublications · Data mining

1 Introduction

Studies analyzing costs over time have shown that adverse drug reactions (ADRs) cost over \$136 billion a year [13]. One significant cause of ADRs are drug-drug interactions (DDIs) which greatly affect older adults due to the multiple drugs they are taking [3]. A DDI occurs when the effect of any given drug is altered by another drug which results in an unpredictable effect. While new drugs, before market approval, are tested in both *in vivo* and *in vitro* methods [19], it is unfeasible to test their interactions with all other approved and experimental drugs. In the recent years, computational approaches have been trying to infer potential DDI signals using a wide variety of sources [6, 9, 16, 17], however these methods produce thousands of statistically plausible predictions [6, 8], thus making the task of testing them in an experimental setting impractical.

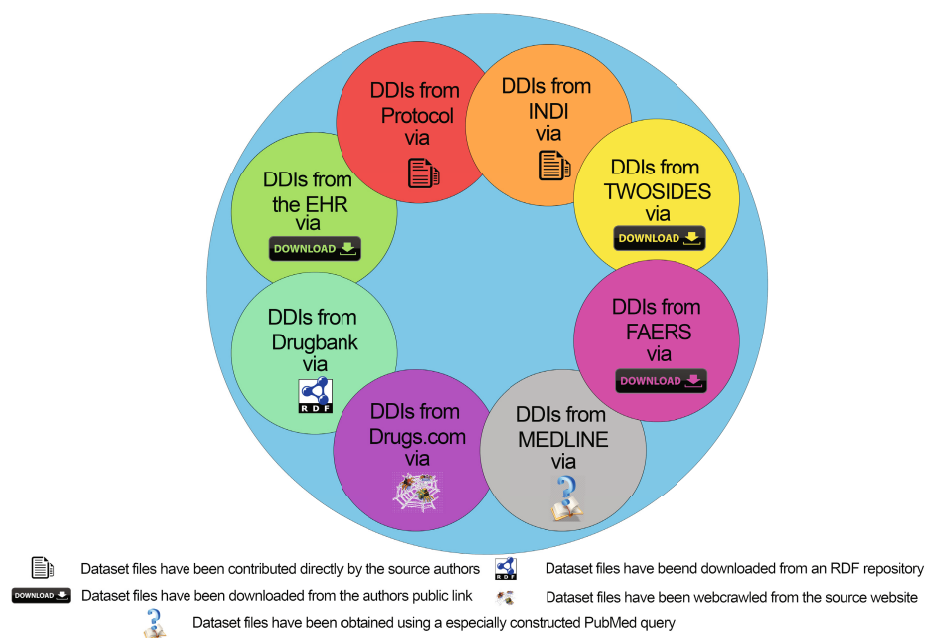


Fig. 1. Overview of the Linked Drug-Drug Interactions dataset and the data sources that it incorporates and their original formats

We are therefore facing the need of combining data from various disparate sources, but at the same time have to remain aware of their provenance due to differences in individual quality and overall confidence. Depending on their primary source and their provenance path, two data entries about the same DDI might substantially increase our confidence about a relation. By relying on Semantic Web technologies such as RDF – and more specifically nanopublications [7] – these provenance paths can be represented in an explicit and uniform manner. In this paper we introduce LInked Drug-Drug Interactions (LIDDI), a dataset that consolidates multiple data sources in one cohesive linked place that allows researchers to have immediate access to multiple collections of DDI predictions from public databases and reports, biomedical literature, and methods that take different and sometimes more comprehensive approaches.

With great potential for use in drug safety studies, LIDDI provides the linking components to branch from DDIs into individual properties of each drug via Drugbank and UMLS, as well as mappings to other types of biomedical ontologies which are used to describe drug-drug interactions. In terms of direct applications, the resources in our dataset allow researchers to quickly determine if there is support by other DDI sources for their own predictions, allowing them to evaluate them via consensus rather than only by independent statistical measures (AUROC values, odds ratios, etc). LIDDI offers all the mappings needed to bridge the incorporated resources into a single comparable entity, providing extra value

for researchers looking to bridge the data sources we have connected, for example Drugbank drug identifiers to UMLS Concept Unique Identifiers (CUI) to Medical Subject Headings (MeSH) codes, opening endless possibilities for reuse.

To our knowledge, this is the first public dataset exemplifying the use of nanopublications for the integration of knowledge from multiple diverse data sources in the field of drug safety surveillance. We believe it will facilitate cross-referencing of results between different domains not previously available, as well as the enrichment of drugs involved in the DDIs thanks to the linkages provided.

2 Methods

2.1 Nanopublications with Trusty URIs

Nanopublications [7] are a concept to use Semantic Web techniques (most importantly RDF and named graphs) to closely link data to their provenance and meta-data in a uniform manner. The vision is that small data packages should become the primary format for research outputs instead of narrative articles [15]. Technically, a nanopublication consists of an assertion graph with triples expressing an atomic statement (about drug-drug interactions in our case), a provenance graph that reports how this assertion came about (e.g. where it was extracted from or what mechanism was used to derive it), and a publication information graph that provides meta-data for the nanopublication (such as its creators and a timestamp).

Trusty URIs [11,12] are a recent proposal to make URI links for digital artifacts verifiable, immutable, and permanent. Such URI identifiers contain a cryptographic hash value calculated on the content of the represented digital artifact — i.e. RDF content of nanopublications in our case — in a format-independent way. With trusty URIs, any reference to an artifact thereby comes with the possibility to verify with 100% confidence that a retrieved file really represents the correct and original state of that resource.

2.2 Data Schema

We represent DDIs in the following way: We instantiate the DDI class from the Semantic science Integrated Ontology (SIO) [5]. We define a vocabulary in our own namespace, as is normally done with Bio2RDF [4], to assign domain-specific predicates to point from the DDI to each drug, and to the clinically-relevant event resulting from the drug interaction. This assertion is stored in the assertion graph of the nanopublication. We also create nanopublications to capture the mappings between each drug and our seven sources. We use BioPortal PURL identifiers for UMLS URIs. We use the PROV ontology [14] to capture the provenance of each DDI by linking the assertion graph URI to a PROV activity. This activity is linked to the software used to generate the drug and event mappings, as well as to the citation for the method, the generation time, and the dataset where the DDIs were extracted from. In this last case we also provide a direct link to the original dataset used. Figure 2 highlights our general schema with a particular example for a DDI that contains a drug-drug set.

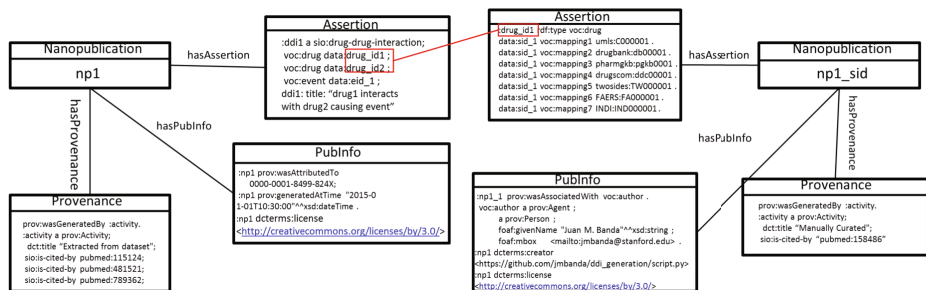


Fig. 2. Dataset Schema shows two types of nanopublications included in the dataset. The first captures the DDI, and the second captures drug mappings. A third one not show captures the event mappings.

3 Data Sources Aggregated

In this section we provide details about the transformations we performed on each of the data sources to be able to normalize them and use them together. Due to space constraints we provide a very high-level view of how each source procures and generates its data as this lies out of the scope of this dataset paper. All the mappings provided within our dataset for UMLS concept unique identifiers (CUI), RxNORM codes, MeSH codes and Drugbank identifiers, were done as an initial step once we acquired all our sources.

Electronic Health Records (EHR). In [9] the authors used Stanford Translational Research Integrated Database Environment (STRIDE) dataset comprising 9 million unstructured clinical notes corresponding to 1 million patients that span a period of 18 years (1994–2011) to mine DDIs from the EHR. Published as a comma delimited file in the supplemental materials, we use the highest confidence predictions as the initial basis of our dataset in terms of drugs (345) and events (10). This dataset provides drug-drug-event sets of UMLS concept unique identifiers (CUI), drug/event names in string literals.

FDA Adverse Event Reporting System (FAERS). We analyzed over 3.2 million reports found on the FDA Adverse Event Reporting System (FAERS), since there are no established algorithmic ways of determining the statistical significance of a drug-drug interaction from FAERS reports without any additional external information, we set a threshold of each DDI to have at least ten appearances in different FAERS reports for it to appear on our dataset. In this data source, the drugs are given in string literals that normalized to RxNorm drug names and the events normalized to MeDRA.

DrugBank. We used the RDF versions of Drugbank [18] made publicly available through Bio2RDF. The used version was dated as 2013-07-25. We proceeded to strip the Drugbank identifiers and their labels into string literals mapped

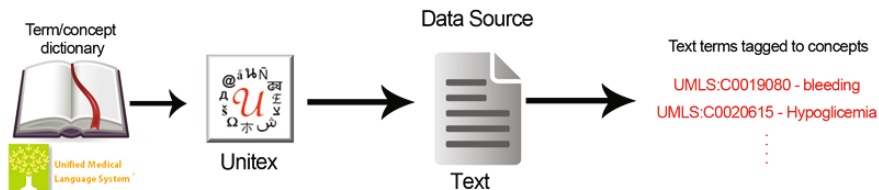


Fig. 3. Text-tagging pipeline. We build a term/concept dictionary from ontologies in UMLS. We then proceed to tag concepts found in any piece of text from any source via Unitex, resulting in a bag of UMLS CUIs for each input.

into RxNorm/SNOMED string literals. In order to extract DDIs we used the `drugbank_vocabulary:Drug-Drug-Interaction` type to match interacting drugs, we then used the `rdfs:label` to extract the event by concept tagging the indication section with a modified version of the NCBO annotator used in [9]. This concept tagging process is described on Figure 3.

Drugs.com. In order to mine DDIs from this resource, we used a web-crawling agent developed in Python. We programmatically checked the individual drugs.com page for each of the 345 drugs in our set and extracted all their interacting drugs. For each of the interacting drugs we build the event part of our DDI and by annotating the complementary free-text section, the same way as on Figure 3, which describes the interaction (similar to the Drugbank indication).

MEDLINE. In order to find potential DDIs signals in the biomedical literature, we have adapted a clever method developed by Avilach et al. [1], that allows for the use of a query with certain MeSH terms and qualifiers to find adverse events in the literature. We modified this query by adding a MeSH term for “Drug interactions” and modified the search to allow for two drugs instead of one from the original approach. These modifications of the method with some manipulation produces drug-drug-event sets.

TWOSIDES. Provided as a download by Tatonetti et al. [16], TWOSIDES contains polypharmacy side effects for pairs of drugs. Mined from FDAs FAERS, this method is designed to complement modern signal detection approaches, providing stratification, dampening or removing the effect of co-variables, without needing to divide drug-exposed reports into strata. This resource is available in a CSV format and features drugs in RxNorm normalized string literal form and events in UMLS CUI form.

Inferring Drug Interactions (INDI). Currently available on a website for checking drug-drug interactions and provided to us by the authors [6] in a comma separated file. This prediction method infers both pharmacodynamic and pharmacokinetic interactions based on their similarity to existing known adverse events that result from a common metabolizing enzyme (CYP). The provided resources consisted of drug-drug sets (no events) in UMLS CUI form.

Table 1. LIDDI DDI event distribution between sources

| Event Name | EHR | MEDLINE | Drugbank | Drugs.com | FAERS | TWOSIDES | INDI* | Protocol* |
|--------------------|--------|---------|----------|-----------|-------|----------|-------|-----------|
| Arrhythmia | 700 | 68 | 286 | 3,148 | 100 | 4,632 | NA* | NA* |
| Bradycardia | 254 | 88 | 408 | 4,896 | 194 | 4,824 | NA* | NA* |
| Hyperkalaemia | 1,888 | 42 | 422 | 4,248 | 146 | 3,840 | NA* | NA* |
| Hypoglycaemia | 1,460 | 386 | 796 | 6,214 | 104 | 5,150 | NA* | NA* |
| Long QT syndrome | 14 | 270 | 334 | 3,510 | 2 | 0 | NA* | NA* |
| Neutropenia | 4,608 | 192 | 402 | 4,218 | 616 | 3,702 | NA* | NA* |
| Pancytopenia | 1,880 | 4 | 270 | 3,146 | 148 | 5,440 | NA* | NA* |
| Parkinsonism | 144 | 0 | 566 | 5,978 | 70 | 884 | NA* | NA* |
| Rhabdomyolysis | 122 | 198 | 392 | 3,842 | 214 | 3,264 | NA* | NA* |
| Serotonin syndrome | 896 | 0 | 384 | 3,960 | 122 | 1,094 | NA* | NA* |
| Total: | 11,966 | 1,248 | 4,260 | 43,160 | 1,716 | 32,830 | 8,370 | 224 |

Similarity-Based Modeling Protocol (Protocol). Provided to us by the authors, Vilar et al. [17], this protocol integrates a gold standard of DDIs with drug similarity information extracted from sources like: 2D and 3D molecular structure, interaction profile, target similarities, and side-effect similarities. This method generates drug interaction candidates that are traceable to pharmacological or clinical effects. We were provided with the resulting drug-drug (no events) sets with Drugbank identifiers.

4 Statistics and Access

Table 1 shows the total number of DDIs found with support in each of the multiple sources we integrated. Note that these are not unique DDIs in the sense that we have a drug1-drug2-event set that is equivalent to drug2-drug1-event set as the directionality does not matter, thus counting them twice. We left this reflexive drug drug interactions in our dataset to conform to the standard employed by bio2RDF’s DrugBank repository, in order to not limit the discovery of any potential interaction when a query is performed on any given single drug.

The dataset in its entirety contains a total of 98,085 nanopublications out of which 345 are used for drug mappings, 10 for event mappings and the remainder for DDIs extracted from the data sources we used. The dataset has a total of 392,340 graphs (four per nanopublications) and 2,051,959 triples, taking 723 MB in nquads representation.

LIDDI can be accessed as a bulk download via figShare [2] and SPARQL endpoint at: <http://http://liddi.stanford.edu:8890/sparql>. The dataset is also made available via the recently installed nanopublication server network [10]. The command ‘np’ from the Java nanopublication library¹ can be used to download the dataset [20]:

```
$ np get -c -o out.trig RA7SuQ0e661LJdKpt5E0S2DKykf1ht9LFmNaZtFSDMrXg
```

¹ <https://github.com/Nanopublication/nanopub-java>

5 Potential for Applications and Future Work


The first application that this dataset has been used for involves ranking DDI predictions from EHR data. By taking the EHR data source published by [9] and using this resource, researchers are ranking predictions in order to determine which ones are more feasible for experimental evaluation. By leveraging such diverse resources, the researchers have been able to use the data sources as voting mechanisms for certain DDIs to be prioritized over others. Similar studies can be performed to determine the priority of the other data sources and their predictions just using LIDDI and selecting a different resource to evaluate. The authors of each data source could enhance their own predictions by using their method and consider other sources we compiled in this dataset as gold-standards or at least 'silver-standards' as there is no de facto community-wide gold standard for drug safety when it comes to drug-drug interactions. Multiple applications can revolve around enhancing the DDIs listed here by enriching the drugs and events with properties available in the Drugbank, RxNorm and MeSH linked graphs that are not easily accessible via traditional means. This might lead to even further linkage of other resources tying all things back to a described DDI and helping to further explain its reason and impact. As more researchers become aware of this resource they should be encouraged to contribute to it by adding their own DDI predictions, thus enhancing the overall availability of DDI predictions sets for scientific comparisons. We will continue to contact authors and map their data sets into LIDDI to have a more diverse and rich set of resources. We also plan at some point incorporate more drugs and more events into the resource as they become available for it to have a wider coverage of the current drug space.

References

1. Avillach, P., Dufour, J.-C., Diallo, G., Salvo, F., Joubert, M., Thiessard, F., Mougín, F., Trifirò, G., Fourrier-Réglat, A., Pariente, A., Fieschi, M.: Design and validation of an automated method to detect known adverse drug reactions in medline: a contribution from the eu-adr project. *Journal of the American Medical Informatics Association* **20**(3), 446–452 (2013)
2. Banda, J.M., Kuhn, T., Shah, N.H., Dumontier, M.: Liddi: Provenance-centered dataset of drug-drug interactions. figshare July 17, 2015. <http://dx.doi.org/10.6084/m9.figshare.1486478>
3. Bushardt, R.L., Massey, E.B., Simpson, T.W., Ariail, J.C., Simpson, K.N.: Polypharmacy: Misleading, but manageable. *Clinical Interventions in Aging*. **3**(2), 383–389 (2008). 18686760[pmid] *Clin Interv Aging*
4. Callahan, A., Cruz-Toledo, J., Ansell, P., Dumontier, M.: Bio2RDF release 2: improved coverage, interoperability and provenance of life science linked data. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) *ESWC 2013. LNCS*, vol. 7882, pp. 200–212. Springer, Heidelberg (2013)
5. Dumontier, M., Baker, C.J., Baran, J., Callahan, A., Chepelev, L.L., Cruz-Toledo, J., Nicholas, R., Rio, D., Duck, G., Furlong, L.I., et al.: The semanticscience integrated ontology (sio) for biomedical research and knowledge discovery. *J. Biomedical Semantics* **5**, 14 (2014)

6. Gottlieb, A., Stein, G.Y., Oron, Y., Ruppin, E., Sharan, R.: Indi: a computational framework for inferring drug interactions and their associated recommendations. *Molecular Systems Biology* **8**, 592–592 (2012). 22806140[pmid] *Mol. Syst. Biol*
7. Groth, P., Gibson, A., Velterop, J.: The anatomy of a nano-publication. *Information Services and Use* **30**(1), 51–56 (2010)
8. Haerian, K., Varn, D., Vaidya, S., Ena, L., Chase, H.S., Friedman, C.: Detection of pharmacovigilance-related adverse events using electronic health records and automated methods. *Clinical pharmacology and therapeutics* **92**(2), 228–234 (2012)
9. Iyer, S.V., Harpaz, R., LePendu, P., Bauer-Mehren, A., Shah, N.H.: Mining clinical text for signals of adverse drug-drug interactions. *J. Am. Med. Inform. Assoc.* **21**(2), 353–362 (2014)
10. T. Kuhn, C. Chichester, M. Krauthammer, and M. Dumontier. Publishing without publishers: a decentralized approach to dissemination, retrieval, and archiving of data. In: *Proceedings of ISWC 2015. Lecture Notes in Computer Science*. Springer (2015)
11. Kuhn, T., Dumontier, M.: Trusty URIs: verifiable, immutable, and permanent digital artifacts for linked data. In: Presutti, V., d’Amato, C., Gandon, F., d’Aquin, M., Staab, S., Tordai, A. (eds.) *ESWC 2014. LNCS*, vol. 8465, pp. 395–410. Springer, Heidelberg (2014)
12. Kuhn, T., Dumontier, M.: Making digital artifacts on the web verifiable and reliable. *IEEE Transactions on Knowledge and Data Engineering* (2015)
13. Lazarou, J., Pomeranz, B.H., Corey, P.N.: Incidence of adverse drug reactions in hospitalized patients: A meta-analysis of prospective studies. *JAMA* **279**(15), 1200–1205 (1998). doi:10.1001/jama.279.15.1200
14. Lebo, T., et al.: PROV-O: The PROV ontology. Recommendation, W3C (2013)
15. Mons, B., van Haagen, H., Chichester, C., den Dunnen, J.T., van Ommen, G., van Mulligen, E., Singh, B., Hooft, R., Roos, M., Hammond, J., et al.: The value of data. *Nature genetics* **43**(4), 281–283 (2011)
16. Tatonetti, N.P., Ye, P.P., Daneshjou, R., Altman, R.B.: Data-driven prediction of drug effects and interactions. *Science Translational Medicine* **4**(125), 125ra31 (2012). doi:10.1126/scitranslmed.3003377
17. Vilar, S., Uriarte, E., Santana, L., Lorberbaum, T., Hripcsak, G., Friedman, C., Tatonetti, N.P.: Similarity-based modeling in large-scale prediction of drug-drug interactions. *Nat. Protocols* **9**(9), 2147–2163 (2014)
18. Wishart, D.S., Knox, C., Guo, A.C., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., Hassanali, M.: Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Research* **36**, D901–D906 (2008). 18048412[pmid] *Nucleic Acids Res*
19. Zhang, L., Zhang, Y., Zhao, P., Huang, S.-M.: Predicting drugdrug interactions: An fda perspective. *The AAPS Journal* **11**(2), 300–306 (2009)
20. Linked Drug-Drug Interactions (LDDI) dataset. Nanopublication index, July 17, 2015. <http://np.inn.ac/RA7SuQ0e661LJdKpt5EOS2DKykf1ht9LFmNaZtFSDMrXg>

The GeoLink Modular Oceanography Ontology

Adila Krisnadhi^{1,8}, Yingjie Hu², Krzysztof Janowicz², Pascal Hitzler¹, Robert Arko³, Suzanne Carbotte³, Cynthia Chandler⁴, Michelle Cheatham¹, Douglas Fils⁵, Timothy Finin⁶, Peng Ji³, Matthew Jones², Nazifa Karima¹, Kerstin Lehnert³, Audrey Mickle⁴, Thomas Narock⁷, Margaret O'Brien², Lisa Raymond⁴, Adam Shepherd⁴, Mark Schildhauer², and Peter Wiebe⁴

¹ Wright State University, Dayton, USA

krisnadhi.2@wright.edu

² University of California, Santa Barbara, USA

³ Lamont-Doherty Earth Observatory, Columbia University, New York, USA

⁴ Woods Hole Oceanographic Institution, Woods Hole, USA

⁵ Consortium for Ocean Leadership, Washington, USA

⁶ University of Maryland, Baltimore County, Baltimore, USA

⁷ Marymount University, Arlington, USA

⁸ Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

Abstract. GeoLink is one of the building block projects within Earth-Cube, a major effort of the National Science Foundation to establish a next-generation knowledge infrastructure for geosciences. As part of this effort, GeoLink aims to improve data retrieval, reuse, and integration of seven geoscience data repositories through the use of ontologies. In this paper, we report on the GeoLink modular ontology, which consists of an interlinked collection of ontology design patterns engineered as the result of a collaborative modeling effort. We explain our design choices, present selected modeling details, and discuss how data integration can be achieved using the patterns while respecting the existing heterogeneity within the participating repositories.

Keywords: Ontology design pattern · Data integration · Geoscience · Collaborative modeling

1 Introduction

Like in other branches of science, data holds a very prominent role in conducting research inquiries in ocean science. A number of synthesis centers sponsored by the National Science Foundation (NSF), such as NCEAS and NESCent, have provided evidences that coupling existing data with interdisciplinary collaboration and analyses can lead to exciting and novel scientific insights, which would be almost impossible to achieve traditionally [3, 5, 11]. This leads to the establishing of ocean (and generally, geo-)science data repositories, such as BCO-DMO, DataONE, and IODP, which contributes to a significant improvement particularly in data preservation. Such data repositories are typically designed to serve

specific parts of the geoscience research community, making data management and quality control more tractable. On the flip side, however, data become highly heterogeneous because of the differences in data formats, methods of access, and nuances in the conceptualization. This can cause frustration for researchers when attempting to find and integrate relevant data from these multiple repositories to perform an integrative analysis [12]. This problem and other related knowledge management problems led to the launching of the EarthCube program by NSF. EarthCube is a major, community-led effort to upgrade cyberinfrastructure for the geosciences consisting of various building block projects and research coordination networks, all aiming to enable extensive cross-discipline data sharing and integration, to allow global data discovery, and to transform the way researchers understand the Earth system via data-enabled geosciences research.

GeoLink¹ is one of the EarthCube building block projects aiming to leverage advances in semantic technologies for developing a data integration and discovery framework involving seven major data repositories, mainly in the area of ocean science. Those repositories are BCO-DMO, DataONE, IEDA, IODP, LTER, MBLWHOI Library, and R2R.² The data integration problem faced by this project is both technically and socially challenging, not just because of the lack of direct alignment between data from different repositories, but also due to fundamental differences in the way data and knowledge are modeled. GeoLink tackles this problem by the use of Linked Data [1] and Ontology Design Patterns (ODPs) [2]. Linked Data enables repositories to describe and publish their data using standard syntax featuring links to other data, possibly in different repositories. Meanwhile, ODPs allows a horizontal integration featuring semantic alignment between repositories with possibly independent semantic models.

In this paper, we present the GeoLink modular ontology, which is actually a collection of ODPs developed for the purpose of data integration in the GeoLink project. Before describing the ontology, we start by explaining key points in our modeling approach using ODPs in section 2. Sections 3 and 4 present the ontology and selected modeling details. Due to space restriction, we cannot present the whole ontology in detail, and refer the reader to the more detailed technical report at <http://schema.geolink.org/>. Section 5 describes the availability and external links of this ontology. Finally, Section 6 summarizes this work.

2 Modeling Approach

The GeoLink project aims to provide a framework for horizontal integration amongst data providers. The project, however, does not advocate the creation of an overarching upper ontology for the ocean science because fundamental differences in data modeling and vocabularies between repositories due to differing subdomains, purposes and requirements prevent the realization of such an ontology. Instead, we set out with developing *ontology design patterns (ODPs)*, or more specifically the so-called *content patterns*, each of which is a self-contained,

¹ <http://www.geolink.org/>

² See <http://www.geolink.org/team.html>.

highly modular ontology encapsulating a particular notion within some domain of discourse and can act as a building block of a more complex ontology [2].

The modeling task was conducted through collaborative modeling sessions, which ensure a good *community engagement* by a very active involvement of oceanographers as domain experts and potential end users. The modeling sessions are intended to bridge language and perspective gaps between ontology engineers, domain experts, and end users, which are bound to occur during an ontology development [7]. In a modeling session, we proceeded by focusing on one notion a time, starting from (i) gathering use cases through a set of competency questions [10]; (ii) identifying and visualizing relevant classes and relationships while keeping within the boundary of the focus notion; and (iii) specifying constraints and axioms, initially in semi-formal natural language expressions. Ontology engineers then continued the work by translating the modeling result from the steps above into a formal ontology, while ensuring no axiom makes an overly strong ontological commitment. Since modeling was focused on one notion at a time, we obtained self-contained, highly modular ontology patterns.

3 Ontology Overview

The GeoLink modular ontology comprises of several content patterns (Figure 1). The majority of the content patterns model some concrete notion deemed important by the participating data providers as it reflects an important discovery facet. These include cruise, person, organization, dataset, funding award, program, etc. A few other content patterns represent some form of abstraction introduced typically as a good modeling practice or as a flexible connector between two other patterns. The remaining patterns are auxiliary content patterns that provide more details to some other content patterns. We briefly present an overview of each of these patterns in the following and the reader is referred to the technical report and our OWL implementation for more details.

We start with the abstract patterns **Agent** and **Agent Role**. The Agent pattern defines a central class **Agent** and allows one to express that an agent (e.g., a person or an organization) may perform a role, which is an instance of the **AgentRole** class. The latter is aligned to the **AgentRole** class in the Agent Role pattern. The Agent Role pattern itself is essentially a reification of relations between an agent and the thing the agent is involved in. For example, a person may participate in a cruise as a chief scientist. Then, using the Agent Role pattern, we express this by stating that a cruise provides a role of type chief scientist that is performed by that person. This reification allows to flexibly cover various ways in which an agent may be related to the thing the agent is involved in. The pattern also allows us to express the starting and ending time of a role.

The abstract pattern **Event** describes generic events, which may include cruises, sampling processes, etc. This pattern is inspired by the Simple Event Model (SEM) [4], but augmented with a stronger axiomatization in OWL. In this model, an event is something that occurs at some place and some time, and may provide agent roles performed by agents.

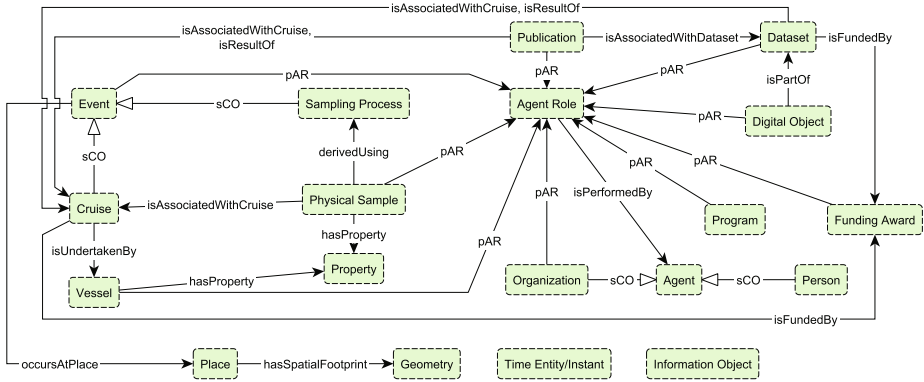


Fig. 1. Schema diagram containing (almost) all patterns in the GeoLink ontology and their main links. All patterns have links to Time Entity/Instant and Information Object, but they are not displayed to make the figure less cluttered. sCO=subClassOf; pAR=providesAgentRole; each box is a pattern, represented by its main class.

The abstract pattern **Information Object**, reused from DOLCE [8], encapsulates information commonly attributed to an object, including name, aliases, description, webpage, and other non-URI identifiers. This pattern allows us to collect many pieces of information about an object that become relevant when it is understood as an information artifact.

The **Place** pattern captures spatial information in the Event pattern above and the rest of the ontology. It expresses that a place has a geometry as its spatial footprint, similar to the relationship between geographic feature and geometry in GeoSPARQL [9]. The Information Object pattern is used to represent other information about a place such as its name and description.

The **Person** pattern is a specialization of the Agent pattern, describing human persons. As an Agent, a person may perform a role in a particular context. Additionally, the Person pattern allows one to say that a person has personal information items. A personal information item is an attribute of a person, such as name, address, etc., that may change during his/her lifetime, and is modeled through the auxiliary **Personal Info Item** pattern. The **Person Name** pattern is also defined as a specialization of the Personal Info Item pattern.

The **Organization** pattern is a specialization of the Agent pattern, describing organizations, including academic institutions, funding agencies, vessel owners, etc. This pattern also models affiliation relationships of an agent to organizations using the Agent Role pattern. Every organization is described by exactly one information object that encapsulates additional information about the organization such as name and location. This last part is modeled by reusing and aligning with the Information Object pattern.

The **Cruise** pattern describes oceanographic cruises. A cruise is modeled as a type of event whose spatiotemporal component is determined by its trajectory. The Agent Role pattern is used to model various roles a person or organization

may hold in relation to a cruise. In addition, the **Vessel** pattern models vessels, which is the physical object with which the cruise is undertaken.

The **Funding Award** pattern describes funding awards given to researchers to carry out their ocean science research activities. It has a starting and ending date and provides roles to agents such as principal investigator, sponsor, etc.

The **Program** pattern captures the notion of ocean science programs. A program is a loose group of activities, funding awards, and other things related to ocean science research that follow certain scientific themes or objectives. It can be in the form of a strategic initiative spanning different projects, or a collaborative network involving many scientists working on different projects which share some common strategic goals.

The **Dataset** pattern models the notion of dataset and common metadata such as description, creator, creation time, etc. A dataset can be associated with features of interests and may contain digital objects. The **Digital Object** pattern represents digital objects, which are understood as file-like objects within a data repository.

The **Physical Sample** pattern minimally represents discrete specimens (rocks, sediments, fluids, etc) collected from the natural environment for scientific study.

4 Selected Modeling Details

In this section, we present selected modeling details from the Cruise pattern, which is arguably one of the most interesting parts of the GeoLink ontology. Oceanographic cruises indeed play a very central role in the professional lives of many ocean scientists, and so it is very natural to utilize them as an aspect of data organization and sharing. In our ontology, an oceanographic cruise is modeled as a type of event whose spatiotemporal component is represented by its trajectory. Like in the Event pattern (Fig. 2), the Cruise pattern employs the Agent Role pattern to model involvement of agents in it (Fig. 3). Alignment of Cruise pattern to the Event pattern is specified through axioms written in description logic (DL) notation in (1)-(5) with `glev:` denoting the namespace prefix of the Event pattern. Specific roles for cruise are defined by subclassing `AgentRole`.

$$\text{Cruise} \sqsubseteq \text{glev:Event}, \text{Port} \sqsubseteq \text{glev:Place}, \text{TimeEntity} \sqsubseteq \text{glev:TimeEntity} \quad (1)$$

$$\text{AgentRole} \sqsubseteq \text{glev:AgentRole}, \text{Agent} \sqsubseteq \text{glev:Agent} \quad (2)$$

$$\text{hasTrajectory} \circ \text{hasFix} \circ \text{hasLocation} \circ \text{hasSpatialFootprint}^- \sqsubseteq \text{glev:occursAtPlace} \quad (3)$$

$$\text{hasTrajectory} \circ \text{hasFix} \circ \text{atTime} \sqsubseteq \text{glev:occursAtTime} \quad (4)$$

$$\text{providesAgentRole} \sqsubseteq \text{glev:providesAgentRole}, \text{isPerformedBy} \sqsubseteq \text{glev:isPerformedBy} \quad (5)$$

We then assert that a cruise has exactly one trajectory, is undertaken by exactly one vessel, and is described by exactly one `InformationObject`. Additionally, this instance of `InformationObject` describes exactly only the cruise. This `InformationObject`, which is aligned to the class of the same name from the

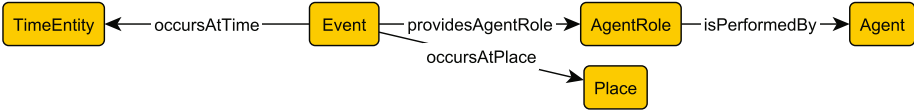


Fig. 2. Event pattern

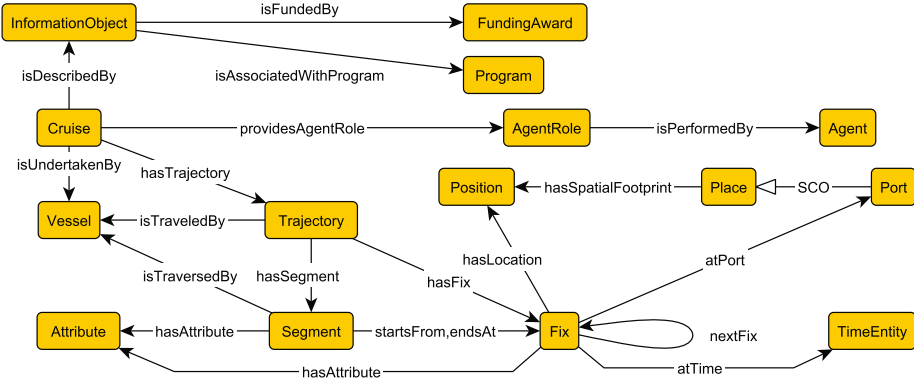


Fig. 3. Cruise with Trajectory and Agent Roles. SCO=subClassOf

Information Object pattern, acts as a proxy through which we describe various information about the cruise not covered by having the cruise as an event. In addition to the data properties (not displayed in the figure) inherited from the Information Object pattern such as identifier, description, webpage, etc., an InformationObject of a cruise may carry information regarding funding award, program, as well as other cruises related to the cruise described by this instance of InformationObject. We also assert that if a cruise is undertaken by a vessel, then this vessel has to travel along the cruise’s trajectory.

$$\text{Cruise} \sqsubseteq (=1 \text{ hasTrajectory.Trajectory}) \sqcap (=1 \text{ isUndertakenBy.Vessel}) \sqcap (=1 \text{ isDescribedBy.InformationObject}) \tag{6}$$

$$\text{InformationObject} \sqsubseteq (=1 \text{ isDescribedBy}^- .\text{Cruise}) \tag{7}$$

$$\text{hasTrajectory}^- \circ \text{isUndertakenBy} \sqsubseteq \text{isTraveledBy} \tag{8}$$

A cruise trajectory in turn represents a route that the cruise takes. To model the notion of cruise trajectory, we reused and extended the Semantic Trajectory pattern, which already provides basic vocabulary and OWL axiomatization [6]. Generally, a trajectory is given by an ordered collection of *fixes*, representing time-stamped locations. Non-spatiotemporal information specific to a fix can be included via its *attributes*, for example, to indicate that the fix is the arrival to some port stop. Between two consecutive fixes, we define a *segment*, traversed by some moving object, e.g., a vessel. Details of the axiomatization for trajectory is given in the technical report and in the OWL implementation.

One other piece of detail we would like to convey is the use of guarded domain and range restrictions. Most of the arrows in Figures 2 and 3 represent object properties and the direction of the arrows goes from the domain part of the property towards its range. A straightforward way to axiomatize these would have been as unguarded domain and range restrictions, i.e., as axioms of the form $P \text{ rdfs:domain } C$ and $P \text{ rdfs:range } C$, which are equivalent to $\exists P.T \sqsubseteq C$ and $T \sqsubseteq \forall P.C$ using DL notation. For patterns, however, this would introduce rather strong ontological commitments which would make future reuse of the patterns more difficult. Hence, we use guarded versions of the restrictions, e.g., for the `hasTrajectory` property, we state the two axioms $\exists \text{hasTrajectory.Trajectory} \sqsubseteq \text{Cruise}$ and $\text{Cruise} \sqsubseteq \forall \text{hasTrajectory.Trajectory}$.

5 Availability

All ODPs for GeoLink are available online from <http://schema.geolink.org/>, including the technical report containing detailed descriptions of all patterns. This will be made available even beyond the duration of the GeoLink project because the participating repositories have committed to continue the integration effort, which is also strongly motivated by ocean science researchers who have been using these repositories for their research activities.

Each pattern resides in its own OWL file with ontology URI of the form [http://schema.geolink.org/version/pattern/\[patternname\]](http://schema.geolink.org/version/pattern/[patternname]). For instance, <http://schema.geolink.org/dev/pattern/agentrole> is the URI where the Agent Role pattern currently resides. The `dev` part in the URI indicates that the pattern is currently under development. A stable release will replace `dev` with a version number. A pattern is typically aligned to another pattern or an external ontology, and this is incorporated by creating a separate OWL file containing axioms for one direction of alignment. For example, the alignment from the Cruise pattern to the Event pattern is provided by the module at <http://schema.geolink.org/dev/pattern/cruise-to-event>, which really contains axioms (1)-(5). Meanwhile, an alignment to the W3C Time ontology³ is provided by the module at <http://schema.geolink.org/dev/pattern/cruise-to-owltime>.

Note that such alignment modules specify one-direction alignment as they do not specify an alignment from the opposite direction. For example, the module at <http://schema.geolink.org/dev/pattern/cruise-to-event> really only specifies an alignment from Cruise to Event, and not from Event to Cruise. In terms of file organization, such alignment modules are imported by the pattern that is the origin of the alignment. So, the module at <http://schema.geolink.org/dev/pattern/cruise-to-event> is imported by the Cruise pattern at <http://schema.geolink.org/dev/pattern/cruise>.

Besides the W3C Time ontology, external linkages also exist to other ontologies and vocabularies. The Place pattern is aligned to the GeoSPARQL ontology,⁴ in particular the Geometry class. Standard geographic features from the

³ <http://www.w3.org/2006/time>

⁴ <http://www.opengis.net/ont/geosparql>

GEBCO gazetteers⁵ are used to populate the Place pattern. In addition, we adopt parts of the SeaVoX standard platform types⁶ to obtain types of vessels.

6 Conclusions

We have presented the GeoLink ontology, consisting of a number of ODPs designed for the oceanography domain. The resulting patterns are sufficiently modular, and thus arguably easier to extend than foundational, top-level ontologies. Currently, the GeoLink project is in the middle of populating the patterns with real data and a very preliminary evaluation demonstrated that the patterns together can serve as an integrating layer of heterogeneous oceanographic data repositories.

Acknowledgments. The presented work has been primarily funded by the National Science Foundation under the award 1440202 “EarthCube Building Blocks: Collaborative Proposal: GeoLink – Leveraging Semantics and Linked Data for Data Sharing and Discovery in the Geosciences.”

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. *International Journal on Semantic Web and Information Systems* **5**(3), 1–22 (2009)
2. Gangemi, A.: Ontology design patterns for semantic web content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) *ISWC 2005*. LNCS, vol. 3729, pp. 262–276. Springer, Heidelberg (2005)
3. Hackett, E.J., et al.: Ecology transformed: NCEAS and changing patterns of ecological research. In: Olson, G.M., et al. (eds.) *Science on the Internet*, pp. 277–296. MIT Press, Cambridge (2008)
4. van Hage, W.R., et al.: Design and use of the Simple Event Model (SEM). *Journal of Web Semantics* **9**(2), 128–136 (2011)
5. Hampton, S.E., Parker, J.N.: Collaboration and productivity in scientific synthesis. *BioScience* **61**(11), 900–910 (2011)
6. Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In: Tenbrink, T., Stell, J., Galton, A., Wood, Z. (eds.) *COSIT 2013*. LNCS, vol. 8116, pp. 438–456. Springer, Heidelberg (2013)
7. Kalbasi, R., et al.: Collaborative ontology development for the geosciences. *Transactions in GIS* **18**(6), 834–851 (2014)
8. Oberle, D., et al.: DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *Journal of Web Semantics* **5**(3), 156–174 (2007)
9. Open Geospatial Consortium: OGC GeoSPARQL - a geographic query language for RDF data. Open Geospatial Consortium (2011). Document 11–052r4

⁵ http://www.gebco.net/data_and_products/undersea_feature_names/

⁶ <http://vocab.nerc.ac.uk/collection/L06/current/>

10. Presutti, V., et al.: eXtreme Design with content ontology design patterns. In: Blomqvist, E., et al. (eds.) Proceedings of the Workshop on Ontology Patterns (WOP 2009), Washington, D.C., USA, October 25, 2009. CEUR Workshop Proceedings, vol. 516. CEUR-WS.org (2009)
11. Reichman, O., Jones, M.B., Schildhauer, M.P.: Challenges and opportunities of open data in ecology. *Science* **331**(6018) (2011)
12. You, J.: Geoscientists aim to magnify specialized web searching. *Science* **347**(6217), 11–11 (2015)

Semantic Bridges for Biodiversity Sciences

Natalia Villanueva-Rosales^{1,2(✉)}, Nicholas del Rio³, Deana Pennington^{1,4},
and Luis Garnica Chavira^{1,2}

¹ Cyber-ShARE Center of Excellence, University of Texas at El Paso, El Paso, USA
{nvillanuevarosales, ddpenninngton, lagarnicachavira}@utep.edu

² Department of Computer Science, University of Texas at El Paso, El Paso, USA

³ Air Force Research Lab, Information Directorate, Rome, USA
nicholas.del_rio@us.af.mil

⁴ Department of Geology, University of Texas at El Paso, El Paso, USA

Abstract. Understanding the impact of climate change and humans on biodiversity requires the retrieval and integration of heterogeneous data sets for the generation of models that provide insights not possible with a single model. Scientists invest a significant amount of time collecting and manually pre-processing data for the generation of such models. The Earth Life and Semantic Web (ELSEWeb) project aims to create a semantic-based, open-source cyberinfrastructure to automate the ingestion of data by models. This paper describes the ontologies at the backbone of ELSEWeb that provide semantic bridges between environmental data sources and species distribution models.

Keywords: Ontology · Data-to-model integration · Model web · Biodiversity · Climate change

1 Introduction

What will happen to native species in national parks under scenarios of climate change? When and where might we expect zoonotic infectious disease to spread? These questions and others can be addressed using species distribution models (SDMs) [1]. SDMs predict where animal or plant species might find suitable habitat given present conditions or under change scenarios. Species might be socially relevant because they are endangered or carry diseases. Conducting “what-if” analyses provides insights of changes in the environment as they occur – or before they occur. Scenario analysis is becoming a key tool for the biodiversity (and other) sciences [2] to understand human impacts on the environment coupled with climate change. SDMs require species occurrence data and environmental data. Species occurrence data contains the location of known occurrences of a species in a given time period. Species occurrence data is mostly available in museums, which have invested in digitization and the development of metadata standards and repositories through the Global Biodiversity Information Facility (GBIF)¹. Environmental data is heterogeneous and

N. del Rio—Affiliated with the University of Texas at El Paso when producing this work.

¹ <http://www.gbif.org>

© Springer International Publishing Switzerland 2015

M. Arenas et al. (Eds.): ISWC 2015, Part II, LNCS 9367, pp. 310–317, 2015.

DOI: 10.1007/978-3-319-25010-6_20

available from multiple sources, e.g., satellite imagery. Scientists spend considerable time deciding what data might be most relevant, where to find it, how to obtain it, and what to do with it since data manipulation may require the use of proprietary tools. In addition, each modelling algorithm has its own constraints, operational requirements, assumptions, parameter and data requirements, usage history, and advocates (or dissidents). Data are required to be further manipulated and assembled into the formats and scales consistent with the selected algorithm. The GEO Model Web initiative [3] aims to increase access to models and interoperability across models, databases, and websites. The Model Web advocates the creation of infrastructure with four underlying principles: open access, minimal barriers to entry, service-driven, and scalability. The Earth, Life and Semantic Web (ELSEWeb) project aims to enable interoperability between data and model providers in a way that data can be automatically retrieved and ingested by a model seamlessly to the user. ELSEWeb follows the four principles of the Model Web. The current implementation of ELSEWeb integrates data sets from the University of New Mexico Earth Data Analysis Center (EDAC)² with the species distribution modelling service provider Lifemapper (Lifemapper)³.

2 Semantic Descriptions of Data and Model Providers

Ontologies in ELSEWeb describe concepts needed to automate the retrieval and manipulation of data for the generation of SDMs, to advance our understanding of how to automatically chain models together – a requirement of the Model Web. These ontologies were created using an iterative, bottom-up approach driven by standards and best practices followed by EDAC and Lifemapper, such as those provided by the Open Geospatial Consortium (OGC)⁴ and the Federal Geographic Data Committee (FGDC)⁵. Non-semantic resources were inspected, interpreted, and expressed as classes and properties using the Web Ontology Language (OWL)⁶. These resources included Web Services' metadata, arbitrary REST-full service descriptions in XML, shell scripts and Java code. ELSEWeb's initial semantic descriptions focused on technical requirements (e.g., format) to enable the seamless integration of data and models from EDAC and Lifemapper respectively [4]. These ontologies were iteratively refined with expertise knowledge from EDAC and Lifemapper partners and aligned to upper-level, and community accepted ontologies and vocabularies such as the Provenance Ontology (PROV-O)⁷. In this paper, ELSEWeb's ontology names, classes and properties are denoted in *italics*.

² <http://edac.unm.edu>

³ <http://lifemapper.org/>

⁴ <http://www.opengeospatial.org/standards>

⁵ <https://www.fgdc.gov/>

⁶ <http://www.w3.org/2002/07/owl#>

⁷ <http://www.w3.org/ns/prov-o>

2.1 ELSEWeb’s Environmental Data Ontology

The *elseweb-data* ontology currently provides concepts to describe datasets with characteristics relevant to SDMs such as spatial/temporal dimensions. The *elseweb-data* ontology covers remote sensing environmental data, spatial in nature, geo-referenced, and measured from some instrument such as a MODIS sensor⁸. Environmental data represent any phenomenon that might be important for providing required habitat for a given species or for constraining the ability of a species to survive such as vegetation type. These biophysical data are usually combined with climate data that often establish thresholds of survival. Fig. 1 illustrates the core classes of the *elseweb-data* ontology with gray nodes and the ontologies it extends. The *elseweb-data* ontology extends the Data Catalog Vocabulary DCAT⁹ to describe the temporal and spatial coverage of data (e.g., *Geographic Region*), the corresponding *Theme* (e.g., *Vegetation*), and how the data can be accessed (e.g. a *File Manifestation*, subclass of *Dataset Manifestation*, describes the format of a file and where it can be downloaded).

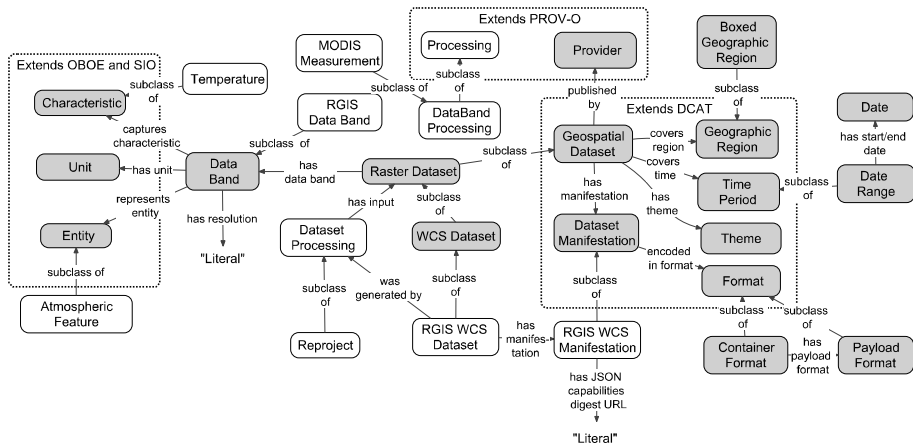


Fig. 1. The *elseweb-data* ontology (gray nodes) provides concepts for describing characteristics of a dataset in a biodiversity scenario: spatial/temporal dimensions, how it can be accessed and the focal entity described. The *elseweb-edac* ontology (white nodes) extends and instantiates the *elseweb-data* ontology to describe data sets published by EDAC. Dashed boxes outline concepts of the ontology that extend existing ontologies and vocabularies.

With respect to spatial coverage, ELSEWeb describes a *Boxed Geographic Region* specified by left longitude, lower latitude, right longitude and upper latitude. This notion follows OGC standards and best practices for Web Services. More generally, OGC Web Coverage Services (WCS) metadata describes how to invoke services for data retrieval, their corresponding spatial/temporal coverage, and the different kinds of formats available for the retrieved file (e.g., PNG or TIFF). Additionally, the OGC metadata schema defines extension points which can be used to reference additional

⁸ <http://modis.gsfc.nasa.gov/>

⁹ <http://www.w3.org/ns/dcat>

metadata. ELSEWeb's data provider, EDAC, relies on these extension points to include FGDC metadata in their services. The design of the *elseweb-data* ontology was largely inspired by this conglomerate-service metadata employed by EDAC. The most specific concepts of *elseweb-data* ontology describe different kinds of Geospatial data sets. For example, a *Raster Dataset* consists of a set of *DataBands*. A *DataBand* captures a *Characteristic* (e.g., *Temperature*) of an *Entity* (e.g., *Air*), with a specific *Unit* (e.g., Fahrenheit) and *Resolution* (e.g. 250 m). By extending the Extensible Observation Ontology (OBOE)¹⁰, *elseweb-data* can answer questions about measurements contained in specific *DataBands* or the observed focal entity. By extending the PROV-O, *elseweb-data* can answer questions about data providers and how the data was generated.

The *elseweb-data* ontology is extended and populated with data provided by EDAC in the *elseweb-edac* ontology, illustrated by white nodes in Fig. 1. For example, information about data sets published by EDAC instantiates the class *RGIS Data Band*, a subclass of the generic *Data Band*. Fig. 1 illustrates that EDAC also offers data processing services, such as *Reproject*, used by EDAC when publishing a *Raster* dataset. Provenance information indicates, for example, that EDAC's data is retrieved from the US National Aeronautics and Space Administration (NASA) and transformed following EDAC's quality assurance guidelines. Once the dataset is published as a service, it becomes an *RGIS WCS Service* that has a corresponding *RGIS WCS Manifestation* with information on how to retrieve the data. Appropriate classes and properties of *elseweb-edac* extend PROV-O classes such as PROV-O Activity. EDAC's geospatial data is exposed through OGC Web Map, Web Feature and Web Coverage Services. EDAC's Web Coverage Services' metadata is leveraged by ELSEWeb's harvester to automatically populate *elseweb-edac*.

2.2 ELSEWeb's Modelling Service Ontology

The *elseweb-modelling* ontology, also in OWL, describes biodiversity modelling services as implemented algorithms, parameters, and data inputs/outputs. Fig. 2 illustrates the core classes of this ontology with gray nodes. Concepts in *elseweb-modelling* that are used to describe a modelling algorithm (e.g. *Species Modelling Algorithm*) and corresponding parameters (e.g., *Species Modelling Parameter*) extend the Semantic Science Integrated Ontology (SIO)¹¹. Concepts describing services, inputs, and outputs are inherited from the Semantic Automated Discovery and Integration (SADI) framework¹² and PROV-O. For example, in the *elseweb-modelling* ontology, a SADI Service Agent is also a PROV-O Agent. This model allows ELSEWeb to describe a computational process with an *Activity* associated with a SADI *Service*, which is also a PROV-O Agent. The combination of PROV-O and SADI provides a unified view to describe discovery, run time, and provenance metadata.

¹⁰ <http://ecoinformatics.org/oboe/oboe.1.0/oboe-core.owl>

¹¹ <http://semanticscience.org/ontology/sio.owl>

¹² <http://sadiframework.org/>

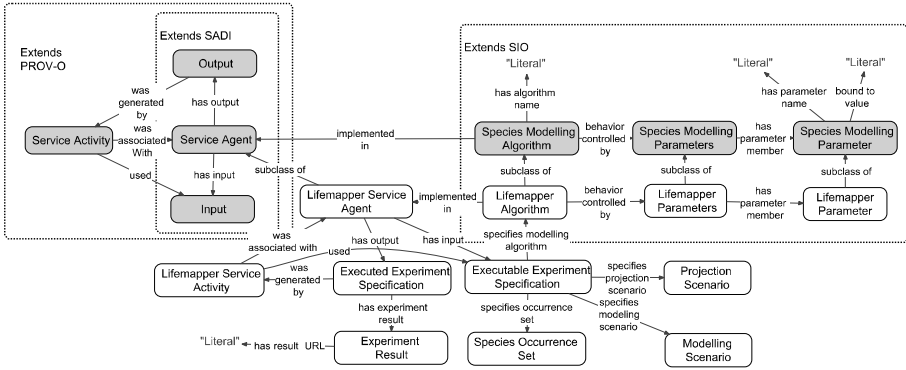


Fig. 2. The *elseweb-modelling* ontology (gray nodes) provides concepts for describing biodiversity modelling algorithms, their inputs, outputs, parameters, and services providing these algorithms. The *elseweb-lifemapper* ontology (white nodes) extends and instantiates the *elseweb-model* ontology to describe SDM services provided by Lifemapper. Dashed boxes outline concepts of the ontology that extend existing ontologies.

The *elseweb-lifemapper* ontology, illustrated by white nodes in Fig. 2, extends and instantiates the *elseweb-modelling* ontology with information about the modelling services provided by Lifemapper. *Elseweb-lifemapper* describes the specific inputs and outputs generated by a modelling service. For example, a *Lifemapper Service Agent* requires an *Executable Experiment Specification* as an input. An *Executable Experiment Specification* is composed of *Species Occurrence Set*, *Modelling Scenario* (e.g., a stack of environmental data used to generate an SDM), *Projection Scenario* (e.g. a stack of environmental data used to generate a projection of the species distribution by applying the SDM to changed conditions), and an *Algorithm Specification* (e.g., *Maximum Entropy*).

The *elseweb-lifemapper* ontology contains concepts that describe how a projection was generated. ELSEWeb provides users with provenance to understand how their specific modelling and projection scenarios were used to create a specific SDM. Lifemapper’s Web Services expose XML files with metadata about available algorithms and their parameters. The *elseweb-lifemapper* ontology is automatically populated using a Java-based harvester that leverages this metadata.

2.3 Semantic Bridges from Data to Model Providers

Scientists often write custom scripts in order to move, augment, and transform source data into alternative forms that suit the needs of target analytical tools [5]. These custom scripts can often lack explicit documentation about data ingestion, transformation processes, and generated outputs, making these scripts difficult to reuse. To facilitate reuse of data transformation services, ELSEWeb creates semantic bridges (i.e., alignments) between the *elseweb-edac* ontology and the *elseweb-lifemapper* ontology. These bridges, specified in the *elseweb-mappings* ontology, provide a declarative, formal specification that can automate the data transformations required to generate an SDM. The *elseweb-mappings* ontology was specified by ELSEWeb team using an

iterative and bottom-up, two-stage process. First, an ad-hoc workflow composed of shell script and Java widgets that performed the necessary transformations was created [4]. Then, this workflow was promoted from the procedural program level to the declarative level using OWL formalisms. For example, an *RGIS WCS Dataset* published by EDAC can be automatically classified as part of an *Executable Experiment Specification*, which in turn is described as the input of a *Lifemapper Service Agent*.

3 Results

ELSEWeb's ontologies are stored in an instance of Virtuoso triple store¹³ with over 350,000 triples and available through an SPARQL endpoint¹⁴. Additional resources can be found at the project's website¹⁵. The linked data section of the website includes SPARQL sample queries to navigate ELSEWeb's knowledge base and answer competency questions. The ontology section provides links to ontology files and additional diagrams. Ontologies can also be retrieved under their designated namespace¹⁶. ELSEWeb currently enables the integration of over 6,650 environmental data sets, 1000 species occurrence data sets and 11 algorithms for the generation of SDMs.

Users can generate SDMs by creating experiments through the Graphical User Interface (GUI)¹⁷. Interested readers can test ELSEWeb by accessing the demo section. After submitting an experiment, SDMs can be retrieved at Lifemapper with the credentials user:elseweb2, password:elsewebtwo. ELSEWeb's GUI also allows the manual submission of experiments through a JSON specification. The service-oriented architecture used in ELSEWeb's infrastructure, GUI design, and a discussion of the technical challenges addressed by the ELSEWeb framework can be found in [6].

Existing semantic web service frameworks such as SADI can leverage semantic bridges and orchestrate the execution of Web Services to dynamically generate the output required. In ELSEWeb, SADI services are used to retrieve and transform data and generate SDMs along with a provenance trace. SADI services semantically describe EDAC's Web Services to retrieve data requested by the user as a measured characteristic with time and spatial constraints. SADI services semantically describe the inputs and outputs of services at Lifemapper that generate SDMs. ELSEWeb uses the SHARE client¹⁸ provided by the SADI framework, to automatically orchestrate the execution of SADI services. Interested readers may refer to [4] for an in-depth description of ELSEWeb's SADI services and the service orchestration process.

¹³ <http://www.openlinksw.com/>

¹⁴ <http://visko.cybershare.utep.edu/sparql>

¹⁵ <http://elseweb.cybershare.utep.edu/>

¹⁶ <http://ontology.cybershare.utep.edu/ELSEWeb/>

¹⁷ <http://elseweb.cybershare.utep.edu/experiments>

¹⁸ <http://biordf.net/cardioSHARE/>

4 Related Work

Efforts towards the integration of heterogeneous data for the creation of biodiversity models include eHabitat [7] and iPlant Collaborative¹⁹. eHabitat implements the Model Web principles using OGC Web Processing Services for multi-purpose modeling, including ecological forecasting under climatic or development scenarios. To the best of our knowledge, eHabitat does not make use of ontologies or generic reasoners for service orchestration. Instead, eHabitat provides web clients where users can manually orchestrate service execution. The iPlant Semantic Web Platform enables the semantic discovery of services and service orchestration using the Simple Semantic Web Architecture Protocol (SSWAP) and Resource Description Graphs [8]. In contrast to ELSEWeb, service orchestration in iPlant is a manual task, facilitated by a GUI where a reasoner leverages service descriptions, encoded in ontologies, to suggest the next service in the pipeline. iPlant Collaborative efforts also include the development of ontologies to support biodiversity knowledge discovery [9]. These ontologies provide complementary concepts that can be used to further inform the models generated at ELSEWeb. In particular, the Biological Collections Ontology (BCO) describes specimen collections and sampling processes, the Environmental Ontology (ENVO) describes habitats, environmental features and materials, and the Population and Community Ontology (PCO) describes communities of biological entities and their qualities and interactions, among other concepts.

5 Conclusions and Future Work

Easier and more efficient approaches for harnessing the vast array of scientific data are essential for the generation of biodiversity models. The ontologies developed in ELSEWeb aim to facilitate the use of data and data transformation and modelling services following the principles of the Model Web. ELSEWeb's ontologies were created by an interdisciplinary group of researchers and developers following an iterative, bottom-up approach driven by community standards and best practices. Semantically described data and models exposed on the Semantic Web enable frameworks like SADI to automatically find, retrieve, and manipulate data to generate SDMs. These capabilities, however, do not address the issues involved in determining whether data and model integration is sensible scientifically even if it can be accomplished technically. In part, this is due to the sheer enormity of the task, given the wide range and diversity of concepts employed in any given biodiversity problem. Related efforts towards creating ontologies for biodiversity knowledge discovery include the development of BCO, ENVO and PCO. Aligning ELSEWeb's ontologies with such ontologies may lead to a more robust integration of domain and technical concepts, enable the automated verification of models, and provide a more comprehensive provenance trace. Future work of ELSEWeb includes the integration of additional data sets and services for the generation of water models. Water models will provide the opportunity to test ELSEWeb with members of academia, industry, society and government to facilitate interdisciplinary collaborations.

¹⁹ <http://www.iplantcollaborative.org/>

Acknowledgements. ELSEWeb was funded by NASA ACCESS grants NNX12AF49A (UTEP), NNX12AF52A (UNM), and NNX12AF45A (KU). This work used resources from Cyber-ShARE Center of Excellence supported by NSF grant HRD-0734825 and HRD-1242122. The authors thank Soren Scott and Karl Benedict (EDAC), and CJ Grady and Aimee Stewart (Lifemapper) for their contributions to the design of ELSEWeb's ontologies and the anonymous reviewers for their insightful comments on this manuscript.

References

1. Krause, C., Pennington, D.: Strategic decisions in conservation: using species distribution modelling to match ecological requirements to available habitat. In: Maschinski, J., Haskins, K. (eds.) *Plant Reintroduction in a Changing Climate: Promises and Perils*, p. 432. Island Press, Washington (2012)
2. Settele, J., Carter, T.R., Kühn, I., Spangenberg, J.H., Sykes, M.T.: Scenarios as a tool for large-scale ecological research: experiences and legacy of the ALARM project. *Glob. Ecol. Biogeogr.* **21**, 1–4 (2012)
3. Nativi, S., Mazzetti, P., Geller, G.N.: Environmental model access and interoperability: The GEO Model Web initiative. *Environ. Model Softw.* **39**, 214–228 (2013)
4. Del Rio, N., Villanueva-Rosales, N., Pennington, D., Benedict, K., Stewart, A., Grady, C.: ELSEWeb meets SADI: Supporting data-to-model integration for biodiversity forecasting. In: *AAAI Fall Symposium on Discovery Informatics* (2013)
5. Kandel, S., Paepcke, A., Hellerstein, J.M., Heer, J.: Enterprise data analysis and visualization: An interview study. *IEEE Trans. On. Vis. Comput. Graph.* **18**, 2917–2926 (2012)
6. Villanueva-Rosales, N., Garnica Chavira, L., Del Rio, N., Pennington, D.: eScience through the integration of data and models: a biodiversity scenario. In: *11th IEEE International Conference on eScience* (2015)
7. Dubois, G., Schulz, M., Skøien, J., Bastin, L., Peedell, S.: eHabitat, a multi-purpose Web Processing Service for ecological modelling. *Environ. Model Softw.* **41**, 123–133 (2013)
8. Gessler, D.D., Bulka, B., Sirin, E., Vasquez-Gross, H., Yu, J., Wegrzyn, J.: iPlant SSWAP (Simple Semantic Web Architecture and Protocol) enables semantic pipelines for biodiversity. *Semant. Biodivers. S4BioDiv 2013* **101** (2013)
9. Walls, R.L., Deck, J., Guralnick, R., Baskauf, S., Beaman, R., Blum, S., Bowers, S., Buttigieg, P.L., Davies, N., Endresen, D., Gandolfo, M.A., Hanner, R., Janning, A., Krishtalka, L., Matsunaga, A., Midford, P., Morrison, N., Tuama, É.Ó., Schildhauer, M., Smith, B., Stucky, B.J., Thomer, A., Wiczorek, J., Whitacre, J., Wooley, J.: Semantics in Support of Biodiversity Knowledge Discovery: An Introduction to the Biological Collections Ontology and Related Ontologies. *PLoS ONE* **9**, e89606 (2014)

IoT and Sensors

FraPPE: A Vocabulary to Represent Heterogeneous Spatio-temporal Data to Support Visual Analytics

Marco Balduini^(✉) and Emanuele Della Valle

DEIB, Politecnico of Milano, Milano, Italy
{marco.balduini,emanuele.dellavalle}@polimi.it

Abstract. Nowadays, we are witnessing a rapid increase of spatio-temporal data that permeates different aspects of our everyday life such as mobile geolocation services and geo-located weather sensors. This big amount of data needs innovative analytics techniques to ease correlation and comparison operations. Visual Analytics is often advocated as a doable solution thanks to its ability to enable users to directly obtain insights that support the understanding of the data. However, the grand challenge is to offer to visual analytics software an integrated view on top of multi-source, geo-located, time-varying data. The abstractions described in the FraPPE ontology address this challenge by exploiting classical image processing concepts (i.e. Pixel and Frame), a consolidated geographical data model (i.e. GeoSparql) and a time/event vocabulary (i.e. Time and Event ontologies). FraPPE was originally developed to represent telecommunication and social media data in an unified way and it is evaluated modeling the dataset made available by ACM DEBS 2015 Grand Challenge.

1 Introduction

Nowadays, we are witnessing a rapid increase of sources exposing geo-located time-varying data such as social media, mobile telecommunication, taxis, etc. Making sense of those data sets typically requires to compare and correlate them. Visual Analytics – the science of analytical reasoning facilitated by interactive visual interfaces [12] – is often advocated as an effective solution for those tasks.

For instance, Figure 1 illustrates a real case of visual analytics for a general audience¹ where: a grid of 6x3 cells is overlaid to a city street map, green circles represent the number of tweets posted in a time interval from each cell, and the fill colour opacity value of each cell is mapped to the number of mobile calls from each cell. As shown in [2], people without specific expertise in data analytic can easily spot the cells where the two signals are correlated.

However, data is not often ready for visual analytics. Usually, geo-located time-varying data of this type has first to be aggregated over time and space.

¹ Interested readers are invited to view <https://youtu.be/MOBie09NHxM>

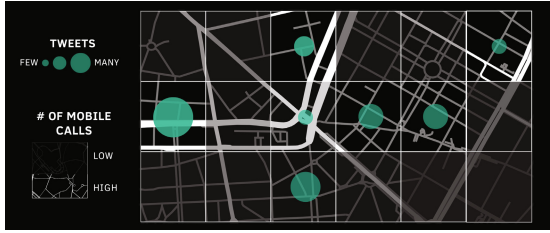


Fig. 1. A real-world example of visual analytics of two heterogeneous datasets.

FraPPE is a vocabulary designed exactly for this purpose following Methontology [6] guidelines. It exploits classical image processing concepts (FRAME and Pixel) – familiar to designers of visual analytics solutions – as well as common sense concepts (Place and Event) using a consolidated geographical data model (GeoSparql [3]) and a time/event vocabulary (Time [8] and Event ontologies [9]).

The basic building blocks of FraPPE were originally developed to represent, in an homogeneous way, heterogeneous data streams for the CitySensing² installation proposed to the public of Milano Design Week 2014. Some of its concepts (i.e., dividing the physical space in cells using a grid and linking time-varying data to cells) were used to publish the dataset of Telecom Italia Big Data Challenge 2014³. In this paper, we present the first formal version of this vocabulary (Section 2) and we evaluate it (Section 3) by assessing its compliance to Tom Gruber’s principles (i.e., clarity, coherence, minimal encoding bias, minimal ontological commitment, extendibility) modelling the dataset of ACM DEBS 2015 Grand Challenge⁴.

FraPPE vocabulary is published in Linked Open Vocabularies⁵ while community discussion, issue tracking and advancement are handled via github⁶. Those resources are maintained and sustained on the long term by the Stream Reasoning research group of Politecnico di Milano. The data of ACM DEBS 2015 Grand Challenge modelled in FraPPE can be queried using a SPARQL endpoint⁷ whose content is described with VoID [1] machine processable metadata⁸.

2 FraPPE

The overall idea of FraPPE is depicted in Figure 2. A portion of the physical world is illustrated using a map in the top-right side of the figure. A GRID, which, in this example, is made of 4 CELLS, sits between the physical world and the film that CAPTURES a FRAME per time-interval. The frame being captured

² <http://citysensing.fuorisalone.it/>

³ <https://dandelion.eu/datamine/open-big-data/>

⁴ <http://www.debs2015.org/call-grand-challenge.html>

⁵ <http://lov.okfn.org/dataset/lov/vocabs/frappe>

⁶ <https://github.com/streamreasoning/FraPPE>

⁷ <http://www.streamreasoning.com/datasets/debs2015/>

⁸ <http://streamreasoning.org/datasets/debs2015/voidrdf>

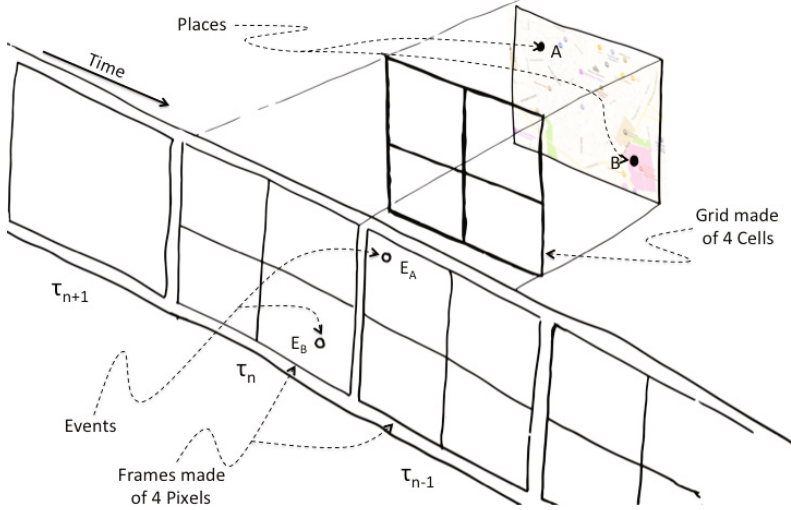


Fig. 2. An high-level view of FraPPE including 3 FRAMES made of 4 Pixels containing Places where Events happens.

in the current time interval τ_n is directly in front of the grid. The previous frame, captured at time τ_{n-1} , is on its right. The next empty frame, which will be captured at time τ_{n+1} is on the left. The two captured frames both have 4 PIXELS (one for each cell). The physical world contains two PLACES (e.g. A the start and B the end points of a journey). The frame captured at τ_{n+1} contains a pixel that accounts for the EVENT E_A occurred in A at τ_{n+1} (e.g., the pick up of some good). In a similar manner, the just captured frame accounts for the event E_B occurred in B at τ_n (e.g., the drop-off of some good).

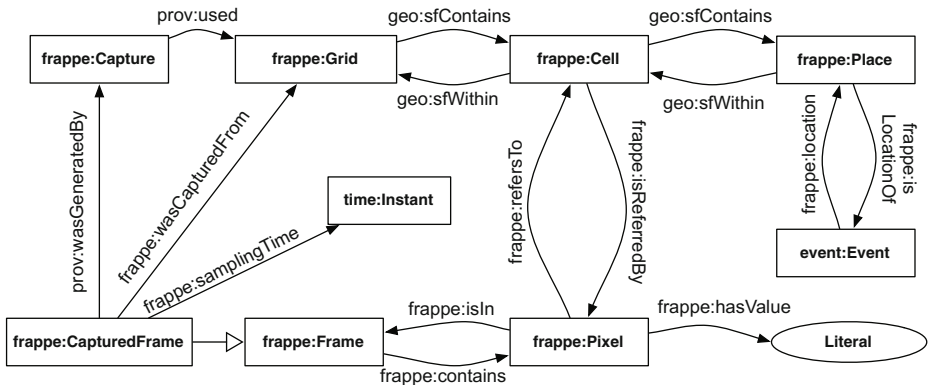


Fig. 3. The core terms of FraPPE vocabulary, which reuse the geosparql vocabulary, the PROV Ontology and the Event Ontology.

More formally, FraPPE ontology is organised in three interconnected parts: the geographical part, the time-varying one and the provenance one (see Figure 3). PLACE, CELL and GRID belong to the geographical part and reuse geosparql vocabulary [3]. They are geosparql **Features** whose default geometry are respectively a **point**, a **surface** and a **multisurface**. EVENT, PIXEL and FRAME are in the time-varying part. The EVENT concept is borrowed from the Event ontology [9]. The provenance part includes the activities CAPTURE and SYNTHETIZE (see also Figure 4) and reuses the PROV Ontology [4] (PROV-O).

An EVENT has a LOCATION in a PLACE that is **sfWithin**⁹ a CELL – the basic spatial unit of aggregation of information in FraPPE – which, in turn, is **sfWithin** a GRID.

A PIXEL is the time-varying representation of a CELL. It is the only element in the conceptual model that carries information through the HASVALUE data property. As in image processing, this value represent a measure of intensity of some phenomena in the real world. For instance, it can represent the number of micro-posts posted in a given time interval within a certain CELL. Each PIXEL REFERS TO a single CELL, contrariwise a CELL could be REFERRED BY many different PIXELS that captures different information associate to the same CELL, e.g., the already mentioned number of micro-posts, but also the number of mobile phone calls or the number of good pick-ups.

Similarly, a FRAME is the time-varying counterpart of a GRID. It is a single complete *picture* in a series forming a *movie*. FraPPE distinguishes between two types of frames: the CAPTUREDFRAMES and the SYNTHETICFRAMES (see Figure 4). A CAPTUREDFRAME CONTAINS a PIXEL for every CELL in the GRID it WAS CAPTUREDFROM. Different FRAMES represent different images of the observed phenomena at the same SAMPLING TIME, e.g., a frame captured the volume of the social activity while another one captured the volume of the mobile phone calls at 12.00.

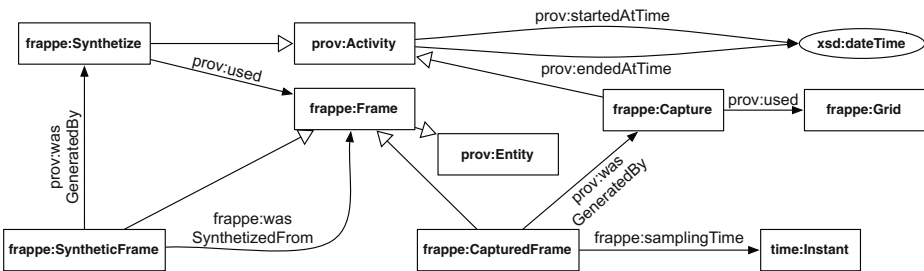


Fig. 4. The part of FraPPE that reuses the provenance ontology.

Figure 4 provides more details on the provenance part of FraPPE . The CAPTUREDFRAME and the SYNTHETICFRAME are specializations of FRAME

⁹ **sfWithin** refers to the **within** relationship defined in the Simple Features standard jointly issued by Open Geospatial Consortium and ISO.

which is an **Entity** in PROV-O. Also GRID is an **Entity**. This is because FraPPE proposes the ternary relationships CAPTURE and SYNTHETIZE as specialisations of the relationship Activity of PROV-O.

This allows to model that a CAPTUREDFRAME wasGeneratedBy a CAPTURE Activity startedAtTime τ_i and endedAtTime τ_j that used a given GRID. The object property WASCAPTUREDFROM is the result of the chaining of those two wasGeneratedBy and used object properties. Moreover, the value of the SAMPLINGTIME data property, which describes the CAPTUREDFRAME, is the one assigned to the startedAtTime data property that describes the captured activity.

Similarly, a SYNTHETICFRAME wasGeneratedBy by a SYNTHETIZE Activity that used one or more FRAMES. The idea is to derive a frame from one or more others. The synthetize operation can be a filter applied to the values of the pixels, or an aggregation of values of pixels across frames or the difference between the values associated to the pixels of two different frames. For a fully fledged algebra of some of the operations we intend to model, we refer interested readers to [11].

For instance, in our work on CitySensing [2], we captured for 2 months a frame every 15 minutes associating the value of each pixel to the volume of mobile phone calls in the 10,000 cells we divided Milan into. In this way, we captured 96 frames per day. Then, we synthetized 96 frames (one for each slot the day is divided into) associating to the value of each pixel a Gaussian distribution ϕ_{avg,std^2} where *avg* is the average and *std* is the standard deviation of the values associated to the respective pixels in the captured frames. Each pixel is, thus, associated with a statistical model of the volume of mobile phone calls. With these models, given a pixel in the frame captured at 12.00, whose value is v , we can compute an anomaly index to associate to the value of a pixel in a new synthetic frame using the formula $2\phi_{avg,std^2}(v) - 1$. A value of that pixel close to 1 (or -1) indicates an extraordinary higher (lower) volume compared to the usual activity in that slot from 12.00 to 12.15 in the associated cell of Milan.

3 Evaluation

In order to evaluate FraPPE, we check if it observes the five principles of Tom Gruber [7]: clarity, coherence, minimal encoding bias, minimal ontological commitment and extendibility.

FraPPE observes the *clarity* principle because all definitions are documented in natural language (see version of FraPPE on github⁶). The terms proposed in FraPPE are: (i) common terms in spatial-related vocabularies (e.g., PLACE, CELL, GRID); (ii) well known terms of the image processing domain (e.g., PIXEL, FRAME, CAPTURE, or SYNTHETIZE); and (iii) terms defined in other ontologies (e.g., Event, Instant, Entity, or Activity). Moreover, they are independent of the social and telecommunication domains, for which FraPPE was originally defined.

Listing 1. Fraction of the model representing ACM DEBS Grand Challenge 2015 Data

```

@prefix frGrid: <http://streamreasoning.org/debsGC/Grids/> .
@prefix frCell: <http://streamreasoning.org/debsGC/Cells/> .
@prefix frPixel: <http://streamreasoning.org/debsGC/Pixels/> .
@prefix frPlace: <http://streamreasoning.org/debsGC/Places/> .
@prefix frEvent: <http://streamreasoning.org/debsGC/Events/> .
@prefix frFrame: <http://streamreasoning.org/debsGC/Frames/> .
@prefix frCapture: <http://streamreasoning.org/debsGC/Captures/> .

frGrid:Grid_1
  gs:sfContains frCell:Cell_1, frCell:Cell_2 .

frCell:Cell_1
  a fr:Cell ;
  rdfs:label "39460"^^xsd:long ;
  fr:isReferredBy frPixel:1356995100000_39460 ;
  gs:sfContains frPlace:A ;
  gs:sfWithin frGrid:Grid_1 .

frPlace:A
  a sf:Point ;
  fr:isLocationOf frEvent:E_B ;
  gs:asWKT "POINT( 40.715008 -73.96244 )"^^gs:wktLiteral ;
  gs:sfWithin frCell:Cell_1 .

frEvent:E_A
  a fr4d:PickUpEvent ; a event:Event ;
  event:time [ a time:Instant ;
    time:inXSDDateTime "2013-01-01T00:00:00"^^xsd:dateTime ] ;
  fr:location frPlace:A> ;
  fr4d:hackLicense "E7750A37CAB07D0DFF0AF7E3573AC141"^^xsd:string ;
  fr4d:medallion "07290D3599E7A0D62097A346EFCC1FB5"^^xsd:string .

frEvent:E_B
  a fr4d:DropOffEvent ; a event:Event ;
  event:time [ a time:Instant ;
    time:inXSDDateTime "2013-01-01T00:02:00"^^xsd:dateTime ] ;
  fr:location frPlace:B ;
  fr4d:connected frEvent:E_A ;
  fr4d:fareAmount "3.5"^^xsd:double ;
  fr4d:mtaTax "5.0"^^xsd:double ;
  fr4d:paymentType "CSH"^^xsd:string ;
  fr4d:surcharge "5.0"^^xsd:double ;
  fr4d:totalAmount "4.5"^^xsd:double ;
  fr4d:tripDistance "0.44"^^xsd:long ;
  fr4d:tripTime "120"^^xsd:long .

frPixel:1356995100000_39460 a fr:Pixel ;
fr:isIn frFrame:1356995100000 ;
fr:refers frCell:Cell_1 .

frFrame:1356995100000
  a fr:CapturedFrame ;
  fr:contains frPixel:1356995100000_39460, frPixel:1356995100000_39461 ;
  fr:samplingTime [ a time:Instant ;
    time:inXSDDateTime "2013-01-01T00:05:00"^^xsd:dateTime ] ;
  fr:wasCapturedFrom frGrid:Grid_1 ;
  prov:wasGeneratedBy frCapture:1356995100000 .

```

Indeed, FraPPE terms can be used for other domains as demonstrated in publishing data for the Telecom Italia Big Data Challenge³.

FraPPE is *coherent*, i.e., all FraPPE inferences at T-box level are consistent with the definitions and in modelling A-boxes containing social, telecommunication, environment, traffic, and energy consumption data, we never inferred inconsistent or meaningless data.

FraPPE has a *minimal encoding bias* because it is encoded in OWL2-QL. Indeed, it uses only subclass axioms, property domain, property range and inverse object properties. We explicitly avoided adding cardinality restrictions, because in CitySensing [2] we use FraPPE to integrate data following an ontology-based data access approach.

FraPPE requires a *minimal ontological commitment*, meaning that, as Tom Gruber recommended, FraPPE makes as few claims as possible about the geo-located time-varying data being modelled allowing who uses FraPPE to specialise and instantiate it as needed.

Last but not least, we tested in details that FraPPE is *extendable* by modelling the dataset made available by ACM DEBS 2015 Grand Challenge⁴. The challenge proposes a taxi route analysis scenario based on a grid of 150x150 Kms with cells of 500x500 m. A stream of data represents the route of a taxi rides in terms of: (i) taxi description, (ii) pick-up and drop-off information (e.g., geographical coordinates of the place and time of the event), and (iii) ride information (e.g., tip, payment type and total amount). In the Listing 1, we report a subset of the information representing a single taxi ride in FraPPE. The pick-up EVENT represents the start of the ride and contains the taxi id. The drop-off EVENT represents the end of the trip and it is connected to all the information about the ride. The fragment models the geographical part of the ride using two PLACES within two different CELLS of a single GRID. Moreover, it models the time varying-part of the ride using two EVENTS captured in two PIXELS of a single FRAME along with the provenance part through the CAPTURE activity. Indeed, we reuse all FraPPE concepts, we specialise EVENT in `PickUpEvent` and `DropOffEvent`, and we add attributes (e.g., `tripTime`, and `totalAmount`) and an object property (i.e., `connected`) specific of the taxi ride domain.

Synthetic frames are also important in the challenge. One of the problems, assigned to the challengers, asks to compute the profitable cells for a given time interval. We wrote a SPARQL query that computes the answer; this is the SYTHETIZE activity that used CAPTUREDFRAMES of the type illustrated in Listing 1 to construct a SYTHETICFRAME where the values of the PIXELS are associated with the profitability of the CELLS they refer to.

4 Conclusions

In this paper, we propose FraPPE, a novel vocabulary that fills in the gap between low-level time-varying geo-located data and the high-level needs of map-centric visual analytics. Vocabularies to publish the low-level data exist, e.g., geosparql vocabulary [3], event ontology [9] or time ontology [8]. FraPPE reuses them. The high-level (oriented to visual analytics) part is missing, but the terms, which we choose for FraPPE vocabulary, are largely used among practitioners. The genesis of FraPPE terms can be found in the Social Pixel approach, proposed in [11], and in common practices used in geo-spatial knowledge discovery and data mining domain, e.g., [5] where the authors discuss on the optimal size of CELLS in GRIDS used for the analysis of human mobility.

FraPPE has an high potential. We demonstrated it applying FraPPE to city data integration within CitySensing [2]. A preliminary version of FraPPE was used in 2014 to publish open the data of Telecom Italia Big Data Challenge that covers the telecommunication, social, environmental, and energy domains. In this paper, we further exemplify such a potential by publishing in RDF the dataset of the ACM DEBS Grand Challenge 2015.

In designing FraPPE, we followed Tom Gruber's principles. Moreover, FraPPE is described using machine processable metadata (i.e., `label`, `creator`, `issued`, `versionInfo`, `priorVersion`, `license`, and `imports`). The ACM DEBS Grand Challenge dataset is accessible via SPARQL⁷ and described using VoID⁸.

As future works, we want to investigate how to improve the modelling of the values associated to the pixels. We started an investigation on ontologies of the units of measurement. Our best candidate, at this moment, is [10], because it allows also to model dimensionless quantities such as the anomaly index. Moreover, we want to investigate an effective approach to link a GRID to Where On Earth Identifiers¹⁰ in order to define a unique ID for every possible CELL on earth. Last, but not least, we want to foster the adoption of FraPPE by publishing more datasets; our intention is to start from the datasets release as open data within the Telecom Italia Big Open Data Challenge 2014.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. In: LDOW 2009 (2009). http://ceur-ws.org/Vol-538/ldow2009_paper20.pdf
2. Balduini, M., Della Valle, E., Azzi, M., Larcher, R., Antonelli, F., Ciuccarelli, P.: Citysensing: visual story telling of city-scale events by fusing social media streams and call data records captured at places and events. *IEEE MultiMedia* **22**(3) (to appear, 2015)
3. Battle, R., Kolas, D.: Geosparql: enabling a geospatial semantic web. *Semantic Web Journal* **3**(4), 355–370 (2011)
4. Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: PROV-O: The PROV Ontology. Tech. rep., W3C (2012)
5. Coscia, M., Rinzivillo, S., Giannotti, F., Pedreschi, D.: Optimal spatial resolution for the analysis of human mobility. In: *ASONAM 2012*, pp. 248–252 (2012)
6. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: Methontology: from ontological art towards ontological engineering (1997)
7. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.* **43**(5–6), 907–928 (1995)
8. Hobbs, J.R., Pan, F.: Time Ontology in OWL, September 2006
9. Raimond, Y., Abdallah, S.: The event ontology (2007). <http://motools.sf.net/event>
10. Rijgersberg, H., van Assem, M., Top, J.L.: Ontology of units of measure and related concepts. *Semantic Web* **4**(1), 3–13 (2013)
11. Singh, V.K., Gao, M., Jain, R.: Social pixels: genesis and evaluation. In: *ICM 2010*, pp. 481–490 (2010)
12. Thomas, J.J., Cook, K.A.: *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr. (2005)

¹⁰ <https://developer.yahoo.com/geo/geoplanet/guide/concepts.html>

The Transport Disruption Ontology

David Corsar^(✉), Milan Markovic, Peter Edwards, and John D. Nelson

dot.rural Digital Economy Hub, University of Aberdeen, Aberdeen AB24 5UA, UK
{dcorsar,m.markovic,p.edwards,j.d.nelson}@abdn.ac.uk

Abstract. This paper presents the Transport Disruption ontology, a formal framework for modelling travel and transport related events that have a disruptive impact on traveller’s journeys. We discuss related models, describe how transport events and their impacts are captured, and outline use of the ontology within an interlinked repository of the travel information to support intelligent transport systems.

1 Introduction

Transport is frequently viewed as a key sector within smart cities for improving citizen’s quality of life [2], [8], [13]. Such visions anticipate that IT systems will utilise data made available by the integration of physical and digital transport infrastructures to address the mobility challenges faced by cities today. One such challenge is minimising the impact of transport disruptions [4]: road congestion is estimated to cost an average of 1% of GDP across the European Union [4], while the absence of real-time information about the impact of disruptions is a major factor in the dissatisfaction with, and reduced attractiveness of public transport [10, 11]. However, tackling such problems requires addressing the challenges of data interoperability, analysis, information extraction, and reasoning presented by such environments - challenges that Semantic Web and linked data are key technologies in overcoming [9].

The shared models and vocabularies provided by ontologies are fundamental to any Semantic Web solution. Ontologies have previously been used to create an integrated ecosystem of the transport information required to support a real-time passenger information system [3]. The Transport Disruption ontology described in this paper enables the extension of such ecosystems with details of travel and transport related events that can have disruptive impacts on mobility. As such, the Transport Disruption ontology contributes a key model to the ongoing work of the Semantic Web community in addressing the data challenges of smart cities, and a major transport challenge faced by society.

This paper is structured as follows: section 2 describes the ontology; section 3 details its application in a use case; section 4 discusses related work; and section 5 concludes the paper.

The research described here is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub; award reference: EP/G066051/1.

2 The Transport Disruption Ontology

The Transport Disruption ontology provides a formal framework for modelling travel and transport related events that can have a disruptive impact on an agent’s planned travel. The ontology is available at its namespace:

<http://purl.org/td/transportdisruption#>, abbreviated to ‘td’ throughout, and the associated github project¹. Figure 1 outlines the ontology schema.

The ontology was defined following an analysis of disruption information provided by transport authorities and operators of bus, rail, ferry, and air public transport services in the UK. This identified the requirement to capture the occurrence of an event in terms of its type, location, time period, compound and causal relationships to other events, and any impact experienced by agent(s) that have to adapt their plan(s) because of it. Following linked data publishing best practise [5], Linked Open Vocabularies² and the Linked Open Data³ cloud were reviewed to identify existing ontologies that could be used to meet these requirements. The selected ontologies and their integration with the Transport Disruption ontology are discussed below.

The main modelling choice focused on the representation of disruptions. The transport research community [6, 7], [12] define the notion of disruptive events, i.e. events that “impact on the supply of transport (infrastructure or services), costs of using transport, or some combination” [7]. Such events can affect single or multiple transport links in a given area, and impact on travellers in a way greater than that experienced through the typical day-to-day variability in travel plans.

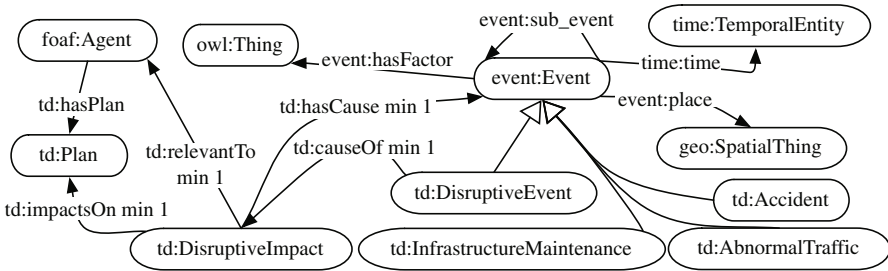


Fig. 1. The Transport Disruption ontology schema, with example transport events.

This is captured by the *td:DisruptiveImpact* and *td:DisruptiveEvent* classes in the Transport Disruption ontology. A disruptive impact is defined as the action of one or more events causing or tending to prevent an agent’s planned travel from continuing as expected. The *td:DisruptiveEvent* class is aligned to the

¹ <https://github.com/transportdisruption/transportdisruption.github.io>

² <http://lov.okfn.org/dataset/lov>

³ <http://lod-cloud.net>

Event ontology⁴ as a subclass of *event:Event*. *td:DisruptiveEvent* is designed to allow events of any type that have caused a *td:DisruptiveImpact* to be classified as disruptive; as such, it is not the intention that this class is extended further. The Event ontology defines events as an arbitrary classification of a space/time region, by a cognitive agent. This aligns with the transport research perspective of disruptive events as, for example, a traveller classifying roadworks occurring within an area and time that they are traveling as causing an undesirable impact on their journey. Use of the Event ontology also allows existing data defined with this model to be used with the Transport Disruption ontology.

td:DisruptiveImpact is an extension point for developers to define the types of impact relevant to their intended usage of the ontology. This may include descriptions of delays to an agent's journey or details of a journey that cannot be completed due to, for example, a cancelled flight. A minimum cardinality constraint defines that each *td:DisruptiveImpact* individual must define the *foaf:Agent* that it is *td:relevantTo*. The Friend-of-a-Friend (FOAF) ontology⁵ was selected as it is a commonly used model for describing people, organisations, and groups, and its use allows reuse of existing FOAF data with the Transport Disruption ontology.

td:Plan provides an extension point for developers to describe an agent's plan that may be impacted by an event. For example, this could be a person's travel itinerary defined by an expected departure time, location and means of transport to a destination; or a bus schedule providing expected arrival times at stops along a route as defined by the bus operator. Each *td:DisruptiveImpact* also references at least one *td:Plan* that it *td:impactsOn*.

A cardinality constraint is defined stating that each individual of type *td:DisruptiveImpact* must reference at least one *event:Event* through the *td:hasCause* property. Similarly, each *td:DisruptiveEvent* must reference at least one *td:DisruptiveImpact* that it caused via the *td:causeOf* property. An *owl:inverseOf* relationship is defined between *td:hasCause* and *td:causeOf*, ensuring the correspondences between individuals of these classes can be materialised by a description logic reasoner. A reasoner can then also infer the *td:DisruptiveEvent* type for *event:Events* that have caused a *td:DisruptiveImpact*.

New concepts extending *event:Event* are defined to capture the different types of events that may disrupt an agent's travel plans. These are based on the analysis of published disruption information and aspects of the DATEX II⁶ specification. DATEX II models the road traffic domain and exchange of data between road management systems, including types of road disruption. The types of events modelled as subclasses of *event:Event* include *td:OperatorAction*, *td:InfrastructureManagement*, *td:RoadClosure*, *td:SpeedRestriction*, *td:InfrastructureConstruction*, *td:InfrastructureMaintenance*, *td:TrafficElementEvent*, *td:AbnormalTraffic*, *td:Accident*, *td:EnvironmentalConditions*, *td:InfrastructureFailure*, *td:InfrastructureFault*, *td:Obstruction*, *td:PublicTransportEvent*, *td:PublicTransportDelay*, *td:PublicTransportCancellation*, and *td:PublicTransportDiversion*.

⁴ Namespace <http://purl.org/NET/c4dm/event.owl#>, abbreviated to 'event'.

⁵ Namespace <http://xmlns.com/foaf/0.1/>, abbreviated to 'foaf'.

⁶ <http://www.datex2.eu/>

The Event ontology defines that events are associated with a temporal region describing their time of occurrence using the OWL-Time ontology⁷. The Timeline ontology⁸ is also recommended to model intervals in terms of start and end timestamps. Defining such temporal intervals for transport-related events is encouraged as it can be used to determine if an event is completed, active, or planned, which are frequently used terms for defining the status of disruptions.

The Event ontology also defines that events are associated with a spatial object that locates the event using the WGS84 ontology⁹. Providers of travel disruption information typically use a variety of location references, including a point with latitude and longitude values, roads, public transport access points (bus stops, railway stations, etc.), and geographic areas such as counties, cities, and localities. Classes from other ontologies representing such locations can be used with the Transport Disruption ontology by defining alignments to the WGS84 ontology. For example, defining the *Road* class from the Linked-GeoData ontology¹⁰ or *Stop* class from the Transit ontology¹¹ as subclasses of *geo:SpatialThing*.

Compound relationships between events can be modelled using the *event:sub_event* property. This can be used to capture the compound nature of disruptions: for example, a *td:RoadClosure* occurring as part of a *td:RoadWorks* event. The *event:factor* property links events to *owl:Things* that are factors of the event. This can capture additional relationships between events, such as a *td:AbnormalTraffic* event contributing to a *td:PublicTransportDelay* event.

3 Social Journeys Use Case

A motivation for the Transport Disruption ontology is to support the use of integrated data sets of travel and transport information to identify disruptions that agents may encounter during travel. The Social Journeys project¹² is developing a system to provide this information for bus users within the city of Aberdeen, UK. This section discusses the use of the Transport Disruption ontology within the project and outlines a versatile approach for obtaining the data required by the system from open data repositories¹³.

3.1 Integrated Travel and Transport Data Sets

The Social Journeys knowledge base (a sample of which is provided in Figure 2) contains several types of data that can be used to geolocate an event, reflecting

⁷ Namespace <http://www.w3.org/2006/time#>, abbreviated to ‘time’.

⁸ Namespace <http://purl.org/NET/c4dm/timeline.owl#>, abbreviated to ‘tl’.

⁹ Namespace http://www.w3.org/2003/01/geo/wgs84_pos#, abbreviated to ‘geo’.

¹⁰ Namespace <http://linkedgeo.org/ontology/>, abbreviated to ‘lgd’.

¹¹ Namespace <http://vocab.org/transit/terms/>, abbreviated to ‘transit’.

¹² <http://www.dotrural.ac.uk/socialjourneys>

¹³ Associated code is available at <https://github.com/SocialJourneys>.

```

sjd:road1 a lgd:Road; sj:way sjd:way1, sjd:way2.
sjd:stop1 a transit:Stop; sj:onWay sjd:way1.
sjd:stop2 a transit:Stop; sj:onWay sjd:way2.
sjd:ss1 a transit:ServiceStop; transit:stop sjd:stop1.
sjd:ss2 a transit:ServiceStop; transit:stop sjd:stop2.
sjd:FA18 a transit:Service; transit:serviceStop sjd:ss1, sjd:ss2.
sjd:roadworks_1 a td:RoadWorks; event:place sjd:road1;
  event:sub_event sjd:laneClosure;
  event:time [
    tl:beginsAtDateTime "2015-04-01T06:30:00"^^xsd:dateTime;
    tl:endsAtDateTime "2015-04-02T17:30:00"^^xsd:dateTime].
sjd:laneClosure a td:LaneClosure; event:place sjd:road1;
  event:time [
    tl:beginsAtDateTime "2015-04-01T06:30:00"^^xsd:dateTime;
    tl:endsAtDateTime "2015-04-02T06:30:00"^^xsd:dateTime].
sjd:serviceDelay1 a td:PublicTransportDelay; transit:service sjd:FA18;
  sj:delayLengthMinutes "10"^^xsd:integer; event:factor sjd:roadworks_1.
sjd:journey a td:Plan; transit:service sjd:FA18;
  sj:expectedDepartureTime "09:30:00"^^xsd:time;
  sj:expectedArrivalTime "10:15:00"^^xsd:time;
  sj:daysOfTravel "2015-04-01"^^xsd:date, "2015-04-02"^^xsd:date,
    "2015-04-03"^^xsd:date.
sjd:david a foaf:Agent; td:hasPlan sjd:journey.

```

Fig. 2. Sample of the Social Journeys knowledge base.

the variety of location references used by providers of travel disruption information. This includes details of roads within Aberdeen, which are extracted¹⁴ from OpenStreetMap¹⁵ (OSM). The exported XML file consists of *nodes*, geospatial points with latitude and longitude values, and *ways*, ordered lists of nodes used to represent polygons (e.g. buildings) and polylines (e.g. roads) with attributes providing further details such as the building or road name. A script converts this into RDF and imports it to the Social Journeys knowledge base.

Access points to public transport and details of localities are used to associate events with bus stops, railway stations, airports and areas (localities) within the city. A triplification script converts the CSV data files for the NaPTAN and NPTG open data sets¹⁶ published by the UK Department for Transport¹⁷. Extracted bus stops are also linked to the description of the road that they are on.

Details of bus services within the city are extracted from the Traveline open data repository¹⁸ of bus operators, routes, and schedules. This data is

¹⁴ Road details are extracted using the tool at <http://extract.bbbike.org>.

¹⁵ <http://www.openstreetmap.org>

¹⁶ NaPTAN provides details of public transport access points in the UK, (<http://data.gov.uk/dataset/naptan>), and NPTG defines the areas of the UK served by public transport (<http://data.gov.uk/dataset/nptg>).

¹⁷ <http://data.gov.uk/publisher/department-for-transport>

¹⁸ <http://www.travelinedata.org.uk/traveline-open-data>

available as XML files encoded using the TransXChange schema¹⁹; a parser creates RDF descriptions using the Transit ontology of bus services that use stops within Aberdeen. The extracted bus service descriptions include the service name, operator, and references to the bus stops used.

Event descriptions are currently manually created based on reports from sources including the transport authority²⁰, bus operator²¹, and a local radio station²². Scripts to create event descriptions automatically are being developed.

Finally, as part of the system being developed, travellers are requested to provide details of their planned bus journeys. Each journey is described in terms of the departure and arrival times, days of travel, and the journey stages. Each stage defines the origin bus stop, bus service used, and destination stop.

3.2 Identifying Potential Disruptive Impacts

Potential travel disruptions can be identified by performing spatial and temporal matching of the events and planned journeys in the knowledge base. For example, a spatial match can be based on a common piece of infrastructure (e.g. road), while a temporal match can be determined if the time period of an event's occurrence intersects with that of a journey, as illustrated by following query:

```

construct {
  _: _ a td:DisruptiveImpact; td:impactsOn ?journey;
    td:hasCause ?event; td:relevantTo ?agent.
} where {
  select distinct ?journey ?event ?agent where {
    ?journey a td:Plan; transit:service ?service;
      sj:expectedArrivalTime ?arrivalTime;
      sj:expectedDepartureTime ?departureTime; sj:daysOfTravel ?travelDay.
    ?agent td:hasPlan ?journey.
    ?event a event:Event; event:place/sj:way ?way;
      event:time/tl:beginsAtDateTime ?eventStartTime;
      event:time/tl:endsAtDateTime ?eventEndTime.
    ?service transit:serviceStop/transit:stop/sj:onWay ?way.
    bind (xsd:dateTime(xsd:string(?travelDay)+"T"+xsd:string(?arrivalTime))
      as ?ja)
    filter (?eventStartTime <= ?ja)
    bind (xsd:dateTime(xsd:string(?travelDay)+"T"+xsd:string(?departureTime))
      as ?jd)
    filter (?eventEndTime >= ?jd)
  }
}

```

The *td:DisruptiveImpact* individuals returned by this query could be used by the system to alert users about the possible disruption. The simple approach shown to spatial matching could be extended to include more complex spatial

¹⁹ <https://www.gov.uk/government/collections/transxchange>

²⁰ <http://trafficscotland.org/>

²¹ http://www.firstgroup.com/ukbus/aberdeen/travel_news/service_updates/

²² <http://www.northsound1.com/travel>

reasoning, for example, using GeoSparql²³ functions for boundary box matches. Storing the *td:DisruptiveImpact* individuals (after assigning a URI) would enable stakeholders to perform various types of analysis. For example, the bus operator could attempt to identify which bus services may be affected by events today or tomorrow, or, as part of reviewing historical performance, query for any services that were affected by disruptions during a specific time period.

This example illustrates the initial work necessary for intelligent travel disruption reporting. However, the knowledge base is by its nature dynamic, changing as further details of events become available or new events occur. Future work will focus on recording the provenance of knowledge base updates and any reasoning performed with the data. This will involve alignment of the Transport Disruption ontology with PROV-O²⁴, the W3C recommendation for representing provenance information. Indicating the quality, particularly of *td:DisruptiveImpact* and *td:DisruptiveEvent* individuals will be a key factor in supporting people decide if they should act upon the data or not. We plan to use the Qual-O ontology and assessment framework [1] to define metrics that evaluate the quality of such data. For example, a metric could rate the relevancy of a *td:DisruptiveImpact* to a person by determining if roadworks are located on their expected route of travel: if so the relevancy would be high, otherwise the relevancy score would be reduced based on the proximity of the roadworks to the route. This reflects the observation that while roadworks may be on the same road (as in the example above) they may not actually be encountered by the person.

4 Related Work

Along with semantic models of public transport routes and schedules provided by the Transit and GTFS²⁵ ontologies, models have also been defined for other aspects of the transport domain. These include the Road Traffic Management Ontology²⁶, the scope of which is limited to describing the actions a moving vehicle can perform (e.g. accelerate, change lane), and its relation to other vehicles (e.g. relative speed, road position). The draft Road Accident Ontology²⁷ models road accidents in terms of the vehicles and living beings involved, relevant documents (e.g. driver licence, insurance certificate), location, and organisations (e.g. insurance companies). However, it is limited to only defining a single type of event (road accident) and does not consider any consequential impact. The Passim ontology²⁸ models systems that convey transport information to travellers. Passim models such systems in terms of name, how it is accessed (website, SMS, mobile application), and coverage in terms of the modes of transport and towns, cities, and geographic regions that information is provided for.

²³ <http://geosparql.org>

²⁴ <http://www.w3.org/ns/prov#>

²⁵ <http://vocab.gtfs.org/terms>

²⁶ <http://lov.okfn.org/dataset/lov/vocabs/traffic>

²⁷ <http://www.w3.org/2012/06/rao.html#owl>

²⁸ <http://data.lirmm.fr/ontologies/passim>

5 Conclusion

Ontologies provide a key technology for supporting data integration. Alignment of the Transport Disruption ontology with existing models, such as Event, FOAF, Transit, and LinkedGeoData extends the existing semantic modelling capabilities for integrated mobility data sets. The Transport Disruption ontology enables descriptions of travel and transport related events and their disruptive impacts on mobility. The defined event types and details of their impacts can be extended as necessary for use in different applications. As such, we argue that the Transport Disruption ontology provides a necessary component in enabling the contribution of Semantic Web efforts to addressing the mobility challenges faced by society today and in future smart cities. Along with the future work discussed above, we plan further evaluation of the ontology through use cases explored in collaboration with the Semantic Web and transport research communities.

References

1. Baillie, C., Edwards, P., Pignotti, E.: Qual: A provenance-aware quality model. *Journal of Data and Information Quality* **5**(3), 12:1–12:22 (2015)
2. BSI Group. PAS 181:2014 Smart city framework - Guide to establishing strategies for smart cities and communities (2014)
3. Corsar, D., Edwards, P., Baillie, C., Markovic, M., Papangelis, K., Nelson, J.: Short paper: citizen sensing within a real time passenger information system. In: *Proceedings of the 6th International Workshop on Semantic Sensor Networks.*, vol. 1063, pp. 77–82. CEUR (2013)
4. European Commission. ITS Roadmap Outline Intelligent Transport Systems for more efficient, safer, cleaner road transport (2007)
5. Hyland, B., Ateazing, G., Villazon-Terrazas, B.: Best practices for publishing linked data. W3C Working Group Note (2009). <http://www.w3.org/TR/ld-bp/>
6. Jenelius, E., Mattsson, L.-G.: Road network vulnerability analysis of area-covering disruptions: A grid-based approach with case study. *Transportation Research Part A: Policy and Practice* **46**(5), 746–760 (2012). Network vulnerability in large-scale transport networks
7. Marsden, G., Docherty, I.: Insights on disruptions as opportunities for transport policy change. *Transportation Research Part A: Policy and Practice* **51**, 46–55 (2013)
8. MasterCard Enterprise Partnerships. Connecting cities Mobility: Unlocking potential in emerging markets (2015)
9. Omitola, T., Breslin, J., Barnaghi, P. (eds.): S4SC 2014 Semantics for Smarter Cities. CEUR Workshop Proceedings, vol. 1280 (2015)
10. Passengerfocus. Bus passenger views on value for money (2013)
11. Passengerfocus. Passenger information when trains are disrupted (2014)
12. Pender, B., Currie, G., Delbosc, A., Shiwakoti, N.: Improving bus bridging responses via satellite bus reserve locations. *Journal of Transport Geography* **34**, 202–210 (2014)
13. The European Innovation Partnership on Smart Cities and Communities. Strategic implementation plan (2013)

Experiments

LOD Lab: Experiments at LOD Scale

Laurens Rietveld^(✉), Wouter Beek, and Stefan Schlobach

Department of Computer Science,
VU University Amsterdam, Amsterdam, The Netherlands
{laurens.rietveld,w.g.j.beek,stefan.schlobach}@vu.nl

Abstract. Contemporary Semantic Web research is in the business of optimizing algorithms for only a handful of datasets such as DBpedia, BSBM, DBLP and only a few more. This means that current practice does not generally take the true *variety* of Linked Data into account. With hundreds of thousands of datasets out in the world today the results of Semantic Web evaluations are less generalizable than they should and — this paper argues — can be. This paper describes LOD Lab: a fundamentally different evaluation paradigm that makes algorithmic evaluation against hundreds of thousands of datasets the new norm. LOD Lab is implemented in terms of the existing LOD Laundromat architecture combined with the new open-source programming interface *Frank* that supports Web-scale evaluations to be run from the command-line. We illustrate the viability of the LOD Lab approach by rerunning experiments from three recent Semantic Web research publications and expect it will contribute to improving the quality and reproducibility of experimental work in the Semantic Web community. We show that simply rerunning existing experiments within this new evaluation paradigm brings up interesting research questions as to how algorithmic performance relates to (structural) properties of the data.

1 Introduction

While the exact size of the Web of Data is unknown, there is broad agreement that the volume of data published according to Linked Open Data (LOD) standards has to be counted in tens, if not hundreds, of billions of triples by now, originating from hundreds of thousands of datasets from various domains and provenance. This amount and broadness of information makes the Web of Data ideal for testing various types of algorithms and an exciting object of study. As this is widely recognized it is no surprise that many research papers have been published in the recent past that use parts of this enormous and rich collection.

Unfortunately, true large-scale evaluation, both in terms of volume and variety have proven to be much harder to come by than one would expect. One of the main reasons for this is the heterogeneity and user-unfriendliness of the most wide-spread dissemination strategy for Linked Data today: datadumps. Most researchers and application programmers will recognize the problem of dealing with various serialization formats and juggling with syntax errors as well as other data document-specific idiosyncrasies. With the core research being

on algorithms and evaluations, data collection, cleaning and harmonization can easily become a barrier too high to overcome.

To avoid these tedious and painful efforts of integrating hundreds of thousands of heterogeneous datasets most current studies with evaluations focus on data published through APIs, e.g., using SPARQL. Although this often provides high-volume datasets for testing, this leads to a strange imbalance in current practice: of the hundreds of thousands of available datasets [15], only around 260 are available through live query endpoints [6], and of the latter less than 10% dominate the evaluation landscape (see Section 2). As such, question-marks have to be put on the generalizability and maybe even validity of many of the results.

Two technological developments of the recent year have changed the situation significantly, though. First, the LOD Laundromat [3], a platform that cleans, harmonizes and republishes Linked Data documents, now serves more than 37 billion triples from over 650,000 data documents in a single, uniform and standards-compliant format. By (re)publishing very many datasets in exactly the same, standards-compliant way, the LOD Laundromat infrastructure supports the evaluation of Semantic Web algorithms on large-scale, heterogeneous and real-world data. In [15] the LOD Laundromat, which had been serving static clean data files until that point, was combined with the Linked Data Fragments (LDF) paradigm [19], thereby offering live query access to its entire collection of cleaned datasets through Web Services (<http://lodlaundromat.org>).

While these Web Services provide a good interface for some use cases, e.g. downloading a specific data document, the large-scale evaluation of a Semantic Web algorithm against thousands of data documents is still relatively time consuming. This is why we present LOD Lab: an integrated approach towards running Linked Data evaluations in the large. The LOD Lab approach is implemented by pairing the LOD Laundromat backend with *Frank*, an open-source¹ and simple yet flexible front-end programming interface for conducting large-scale experiments over heterogeneous data.

Since the LOD Lab approach defaults to running Semantic Web evaluations against hundreds of thousands of data documents, it introduces a problem that would have been considered a luxury problem even two years ago: now that 650,000 datasets are available, choosing suitable ones for specific experiments becomes a non-trivial task. Fortunately, *Frank* facilitates informed selection by filtering on domain vocabularies and by using metadata about the scraping and cleaning process as well as metadata about the structural properties of the data.

This paper makes the following contributions:

- A new way of conducting Linked Data experiments that incorporates both volume and variety while at the same time allowing the set of considered data documents to be limited according to structural constraints. The motivation for this novel approach is given in Section 2.

¹ See <https://github.com/LODLaundry/Frank>

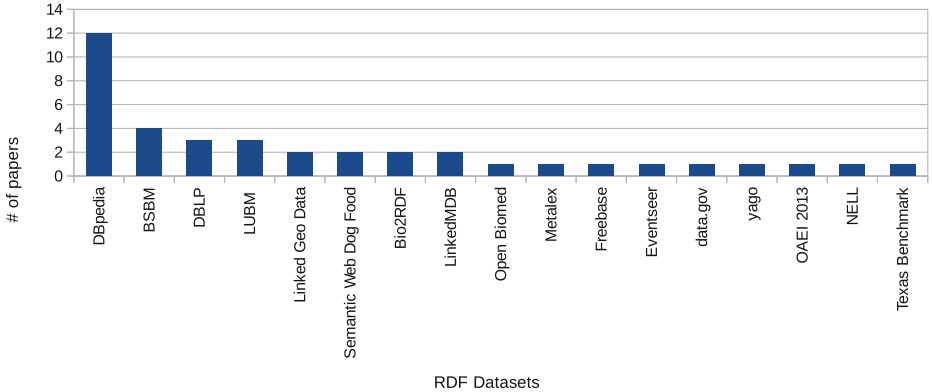


Fig. 1. Overview of datasets used in evaluations of papers accepted in the ISWC 2014 research track. For each dataset the number of articles that use it is shown.

- The introduction of a simple yet versatile programming interface called *Frank* for running large-scale Linked Data evaluations from the command-line. Section 4 discusses the usage, functionality and implementation of *Frank*.
- A demonstration of the viability of the LOD Lab evaluation approach by rerunning three experiments reported in recent Semantic Web conference publications, but now by using hundreds of thousands of data documents. The experiments are described in Section 5.

2 Motivation

Figure 1 gives an overview of the datasets that are used in 20 papers that were accepted in the ISWC 2014 research track. It only includes papers that evaluate Linked Datasets, excluding ones that evaluate algorithms on relatively small ontologies, non-RDF datasets or streamed data. The figure shows that 17 datasets are used in total. The number of datasets per article varies between 1 and 6 and is 2 on average.

The figure shows that most evaluations are conducted on only a handful of datasets. Even the total collection of datasets that are used in these 20 papers is not very large. This implies that many papers evaluate against the same datasets, most often DBpedia. This means that it is generally unclear to what extent published results will transfer to other datasets, specifically those that are only very rarely evaluated against. This is the problem of the *generalizability* of Semantic Web research results (Problem 1).

Problem 1. *By using very few datasets in scientific evaluations, the generalizability of Semantic Web research results is often unknown.*

The reason for Problem 1 is that current evaluation practice does not scale over heterogeneous data, i.e. we face a problem of *variety*. The problem is no

longer with the *volume* of the data since most of the datasets that are never evaluated against are smaller than some of the datasets that are currently used in evaluations. While it is sufficiently easy to obtain, load and evaluate one dataset, contemporary practice shows that it is still difficult to do the same thing for very many datasets.

One critique that may be leveled against our identification of Problem 1 is that the most often used datasets are evaluated most often and that evaluation practice is simply in line with data usefulness or relevance. However, most of the algorithms and approaches that are evaluated in Semantic Web research target generic applicability. Specifically, none of the above 20 papers claims to develop a *dataset-specific* approach. Moreover, that a dataset is popular does not imply that results obtained over it are indicative of Linked Data in general and can be transferred to other datasets. This is specifically true for Linked Data where the expressiveness of the language allows datasets to differ considerably.

Empirical surveys have documented the restricted state of today's Semantic Web deployment.[6,12] Many datasets are only available as data dumps, lack dereferenceable URIs, cannot be downloaded due to HTTP errors, cannot be unpacked due to archive errors, or cannot be loaded into Semantic Web tools due to syntax errors. These idiosyncrasies imply in practice that the human costs to run experiments usually increases linearly with the number of datasets. This implies that eager researchers can use one, two, or even six datasets in their evaluations. There is no way, though, to expect hundreds, thousands or even hundreds of thousands of datasets in their evaluations. This lack of variety is due to the fact that the use of every single dataset requires some manual operations (and often repeatedly very similar operations) in order to overcome the aforementioned idiosyncrasies (Hypothesis 1).

Hypothesis 1. *The main reason why experiments are run on very few datasets is that for every dataset a certain amount of manual labor is needed.*

If Hypothesis 1 is correct, then the solution to Problem 1 is to make the human cost of using datasets independent from the number of datasets that is used (Solution 1). The human cost involved in evaluating against datasets should not only be independent of the number of datasets, but should also be low. Both these features can be achieved by fully automating the tasks of obtaining, loading, and using datasets. The LOD Laundromat [3] solves this problem by providing a fully automated infrastructure for disseminating heterogeneous datasets in a uniform and standardized format. It (re)publishes data as cleaned datadumps and, more recently, through Web Services. Neither method is suitable for large-scale evaluation, which requires tools support for fetching, selecting and application of custom algorithms over the appropriate subset of datasets from the LOD Laundromat.

Solution 1. *Make the human effort needed to obtain, load, and use a collection of datasets independent from the size of the collection.*

While running more evaluations against hundreds of thousands of datasets will increase the generalizability of Semantic Web approaches, it also creates a

new problem: selectivity (Problem 2). Not every evaluation needs to be, should be nor can be performed on all the available datasets published through the LOD Laundromat. So the question arises which datasets to choose.

Problem 2. *There are currently no means to select those datasets that are pertinent to a given algorithm or approach based on properties of the data.*

The ability to select datasets based on properties of the data also relates to another problem. It is well known, and supported by our results in Section 5 that evaluation outcomes sometimes differ radically for different datasets. Even though this is an interesting observation in itself, it is more pertinent to inquire as to *why* and *how* performance differs over datasets. This is a topic that has traditionally not been touched upon very often in the context of Semantic Web evaluations. LOD Lab will radically simplify future studies in the Semantic Web community to gain insight in how the performance of Semantic Web approaches relates to properties of the data (Problem 3).

Problem 3. *Current evaluations do not relate evaluation outcomes such as the performance of the evaluated algorithm or approach to properties of the data.*

The solution to Problems 2 and 3 is to allow datasets to be selected based on various criteria (Solution 2). These criteria should include a dataset's metadata (e.g., when it was crawled) and structural properties of the data (e.g., the number of unique triples it contains).

Solution 2. *Allow datasets to be selected based on their properties, including the dataset metadata, and structural properties of the data.*

3 Related Work

3.1 Evaluation Frameworks and Benchmarks

Evaluation frameworks and benchmarks have played an important role in Semantic Web research. Many of the previous efforts focused on evaluation of storage and query answering, e.g., in the area of RDF processing and SPARQL query answering, such as the Berlin Benchmark [4], SP²Bench [17], LUBM [9] and Fedbench [18] or LDBC[5]. Those benchmarks usually provide datasets and corresponding query sets, in order to level the playing field and allow for a fair comparisons between tools. Such approaches are a useful source for particular Linked Data research areas. However, most of these approaches present a static or even synthetic dataset. LOD Lab differs from the above by allowing experiments over an extremely high percentage of the real datasets that were published.

Relevant is the Ontology Alignment Evaluation Initiative [7] (OAEI) which presents datasets, and gold standards to relate results to, and a framework for doing so. Most importantly, the OAEI has been using the SEALs² evaluation

² <http://www.seals-project.eu/>

platform for years now. SEALs supports experiments on ontology alignment with similar functionality as the LOD Lab supports scalability analytic experiments over multiple various heterogeneous data sources.

3.2 Dataset Collections

The most common large dataset collection to date is a Linked Data crawl published as the Billion Triple Challenge [11] (BTC). The key goal of the Billion Triple Challenge is *‘to demonstrate the scalability of applications, as well as the capability to deal with the specifics of data that has been crawled from the public web’*. BTC has indeed proven to facilitate such research, and it has been used in a wide range of papers. The latest BTC dataset collection was published in 2012, and contains 1.4 billion triples. But lets be frank: where this volume used to be ‘large’, it has now suffered from inflation and is superseded by several larger datasets. Additionally, BTC suffers from the same idiosyncrasies found in other parts of the LOD Cloud: several BTC files contain a sizable number of duplicates and serialization errors³. Although the BTC has proven successful for testing algorithms for ‘large’ data, it lacks the meta-data for dealing with variety: neither dataset characteristics or detailed crawling provenance are available.

Another collection of datasets is LODCache, a Linked Data crawl published via a SPARQL endpoint, exposing (at the time of writing) 34.5 billion triples. Though an interesting source of data, the limitations that the endpoint imposes makes extracting and downloading these datasets difficult. Additionally, no information is published on the crawl mechanism behind it, and the web service lacks meta-data of both the crawl and datasets as well. I.e., this service provides data in a large volume, but lacks the meta-data to *select* datasets.

3.3 Collecting Data on Scale

Some resort to crawling Linked Data themselves considering the lack of available dataset collections. A common tool for this approach is LDspider [13], a Linked Data crawler which supports a wide range of RDF serialization formats, and traverses the Linked Data cloud automatically. This approach requires a large seed list of dataset locations, considering an automatic crawl would need many dereferenceable URIs to automatically discover new datasets. Therefore, LDspider is suitable for some, but crawling larger parts of the LOD Cloud both requires manual effort for curating the seed list, as well as a significant hardware investment.

4 Implementation

LOD Laundromat provides a wealth of data, including the corresponding meta-data such as crawling provenance and structural properties of data documents.

³ See <http://lodlaundromat.org/resource/c926d22eb49788382ffc87a5942f7fb3>

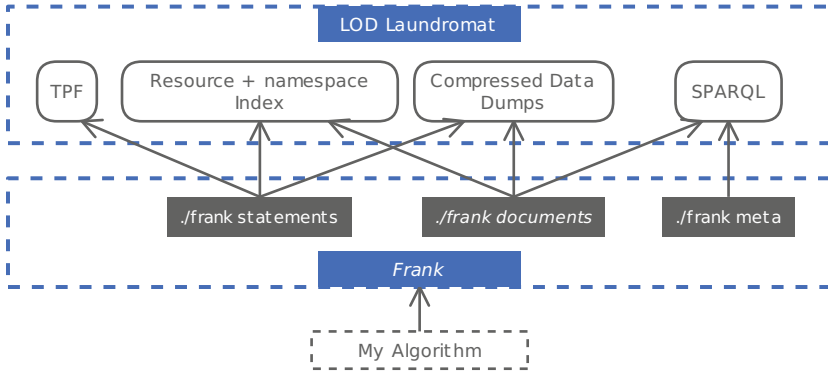


Fig. 2. The implementation architecture for *Frank* and its dependencies on the LOD Laundromat Web Services.

The latter are disseminated through a public SPARQL endpoint⁴. LOD Laundromat data can be accessed by writing a custom script that queries the metadata endpoint to fetch pointers to the relevant data documents. Those pointers either give access to the complete data document or to the Linked Data Fragment API for that particular document. The problem with this approach is that a user needs to be acquainted with the scraping and structural metadata schema used by LOD Laundromat. Since the latter is quite elaborate, designed with versatility rather than usability in mind, the Web Services do not implement Solution 1.

We therefore introduce *Frank*⁵, a Bash interface that makes it easy to run evaluations against very large numbers of datasets. By implementing *Frank* in Bash it can be used by all except Windows users who do not want to install Cygwin⁶. Since *Frank* is a plain text file it requires no installation and no inclusion in a software repository or app store, nor does it depend on a specific programming paradigm. As with any Bash script, in- and output can be straightforwardly piped from/to other programs and scripts.

Frank implements Solution 1 since it allows evaluations over hundreds of thousands of data documents to be run by typing a single command (see Section 5 for the commands that were used to scale-up existing experiments). *Frank* implements Solution 2 by offering mechanisms to select datasets according to their metadata, and structural properties (see below for the concrete properties that are supported).

Below, we discuss the three main features of *Frank*: streamed triple retrieval, streamed document retrieval, and metadata retrieval.

⁴ See <http://lodlaundromat.org/sparql>

⁵ A technical overview of *Frank* was presented at the ESWC Developers Workshop [2].

⁶ See <https://www.cygwin.com/>

4.1 Streamed Triple Retrieval

frank statements allows individual atomic statements or triples to be retrieved. When called without arguments this streams all 37 billion triples by fetching and unpacking the Gzipped LOD Laundromat data dumps. If called with the command-line flags `--subject`, `--predicate`, and/or `--object`, only triples that contain the specified subject-, predicate- and object-term are returned. These three flags mimic the expressivity of the Linked Data Fragment (LDF) [19] Web API. They are expressively equivalent to SPARQL queries with a single-line Basic Graph Pattern (BGP) [10]. LDF supports streamed processing though a self-descriptive API that uses pagination in order to serve large results in smaller chunks. If called with a subject, predicate and/or object flag, **frank statements** interfaces with the LOD Laundromat index⁷ which contains a mapping between all LOD Laundromat resources and documents. For these documents, *Frank* connects with the Linked Data Fragments API for, handling the LDF pagination settings in order to ensure a constant stream of triples. The LDF API is able to answer triple pattern requests efficiently by using the Header Dictionary Triples⁸ (HDT) technology. HDT is a binary, compressed and indexed serialization format that facilitates efficient browsing and querying of RDF data at the level of single-line BGPs. HDT files are automatically generated for all data documents that are disseminated by the LOD Laundromat backend.

4.2 Streamed Document Retrieval

frank documents allows individual documents to be retrieved. The command interfaces with the SPARQL endpoint and LOD Laundromat index in order to find data documents that satisfy the given properties.

The following selection mechanisms are supported by **frank documents**:

- Flags `--minTriples` and `--maxTriples` filter data documents based on the number of unique triples they contain.
- Filtering on the average minimum and maximum degree (as well as *in* and *out* degree), e.g. `--minAvgDegree`
- Flag `--namespace` connects to the LOD Laundromat namespace index, and only returns documents using that particular namespace. This allows for coarse selectivity of domains. For instance datasets that are possibly relevant to the bioinformatics domain can be filtered based on the **drugbank** and **chebi** namespaces. The namespace flag accepts both full URIs and de-facto RDF prefixes⁹ that denote namespaces.
- Flag `--sparql` allows an arbitrarily complex SPARQL query to be evaluated against the LOD Laundromat backend. While not very user-friendly, this flag allows less often used selection criteria to be applied. Since we log SPARQL queries at the backend, we are able to add flags to *Frank* based on often requested queries.

⁷ See <http://index.lodlaundromat.org>

⁸ See <http://www.rdfhdt.org/>

⁹ Prefixes are taken from <http://prefix.cc>.

Data document are identified in the following two ways:

1. The URI from which the data document, cleaned by the LOD Laundromat, can be downloaded (`--downloadUri`). These clean data documents are disseminated by the LOD Laundromat as Gzipped N-Triples or N-Quads. The statements are unique within a document so no bookkeeping with respect to duplicate occurrences needs to be applied. Statements are returned according to their lexicographic order. These statements can be processed on a one-by-one basis which allows for streamed processing by *Frank*.
2. The Semantic Web resource identifier assigned by LOD Laundromat for this particular document (`--resourceUri`).

When neither `--downloadUri` nor `--resourceUri` are passed as arguments *Frank* returns both separated by a white-space.

The streaming nature of *Frank* enables combinations of streamed triple and document retrieval. The following command returns a stream of documents with an average out-degree of 15 that contain at least 100 unique RDF properties. The stream consists of N-Quads where every triple ends in a newline and within-triple newlines are escape according to the N-Quads standard. The graph name of each quadruple is the LOD Laundromat document identifier.

```
$ ./frank documents \
  --resourceUri \
  --minAvgOutDegree 15 \
  --sparql "?doc llm:metrics/llm:distinctProperties ?numProp.
           (FILTER ?numProp > 100)"
| ./frank statements --showGraph
```

4.3 Metadata

`frank meta` retrieves the metadata description of a given data document. It interfaces with the SPARQL endpoint of LOD Laundromat and returns N-Triples that contain provenance and structural properties for that particular document¹⁰.

These structural properties include:

- VoID description properties such as the number of triples, entities, and the number of used properties and classes
- Additional properties not included in VoID directly, such as the number of *defined* properties and classes, and the number of literals, IRIs, and blank nodes.
- Network properties such as degree, in degree and out degree. For each of these properties we present descriptive statistics including the minimum, maximum, median, mean and standard deviation.
- Details on the IRI and literal lengths, with similar descriptive statistics.

¹⁰ We present this metadata collection in more detail in [14].

Other than such structural properties, the LOD Laundromat metadata includes crawling provenance as well, such as:

- A reference to the original download location of the document
- Warnings and errors encountered when fetching and cleaning the document
- Number of duplicate triples
- Temporal information such as the last-modified date of the original file, or the cleaning date of a document.
- Other low-level information on the original file, such as the serialization format, its size, or its line count

5 Evaluation

To illustrate the use of the LOD Lab for evaluation purposes, we re-evaluate parts of three previously published papers. A paper presenting an efficient in-memory RDF dictionary (Section 5.1), a paper compressing RDF in a binary representations (Section 5.2), and a paper exploring Linked Data best practices (Section 5.3). We do not aim to completely reproduce these papers, as we merely intend to illustrate LOD Lab and how *Frank* can be used by others.

Below we discuss these papers in detail and highlight the parts of their experiment we reproduce. For these experiments we illustrate how we used *Frank*, and we present the reevaluated results. The source-code of these evaluations are publicly available¹¹

5.1 Paper 1: RDF Vault

‘A Compact In-Memory Dictionary for RDF data’ [1] is a recent paper from the 2015 Extended Semantic Web Conference, which presents RDF Vault. RDF Vault is an in-memory dictionary, which takes advantage of string similarities of IRIs, as many IRIs share the same prefix. The authors take inspiration from conventional Tries (tree structures for storing data), and optimize this method for RDF data.

The authors measure the average encoding time per entity (time it takes to *store* a string in RDF Vault), average decoding time per entity (time it takes to *get* this string), and the memory use. Additionally, the authors make a distinction between these measurements for literals and URIs, considering literals often lack a common prefix. In the original paper, RDF vault is compared against several baselines (e.g. a classical in-memory dictionary), and evaluated against the following 4 datasets: Freebase, the Billion Triple Challenge datasets, DBpedia and BioPortal.

We use *Frank* to re-evaluate the encoding time of RDF Vault (using the original implementation) against a larger number of datasets: for each document, we measure the average encoding time of literals, IRIs, and both combined. In

¹¹ See <https://github.com/LaurensRietveld/FrankEvaluations>

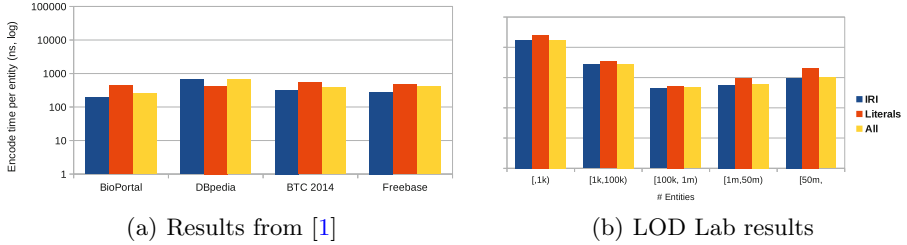


Fig. 3. Average encoding time per entity (ns)

order to compare these results meaningfully with the results from the original paper, we group the documents by number of entities, and present the encoding/decoding time for each group.

In figure 3 we present the original RDF vault results on the left side, and the results obtained via *Frank* on the right side. We collected the results from frank by piping all documents to the evaluation script as follows, where `./rdfVaultEncodeDocument.sh` is a Bash script that reads the *Frank* documents from the standard input, and applies RDF Vault for each of these documents.

```
$ ./frank documents --downloadUri | ./rdfVaultEncodeDocument.sh
```

Both figures show the average encoding time of IRIs, Literals, and both combined. Our results are based on 100,000 LOD Laundromat documents¹², where we grouped documents in buckets by the number of encoded entities. The original results differ between datasets: the average encoding time of IRIs in BioPortal are 1/3 of the DBpedia encoding times. Our results show the influence of the dataset size on the encoding times (particularly considering the y log scale). Smaller datasets of less than 1,000 entities may take up to 30,000 nano seconds per entity. Similarly, datasets with between 1,000 and 100,000 entities show longer encoding times than the original paper as well. For dataset sizes which correspond to the original paper, the results are similar. The re-evaluation of these results clearly show the effect of the dataset size on encoding times. That effect was not investigated in the original paper, because the experiments were only done on a handful of datasets. As we have shown, *Frank* trivially allows to run the original experiments on hundreds of thousands datasets, immediately giving an insight in the unexpected non-monotonic relation between dataset size and encoding time per entity.

Other structural dimensions might be relevant for this paper as well, such as the number of literals in a dataset or the standard deviation of URI or literal lengths. All these dimension are accessible using the LOD Laundromat metadata and the *Frank* interface. E.g., to run the vault experiments for dataset with a high standard deviation in URI lengths, run:

¹² Due to the runtime of RDF Vault and time constraints we were unable to re-evaluate this on the complete LOD Laundromat set.

```
$ ./frank documents \
  --downloadUri \
  --query "{?doc 11m:metrics/11m:IRILength/11m:std ?std .
           FILTER(?std > 50)}"
  | ./rdfVaultEncodeDocument.sh
```

5.2 Paper 2: RDF HDT

‘Binary RDF Representation for Publication and Exchange (HDT)’ [8] is an often cited paper (56 at the time of writing) from the journal of Web Semantics. HDT is a compact binary RDF representation which partitions RDF datasets in three components: Header information, a dictionary, and the actual triples structure. The important gain of HDT is that the HDT files are queryable *in their compressed form* using simple SPARQL triple patterns.

In the original paper, the performance of HDT is evaluated by measuring the compression ratio of HDT compared to other compression algorithms (e.g. Gzip and Bzip2), the compression time, and by measuring the number of entries in the dictionary compared to the total number of triples. The datasets used in this evaluation are Geonames, Wikipedia, DBTune, Uniprot and DBpedia-en. A part of the evaluation is evaluated against the 400 largest datasets in the Billion Triple Challenge (BTC). This is a fairly complete evaluation, considering the number of datasets, and the use of BTC datasets.

The results we re-evaluate¹³ are the compression ratios presented in [8] which were evaluated on Uniprot datasets from different sizes (1, 5, 10, 20, 30 and 40 million triples). We re-evaluate this particular research result using *Frank* by finding dataset of similar sizes ($\pm 10\%$) and by measuring the compression ratio.

The LOD Laundromat documents are fetched using *Frank* and filtered to match the Uniprot dataset sizes. E.g., to select LOD Laundromat documents matching the 1 million Uniprot dataset, *Frank* searches for documents of 1 million with a deviation of 10%, and streams these document to a shell script which downloads and compresses these documents using HDT.

```
$ ./frank documents --minTriples 950000 --maxTriples 1050000
  | ./hdtCompressDocument.sh
```

Table 1 shows the compression ratio for Uniprot datasets on the left side, and the average compression ratio for LOD Laundromat documents on the right side. There is a large difference between Uniprot and the LOD Laundromat datasets in both compression ratio and average document size. Another interesting observation is the high average compression ratio of LOD Laundromat documents around 1 million, compared to other LOD Laundromat documents.

To better understand such differences, we use *Frank* to evaluate RDF HDT along another dimension: the average degree of documents. We did so by searching for three buckets of datasets. Those with a low (1-5), medium (5-10) and high (10+) average degree, all with at least 1 million triples:

¹³ We re-evaluated the latest HDT version accessible at <https://github.com/rdfhdt/>

Table 1. HDT Compression rates: Results from [8] on Uniprot (left side) vs. results from *Frank* (right side)

| Triples (millions) | Original: Uniprot | | | LOD Lab | | |
|-----------------------|-------------------|--------------|----------------------|---------|-------------------|--------------------------|
| | # docs | Size (MB) | Compression Ratio | # docs | Avg. Size (MB) | Avg Compression Ratio |
| 1 | 1 | 89.07 | 3.73% | 179 | 183.31 | 11.23% |
| 5 | 1 | 444.71 | 3.48% | 74 | 799.98 | 4.99% |
| 10 | 1 | 893.39 | 3.27% | 50 | 1,642.60 | 5.43% |
| 20 | 1 | 1,790.41 | 3.31% | 17 | 3,328.57 | 4.15% |
| 30 | 1 | 2,680.51 | 3.27% | 19 | 4,880.26 | 5.09% |
| 40 | 1 | 3,574.59 | 3.26% | 8 | 6,586.95 | 7.25% |

Table 2. HDT Compression rates grouped by avg degree

| Avg. Degree | # docs | Compression Ratio |
|-------------|--------|----------------------|
| 1-5 | 92 | 21.68% |
| 5-10 | 80 | 6.67% |
| 10-∞ | 99 | 4.85% |

```
$ ./frank documents --minAvgDegree 5 --maxAvgDegree 10 --minTriples 1000000
| ./hdtCompressDocument.sh
```

The results (See Table 2) show that an increase in degree of a document comes with a decrease in compression ratio.

These experimentation on a large numbers of datasets across a large number of dimensions is made easy by *Frank*, and allows researchers to both tune their algorithms to different document characteristics, as well as better understand their algorithms behavior under different conditions.

5.3 Paper 3: Linked Data Best Practices

Other than using the LOD Lab for empirical evaluations, we show how it can be used for explorative and observational papers as well. The most cited paper of the International Semantic Web Conference 2014 is ‘Adoption of the Linked Data Best Practices in Different Topical Domains’ [16], where the authors analyze Linked Data best practices by crawling (using LDspider [13]) the LOD Cloud. Seed items for this crawl come from public catalogs, the Billion Triple Challenge, and datasets advertised on public LOD mailing lists. The crawl included 900,129 documents (URIs that were dereferenced) and 8,038,396 resources. Documents are grouped to 1014 datasets using information from catalogs, or Pay-Level-Domain (PLD) otherwise. The paper present a large and diverse set of statistics, including:

1. The number of resource per document
2. Dataset grouped by topical domain. These domains are fetched from online catalogs if any, and manually annotated otherwise
3. Indegree and outdegree of datasets
4. The links occurring between datasets, and the type of predicates used for linking
5. The use of vocabularies in datasets

The crawling mechanism behind these statistics strongly relies on dereferenceable URIs. As a consequence, there is a strong link between a crawled document and the URI it is crawled from: we know which URI is the ‘authority’ for a document. This offers opportunities for e.g. grouping the datasets by PLD and finding links between datasets. This crawling mechanism differs from the LOD Laundromat, which mostly consists of (often compressed) data dumps¹⁴. As a consequence, in LOD Laundromat, the URL <http://data.dws.informatik.uni-mannheim.de/dbpedia/2014/en/> (the official DBpedia download location) does not directly match with <http://dbpedia.org/resource/Amsterdam>, making it difficult to know the authoritative-ness of the download dump URI. I.e., the LOD Laundromat crawls many more documents and triples (including those not accessible as dereferenceable URI), but lacks information on the authoritative-ness of URIs. Vice versa, the used crawl in [16] crawls only a fraction of the LOD Laundromat size, but retains the notion of authority. As a result, the original paper has statistics on DBpedia as a whole, where the LOD Lab results are separate for each independent DBpedia data dump.

These differences in features between both crawling mechanisms restricts the ability of *Frank* to reproduce all of the statistics from [16]. However, we chose to focus on re-evaluating the used vocabularies on the LOD Cloud, which does not suffer from these difference in crawling mechanisms. Instead, *Frank* offers a more complete perspective on the use of vocabularies, considering the number of crawled triples.

We reproduced this experiment by simply streaming all the LOD Laundromat download URIs to a script counting the namespaces¹⁵:

```
$ ./frank documents --downloadUri | ./countNamespacesForDocument.sh
```

Table 3 shows the 10 most frequent occurring namespaces in documents. In the original paper these counts are grouped by dataset (i.e. groups of documents), where we present these statistics on a document level alone.

This table shows striking differences: where the *time* namespace is used in 68.20% of the LOD Laundromat documents, it does not occur in the top 10 list of [16]. Similarly, the *cube* namespace occurs in 23.92% of LOD Laundromat documents, and is missing from the original top 10 list as well.

The crawling method behind both approaches, and the method used by [16] to group documents as datasets can explain these discrepancies. Therefore, we

¹⁴ See [3] for more information.

¹⁵ Using the namespace list of <http://prefix.cc>, similar to the original paper.

Table 3. Top 10 namespaces used in documents

| Prefix | Original [16] | | Prefix | LOD Lab | |
|--------|---------------|------------|-----------|---------|--------|
| | #datasets | % datasets | | #docs | % docs |
| rdf | 996 | 98.22% | rdf | 639,575 | 98.40% |
| rdfs | 736 | 72.58% | time | 443,222 | 68.19% |
| foaf | 701 | 69.13% | cube | 155,460 | 23.92% |
| dcterm | 568 | 56.01% | sdmxdim | 154,940 | 23.84% |
| owl | 370 | 36.49% | worldbank | 147,362 | 22.67% |
| wgs84 | 254 | 25.05% | interval | 69,270 | 10.66% |
| sioc | 179 | 17.65% | rdfs | 30,422 | 4.68% |
| admin | 157 | 15.48% | dcterms | 26,368 | 4.06% |
| skos | 143 | 14.11% | foaf | 20,468 | 3.15% |
| void | 137 | 13.51% | dc | 14,423 | 2.22% |

do not claim to have the right answer for these kind of statistics. Instead, we show that the LOD Lab approach allows for large scale comparisons for these kinds of Linked Data observational studies.

6 Conclusion

The distributed nature of the Semantic Web, the wide range of serialization formats, and the idiosyncrasies found in datasets, make it difficult to use the Semantic Web as a true large-scale evaluation platform. As a consequence, most research papers are only evaluated against a handful of datasets.

In this paper we presented LOD Lab, a new way of conducting Linked Data experiments that incorporates both volume and variety while at the same time allowing the set of considered data documents to be limited according to domain-specific and/or structural constraints. This is achieved by using the LOD Laundromat backend together with the simple yet versatile programming interface *Frank* that allows large-scale Linked Data evaluations to be run from the command-line.

The viability of the LOD Lab approach was demonstrated by scaling up three experiments reported in recent Semantic Web conference publications. These re-evaluations show that evaluations over Linked Data can now be performed without the human effort having to increase linearly in terms of the number of datasets involved. In addition, the re-evaluations show that the combination of volume, variety and selectivity facilitates a more detailed analysis of Semantic Web algorithms and approaches by relating evaluation outcomes to properties of the data.

References

1. Bazoobandi, H.R., de Rooij, S., Urbani, J., ten Teije, A., van Harmelen, F., Bal, H.: A compact in-memory dictionary for RDF data. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 205–220. Springer, Heidelberg (2015)
2. Beek, W., Rietveld, L.: Frank: The lod cloud at your fingertips. In: Developers Workshop, ESWC (2015)
3. Beek, W., Rietveld, L., Bazoobandi, H.R., Wielemaker, J., Schlobach, S.: LOD laundromat: a uniform way of publishing other people's dirty data. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 213–228. Springer, Heidelberg (2014)
4. Bizer, C., Schultz, A.: The berlin sparql benchmark (2009)
5. Boncz, P., Fundulaki, I., Gubichev, A., Larriba-Pey, J., Neumann, T.: The linked data benchmark council project. *Datenbank-Spektrum* **13**(2), 121–129 (2013)
6. Buil-Aranda, C., Hogan, A., Umbrich, J., Vandenbussche, P.-Y.: SPARQL web-querying infrastructure: ready for action? In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 277–293. Springer, Heidelberg (2013)
7. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: six years of experience. In: Spaccapietra, S. (ed.) *Journal on Data Semantics XV*. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011)
8. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary rdf representation for publication and exchange (hdt). *Web Semantics: Science, Services and Agents on the World Wide Web* **19**, 22–41 (2013)
9. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web* **3**(2), 158–182 (2005)
10. Harris, S., Seaborne, A.: SPARQL 1.1 query language, March 2013
11. Harth, A.: Billion Triples Challenge data set. Downloaded from (2012). <http://km.aifb.kit.edu/projects/btc-2012/>
12. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An Empirical Survey of Linked Data Conformance. *Web Semantics: Science, Services and Agents on the World Wide Web* **14**, 14–44 (2012)
13. Isele, R., Umbrich, J., Bizer, C., Harth, A.: Ldspider: an open-source crawling framework for the web of linked data. In: 9th International Semantic Web Conference (ISWC 2010). Citeseer (2010)
14. Rietveld, L., Beek, W., Schlobach, S.: LOD in a box: The C-LOD meta-dataset (Under submission). <http://www.semantic-web-journal.net/system/files/swj868.pdf>
15. Rietveld, L., Verborgh, R., Beek, W., Vander Sande, M., Schlobach, S.: Linked data-as-a-service: the semantic web redeployed. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 471–487. Springer, Heidelberg (2015)

16. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
17. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: Sp 2 bench: A sparql performance benchmark, icde, Shanghai, China (2009)
18. Schmidt, M., Görlitz, O., Haase, P., Ladwig, G., Schwarte, A., Tran, T.: FedBench: a benchmark suite for federated semantic data query processing. 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. 585–600. Springer, Heidelberg (2011)
19. Verborgh, R., et al.: Querying datasets on the web with high availability. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 180–196. Springer, Heidelberg (2014)

Strategies for Efficiently Keeping Local Linked Open Data Caches Up-To-Date

Renata Dividino¹(✉), Thomas Gottron¹, and Ansgar Scherp²

¹ WeST – Institute for Web Science and Technologies,
University of Koblenz-Landau, Koblenz, Germany
dividino@uni-koblenz.de

² Kiel University and Leibniz Information Center for Economics, Kiel, Germany
asc@informatik.uni-kiel.de

Abstract. Quite often, Linked Open Data (LOD) applications pre-fetch data from the Web and store local copies of it in a cache for faster access at runtime. Yet, recent investigations have shown that data published and interlinked on the LOD cloud is subject to frequent changes. As the data in the cloud changes, local copies of the data need to be updated. However, due to limitations of the available computational resources (e.g., network bandwidth for fetching data, computation time) LOD applications may not be able to permanently visit all of the LOD sources at brief intervals in order to check for changes. These limitations imply the need to prioritize which data sources should be considered first for retrieving their data and synchronizing the local copy with the original data. In order to make best use of the resources available, it is vital to choose a good scheduling strategy to know when to fetch data of which data source. In this paper, we investigate different strategies proposed in the literature and evaluate them on a large-scale LOD dataset that is obtained from the LOD cloud by weekly crawls over the course of three years. We investigate two different setups: (i) in the single step setup, we evaluate the quality of update strategies for a single and isolated update of a local data cache, while (ii) the iterative progression setup involves measuring the quality of the local data cache when considering iterative updates over a longer period of time. Our evaluation indicates the effectiveness of each strategy for updating local copies of LOD sources, i.e., we demonstrate for given limitations of bandwidth, the strategies' performance in terms of data accuracy and freshness. The evaluation shows that the measures capturing change behavior of LOD sources over time are most suitable for conducting updates.

1 Introduction

Quite often, LOD applications pre-fetch data from the Web and store local copies of it in a cache or build an index over it to speed up access and search. Recent investigations [1, 2, 10, 11, 15, 22] have shown that data published and interlinked on the LOD cloud is subject to frequent changes. As the data in the cloud changes, these caches or indices no longer reflect the current state of the data

anymore and need to be updated. Käfer et al. [17] observed a subset of the LOD cloud over a period of 29 weeks and concluded (among others) that the data of 49.1% of the LOD sources changes. Likewise, Gottron et al. [12] observed LOD data over a period of 77 weeks and described that the accuracy of indices built over the LOD sources drops by 50% after already 10 weeks. These outcomes indicate that almost half of the LOD sources are not appropriate for long-term caching or indexing. Unquestionably, data on the LOD cloud changes and knowledge about these changes, i. e., about the change behavior of a dataset over time, is important as it affects various different LOD applications such as indexing of distributed data sources [18], searching in large graph databases [13], search optimization [19], efficient caching [8, 14, 24], and recommending vocabularies to Linked Data engineers [21].

LOD applications relying on data from the LOD cloud need to cope with constant data updates to be able to guarantee a certain level of quality of service. In an ideal setting, a cache or an index is kept up-to-date by continuously visiting all data sources, fetching the most recent version of the data and synchronizing the local copies with it. However, in real world scenarios LOD applications must deal with limitations of the available computational resources (e.g., network bandwidth, computation time) when fetching data from the LOD cloud. These limitations imply the necessity to prioritize which data sources should be first considered for retrieving their data. In order to make best use of the resources available, it is vital to choose a good scheduling strategy for updating local copies of the LOD data sources. While there exists research on the freshness analysis of the cached data for answering SPARQL queries in a hybrid approach such as Umbrich et al [24], to the best of our knowledge, there is no work addressing strategies to efficiently keep local copies of LOD source up-to-date.

Intuitively, a strategy dedicated to update data caches build out of data from the LOD cloud would make use of the HTTP protocol. The *Last-Modified* HTTP header field denotes when a LOD source behind this URI has been changed last. However, in a previous investigation [9], we showed that only very few LOD sources (on average only 8%) provide correct update values. Consequently, applications relying on such information are susceptible to draw wrong conclusions. Thus, this method is inappropriate for probing a LOD source for whether or not it has been changed since the last retrieval of its data. The only alternative is to actually retrieve the data from the sources and check it for changes.

In this paper, we consider update scheduling strategies for maintaining indices of web documents and metrics initially developed to capture the data changes in LOD sources and analyze their effectiveness for updating local copies of LOD sources. Scheduling strategies aim for deriving an order for data sources to when they should be visited. Consequently, the application updates its local copy by fetching data from the data sources following this order. The simplest strategy is to visit the data sources in an arbitrary but fixed order, which guarantees that the local copy of every data source is updated after a constant interval of time. Alternative strategies explore the different features provided by the data sources, e. g., their size, to assign an importance score to each data source, and

thus deriving an order. An established scheduling strategy for the Web leverages the PageRank algorithm [20], where a score of importance is given to each data source regarding its centrality in the link network with other data sources.

When considering a set of LOD sources, certainly some of them change more or less often than others [17]. For example, it is not likely that in a short time interval every LOD source changes. Thus, many sources may provide the same information during this entire interval. Therefore, it is not necessary to fetch data from such sources. However, whenever data of a source changes, an update is required. Accordingly, some sources should be fetched at shorter/longer time-intervals. This implies that each LOD source could be given a different update importance, which is based on their change behaviour. We consider change metrics for LOD data sources presented in [11]. These metrics measure the change rate of a dataset based on the changes that have taken place between two points in time. Furthermore, in previous work [10], we propose the notion of dynamics of LOD datasets. The dynamics function measures the accumulated changes that have occurred within a data set in a given time interval. Even though these metrics were not directly proposed to support scheduling strategies for data updates, measures that capture the change rate or dynamics of a LOD dataset may indeed be used for conducting updates.

We evaluate different strategies on a large-scale LOD dataset from the Dynamic Linked Data Observatory (DyLDO) [17] that is obtained via 149 weekly crawls in the period from May 2012 until March 2015. We investigate two different setups: (i) in the *single step* setup we evaluate the quality of update strategies for a single and isolated update of a local data cache, while (ii) the *iterative progression* setup involves measuring the quality of the local data cache when considering iterative updates over a longer period of time. Quality is measured in terms of precision and recall with respect to the gold standard, i. e., we check the correctness of data of the (updated) local copy with respect to the data actually contained in the LOD cloud. We assume that only a certain bandwidth for fetching data from the cloud is available, and we investigate the effectiveness of each strategy for different bandwidths. Therefore, in the first setup, we can observe the relation between strategies and restrictions of bandwidth (i. e., if the strategies show comparatively uniform performance over all restrictions or if better/worse performance depends on a given restriction), and use such findings as parameters for the second setup. The second setup evaluates the behavior of the strategies in a realistic scenario (e. g., a LOD search engine updating its caches). Our evaluation indicates the most effective strategies for updating local copies of LOD sources, i. e., we demonstrate for given restrictions of bandwidth, which strategy performs better in terms of data accuracy and freshness.

2 Foundations

Linked Data that is crawled from the cloud can be represented in the form of N-Quads¹. Technically, a quad (s, p, o, c) consists of an RDF triple where s , p , and o

¹ W3C Recommendation <http://www.w3.org/TR/n-quads/>

correspond to the subject, predicate and object and the context c , i. e., the data source on the Web where this RDF triple was retrieved.

We assume that data from the various LOD sources is retrieved at some fix point in time t . We consider that an application visits and fetches data of LOD sources at a regular interval (say, once a week). Consequently, a LOD data source is defined by a context c and the data it provides at points in time t , i. e., the set of RDF quads $X_{c,t}$. Furthermore, we denote the size of a data set with $|X_{c,t}|$ to indicate the number of triples contained in the data set at context c at the point in time t .

In this paper, we rely on local copies of the LOD sources. Such a copy typically covers several data sources. Given a set $C = \{c_1, c_2, \dots, c_m\}$ of contexts of interest, we can define the overall dataset as:

Definition 1. *Dataset*

$$X_t = \bigcup_{c \in C} X_{c,t} \quad (1)$$

Example 1. As a matter of example, we consider that data comes from three different data sources: *dbpedia.org*, *bbc.co.uk*, and *musicbrainz.com*, and data is retrieved at two different points in time, on May 8th, 2013 and on June 10th, 2013.

$$\begin{aligned} X_{2013-05-08} &= \{X_{\text{dbpedia.org},2013-05-08}, X_{\text{musicbrainz.com},2013-05-08}, X_{\text{bbc.co.uk},2013-05-08}\}, \\ X_{2013-06-10} &= \{X_{\text{dbpedia.org},2013-06-10}, X_{\text{musicbrainz.com},2013-06-10}, X_{\text{bbc.co.uk},2013-06-10}\} \end{aligned}$$

Finally, for distinct points t_1, t_2, \dots, t_n in time, we define a series of datasets over time:

Definition 2. *Series of Datasets*

$$\mathcal{X} = (X_{t_1}, X_{t_2}, \dots, X_{t_n}) \quad (2)$$

Example 2. Our example dataset is composed by data retrieved at two different points in time (see Example 1) such that $\mathcal{X} = (X_{2013-05-08}, X_{2013-06-10})$.

3 Update Scheduling Strategies

Due to limitations such as bandwidth restrictions and the frequent data changes in the LOD cloud, LOD applications relying on data from the LOD cloud need to prioritize which data sources should be first considered in order to achieve an optimal accuracy of their local copies under the given constraints. Therefore, applications make use of a scheduling strategy for data updates. A scheduling strategy aims for deriving an order of importance for data sources based on a set of data features. In the ideal case, a strategy would derive an order such that the application would visit only the subset of LOD sources which have actually been changed. In this section, we introduce a formal specification of update functions and a set of data features used by update strategies for deriving such an order.

3.1 Data Updates

Whenever an application needs to update a local copy covering the data sources in $c \in C$, at time t_{i+1} , it would technically be sufficient to fetch the complete dataset $X_{t_{i+1}}$. However, this would imply to visit all data sources c , retrieve their most recent version of the data $X_{c,t_{i+1}}$ and integrate it into one dataset $X_{t_{i+1}}$. Due to limitations such as network bandwidth capacity for downloading data or computation time, we assume that only a certain fraction of the data can actually be retrieved fresh from the cloud and processed in a certain time interval.

Thus, applications need to apply a scheduling strategy to efficiently manage the accuracy of the data. Based on features extracted from the dataset retrieved at an earlier point in time t_i , a scheduling strategy indicates which data sources c should be visited (i. e., visit the URI c and fetch the latest version of the data made available at this URI) in the time slice between t_i and t_{i+1} . The update strategy can simply be seen as a relation:

Definition 3. *Update Strategy*

$$U \subset C \times \{1, \dots, n\} \quad (3)$$

A tuple (c, i) in this relation indicates a point in time t_i at which data from a data source c should be updated.

Furthermore, we define the constraint of the bandwidth as a restriction to download at most up to K triples.

Definition 4. *Constraint of the Bandwidth*

$$\text{For a given } i : \sum_{(c,i) \in U} |X_{c,t_i}| \leq K \quad (4)$$

For any given constraint of the bandwidth, it is possible to retrieve data from the sources in their order of preferences until the limit has been reached.

Example 3. Suppose our dataset has been updated the last time on May 8th, 2013 (see Example 1), and we want to again update our local copy on June 10th, 2013. However, due to limitations, the constraints of the bandwidth enables only $K = 12,000$ triples to be fetched per time slice. For such constraints, we suppose we cannot fetch all the data since $|X_{\text{dbpedia.org},2013-05-08}| + |X_{\text{musicbrainz.com},2013-05-08}| + |X_{\text{bbc.co.uk},2013-05-08}| \geq 12,000$. Nevertheless, without violating these restrictions, we suppose we can entirely fetch data from the first two data sources: *dbpedia.org* and *musicbrainz.com*.

We define a *last update* function lu to identify for a specific data source and a given point in time when its data was updated last:

Definition 5. *Last Update Function*

$$lu(c, i) = \operatorname{argmax}_{j \leq i} \{(c, j) \in U\} \quad (5)$$

This function can be used recursively to identify, for instance, the update prior to the last update by $lu(c, lu(c, i) - 1)$.

Example 4. In the previous example, we updated our local copy by fetching data from *dbpedia.org* and *musicbrainz.com*, then the *last update* time of the data sources are given as:

$$\begin{aligned} lu(dbpedia.org, 2013-06-10) &= lu(musicbrainz.com, 2013-06-10) = 2013-06-10, \\ lu(bbc.co.uk, 2013-06-10) &= 2013-05-08 \end{aligned}$$

Using the last update function lu at time t_i , we can define the aggregated data set according to an update strategy, i. e., which version of which data source is part of the current local copy. This aggregated data set X'_{t_i} is defined as:

Definition 6. *Aggregated Data Set*

$$X'_{t_i} = \bigcup_{c \in C} X_{c, t_{lu(c, i)}} \quad (6)$$

Example 5. Following Example 4, our updated dataset for June 10th, 2013 is given as: $X'_{2013-06-10} = \{X_{dbpedia.org, 2013-06-10}, X_{bbc.co.uk, 2013-05-08}, X_{musicbrainz.com, 2013-06-10}\}$

Finally, using this notation, we can easily construct the history of a particular data source in the course of execution of an update plan over time up to time t_i :

Definition 7.

$$\mathcal{H}(c, t_i) = \{X_{c, t_j} \mid (c, j) \in U, t_j \leq t_i\} \quad (7)$$

Example 6. The history of our sample data source *dbpedia.org* is given as:

$$\mathcal{H}(dbpedia.org, 2013-06-10) = \{X_{dbpedia.org, 2013-06-10}, X_{dbpedia.org, 2013-05-08}\}$$

3.2 Data Features

In the following, we present features proposed in the literature to improve the freshness of cached data. Here, we restrict ourselves to define only those features that will actually be used in our experiments. Please note that the set of features of a data source can always be extended.

Age provides the time span since the data source has been last visited and updated [4]. It captures 'how old' is the data provided by a data source:

$$f_{Age}(c, X'_{t_i}) = t_i - t_{lu(c, i)} \quad (8)$$

PageRank provides the PageRank of a data source in the overall data set at the (last known) time [20]:

$$f_{PageRank}(c, X'_{t_i}) = PR(X_{c,t_{lu(c,i)}}) \tag{9}$$

Size provides the (last known) number of triples provided by a data source:

$$f_{Size}(c, X'_{t_i}) = |X_{c,t_{lu(c,i)}}| \tag{10}$$

ChangeRatio provides the absolute number of changes of the data in a data source between the last two (known) observation points in time [7].

$$f_{Ratio}(c, X'_{t_i}) = |X_{c,t_{lu(c,i)}} \setminus X_{c,t_{lu(c,lu(c,i)-1)}}| + |X_{c,t_{lu(c,lu(c,i)-1)}} \setminus X_{c,t_{lu(c,i)}}| \tag{11}$$

ChangeRate provides the change rate between the observed data in the two (last known) points in time of a data source [11].

$$f_{Change}(c, X'_{t_i}) = \Delta(X_{c,t_{lu(c,i)}}, X_{c,t_{lu(c,lu(c,i)-1)}}) \tag{12}$$

In this case, the change rate Δ is a function (metric) to measure the change rate between two data sets. We will use two Δ functions:

Jaccard distance:

$$\Delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}}) = 1 - \frac{|(X_{c,t_{lu(c,lu(c,i)-1)}}) \cap (X_{c,t_{lu(c,i)}})|}{|(X_{c,t_{lu(c,lu(c,i)-1)}}) \cup (X_{c,t_{lu(c,i)}})|}$$

Dice Coefficient:

$$\Delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}}) = 1 - \frac{2 * |(X_{c,t_{lu(c,lu(c,i)-1)}}) \cap (X_{c,t_{lu(c,i)}})|}{|(X_{c,t_{lu(c,lu(c,i)-1)}})| + |(X_{c,t_{lu(c,i)}})|}$$

Dynamics measures the behavior of the data source observed over several points in time [10], where the dynamics of a data source is defined as the aggregation of absolute changes, as provided by Δ -metrics.

$$f_{Dynamic}(c, X'_{t_i}) = \sum_{i=0}^j \frac{\Delta(X_{c,t_{lu(c,lu(c,i)-1)}}, X_{c,t_{lu(c,i)}})}{t_{lu(c,i)}, t_{lu(c,lu(c,i)-1)}}, j \leq i.$$

3.3 Update Function

As a large number of LOD sources are available but only a limited number of sources can be fetched per run, it is required to determine which sources should be visited first. By using the vector of features of each data source, we define an update function $\rho : f \rightarrow R$, which assigns a preference score to a data source based on the observed features at time t_i .

An update strategy is defined by ranking the data sources according to their preference score in descending or ascending order, and fetching them starting

from the top ranked entry to some lower ranked entry. For instance, if we consider f_{Size} to be the feature observed at time t_i for all $c \in C$, ρ could be defined as the rank of the data sources in ascending order (from the smallest to the biggest ones).

Furthermore, the bandwidth defines the amount of data that can be fetched per run. Consequently, at some point in time t_i , data of a set of data sources is updated until the bandwidth constraint has been consumed completely. For the sake of clarity, we discard a data source when its data cannot completely be fetched, i. e., when the last started fetch operation cannot be entirely executed because of the bandwidth limit being violated while reading the data. We consider that the tuple (c, i) is considered to be in the update relation U defined above, if the data from c can be entirely fetched based on the given order and the available bandwidth. Without loss of generality, we assume that the data sources are visited in a sequential order. However, it is left to the implementation to decide whether data from the different data sources should be fetched in sequential or parallel processing.

4 Evaluation

In this section, we consider the scheduling strategies described in Section 3 and analyze their effectiveness for updating local copies of the LOD sources. We evaluate these strategies on a large-scale and real world LOD dataset. Our evaluation goal is to show which of the update strategies produce better updates of the LOD sources, i. e., we demonstrate for given restrictions of bandwidth, which strategy performs better in terms of data accuracy and freshness.

4.1 Data

Our evaluation dataset is obtained from the Dynamic Linked Data Observatory (DyLDO). The DyLDO dataset has been created to monitor a fixed set of Linked Data documents (and their neighborhood) on a weekly basis². Our evaluation dataset is composed of 149 weekly crawls (in the following we will refer to a crawl as a snapshot) corresponding to a period over the last three years (from May 2012 to March 2015). Furthermore, the DyLDO dataset contains various well known and large LOD sources, e. g., dbpedia.com, musicbrainz.com, and bbc.co.uk as well as less commonly known ones, e. g., advogato.org, statistics.data.gov.uk, and uefa.status.net. For more detailed information about the DyLDO dataset, we refer the reader to [17]. As we use weekly crawls obtained from the DyLDO dataset, we are only able to grab changes occurring between consecutive weeks (e. g., daily changes are not considered).

To gain a better insight into our evaluation dataset, let us first look at the evolution of the snapshots. The number of data sources per snapshot ranges between 465 and 742. On average, a snapshot is composed of 590 data sources.

² For sake of consistency, we use only the kernel seeds of LOD documents.

During the period studied, the number of data sources per snapshot slightly decreased, due to data sources going temporarily or permanently offline. Looking at consecutive snapshots, on average 1.05% of the data sources per snapshot are new and previously unseen (data sources birth rate), and 1.36% of the data sources disappear each week (death rate). On average, 99.3% of the data sources remain in existence between consecutive snapshots, and 37.3% of them change on a weekly basis. Taking the first snapshot as reference, only 13.9% of the data sources remain unchanged over the entire interval studied. This overview confirms prior findings [17] indicating that a high portion of the data on the LOD cloud changes.

To provide better insights into how changes are distributed over the data sources, we randomly sampled an arbitrary point in time (June 1st, 2014) and check for the distribution of triples over data sources. We observe that most of the data sources, 78.3%, are small (containing less than 1,000 triples) and they contribute only 0.5% of all triples retrieved at this point in time. The few big data sources (0.6%) that are left (up to 1,000,000 triples), contribute more than 49.2% of all triples. Furthermore, we observe that most of the changes (66.7%) take place in the data sources with more than 1,000,000 triples.

4.2 Evaluation Methodology

Ideally, scheduling strategies should prioritize an update of data sources which provide modified data. Note, that we do not consider the task of discovering new data sources for inclusion into the data cache. Rather, we want to maintain an as fresh-as-possible local copy of a fixed predefined set of LOD sources. To be able to evaluate different scheduling strategies, we use the following scenarios:

Single-Step. We evaluate the quality of update strategies for a single and isolated update of a local data cache, i. e., starting from a perfectly accurate data cache at time t_i , our goal is to measure which quality can be achieved with different update strategies at time t_{i+1} , for varying settings of bandwidth limitations.

Iterative Progression. We evaluate the evolution of the quality of a local data cache when considering iterative updates over a longer period of time, i. e., starting from a perfect data cache at time t_i , our goal is to measure how good is an update strategy in maintaining an accurate local copy at subsequent points in time $t_{i+1}, t_{i+2}, \dots, t_{i+n}$ when assuming a fixed bandwidth. In our experiment, we consider four iterations.

We implemented update strategies based on rankings according to the features presented in Section 3.2:

1. *Age* updates from the last to the most recently updated data source.
2. *Size-SmallestFirst* updates from the smallest to the biggest data source.
3. *Size-BiggestFirst* updates from the biggest to the smallest data source.
4. *PageRank* updates from the highest to lowest PageRank of a data source.

5. *ChangeRatio* updates from the most to the least changed data source based on set difference applied to the last two retrieved versions of the data.
6. *ChangeRate-J* updates from the most to the least changed data source based on Jaccard distance applied to the last two retrieved versions of the data.
7. *ChangeRate-D* updates from the most to the least changed data source based on Dice Coefficient applied to the last two retrieved versions of the data.
8. *Dynamics-J* updates from the most to the least dynamic data source based on Jaccard distance and previous observed snapshots of the data.
9. *Dynamics-D* updates from the most to the least dynamic data source based on Dice Coefficient and previous observed snapshots of the data.

Please note that we analyze the strategy *Age* only for the *Iterative Progression* scenario. *Age* cannot be used in the *Single Step* scenario. Since we build the follow-up copy from a perfect local copy, the feature *Age* would assign the same value to each data source.

The data features used by the strategies are extracted based on the available history information. In our experiment, the history is composed of the last four updates. In the first setup, the task to be accomplished by the strategies is to compute an update order for all data sources at the point in time t_{i+1} . For the strategies *Size* and *PageRank*, we use information about data retrieved from the last update time t_i . *ChangeRatio* and *ChangeRate* are calculated over the last two updates t_{i-1} and t_i , and *Dynamics* is calculated over the complete history for points in time t_{i-4} to t_i . For the *Iterative Progression* setup, we start with a perfect data cache at t_i . The task is to compute the updates iteratively at the next points in time t_{i+1} to t_{i+4} . In the first step, the history setup is the same as the *single-step* setup and the size of the history increases along with the iterations.

In order to make the results of the different setups comparable, and due to the fact that the iterative setup considers four iterative updates, the snapshots used in the single step evaluation are the same ones which are evaluated in the first place in the iterative evaluation setup (every fifth snapshot of the dataset). Additionally, we simulate network constraints by limiting the relative bandwidth, i. e., that only a certain ratio of triples can be fetched for updating a local copy at a given point in time. In the simulation, we stepwise increase the bandwidth constraint from 0% to 5% in intervals of 1%, from 5% to 20% in intervals of 5%, and from 20% to 100% in intervals of 20% of all available triples.

LOD sources are from time to time unavailable, i. e., some LOD sources cannot be reached by any application at a certain point in time, but may be again reachable at a later point in time. Nevertheless, the implemented strategies do not differentiate whether a source is unavailable for a period of time, or is deleted from the cloud. Whenever a LOD source is deleted or unavailable at point in time t_i , no triples are delivered and the empty set is considered for further computations.

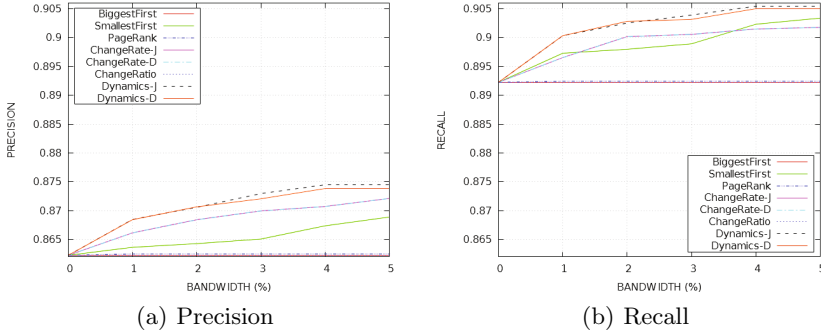


Fig. 1. Quality Outcomes for the Single Step Setup at Low Bandwidth Level (5%).

4.3 Metrics

The quality of an update strategy is measured in terms of micro average recall and precision over the gold standard, i.e. the perfect up-to-date local copy:

$$p_{micro}(X'_t, X_t) = \frac{\sum_{c \in C^{X'_t}} |X_{c,t} \cap X'_{c,t}|}{\sum_{c \in C^{X'_t}} |X'_{c,t}|} \tag{13}$$

$$r_{micro}(X'_t, X_t) = \frac{\sum_{c \in C^{X'_t}} |X_{c,t} \cap X'_{c,t}|}{\sum_{c \in C^{X'_t}} |X_{c,t}|} \tag{14}$$

4.4 Results

Single-Step Evaluation. Figure 2(a) and Figure 2(b) show the average precision and recall over all snapshots the *single-step* setup. The x -axis represents the different levels of constraints of relative bandwidth (in percent) and the quality in terms of precision and recall is placed on the y -axis. We observe that precision ranges from 0.862 to 1 and recall from 0.892 to 1 for bandwidth from 0% to 100% (see Figure 2(a) and Figure 2(b)). This implies that if no updates are executed (no bandwidth is available), our dataset is on average 87% correct (F measure) after one update. This value can be interpreted as the probability to get correct results when issuing an arbitrary query on the data.

Overall, the *Dynamics* strategies outperform all other strategies. First, we look at the precision curve. For very low relative (see Figure 1(a)) bandwidth (from 0% to 10%) the *Dynamics* strategies perform best, followed by the *ChangeRate* strategies. Already only with 3% available bandwidth, precision improvements is from 0.862 to 0.873 for *Dynamics*, and to 0.869 for *ChangeRate*. With 10% bandwidth the improvement gets to 0.888 for *Dynamics* and 0.879 for *ChangeRate* while the third best strategy, *SmallestFirst*, achieves 0.877 and the others strategies do not achieve scores higher than 0.862. For the higher relative bandwidths, the *ChangeRate* and *Dynamics* strategies are comparable and

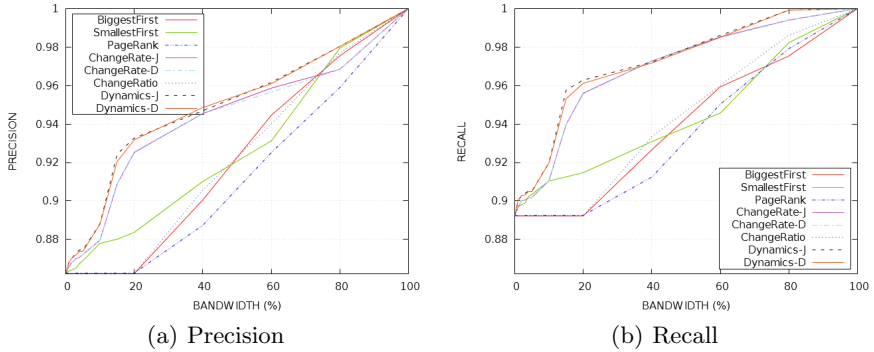


Fig. 2. Quality Outcomes for the Single Step Setup

show only small differences in performance. Turning to recall (see Figure 1(b)), *ChangeRate* and *Dynamics* perform quite similar over the entire interval and clearly outperform all strategies and all bandwidth constraints. For even only 15% bandwidth available, the recall values are above 0.957 while all other strategies achieve at most 0.93.

LOD sources vary in their sizes. As shown, most of the big data sources change frequently and, consequently, they are in the top ranked entries for strategies such as the *Dynamics* and *ChangeRate*. Also, some of the smaller data sources have a high change frequency. Therefore, in the ranking list provided by the *Dynamics* and *ChangeRate* strategies, a mix of big and small sources can be found in the top entries. For strategies such *BiggestFirst* and *ChangeRatio* only/most of the biggest sources are top ranked. In contrast, by the strategy *SmallestFirst* only the smallest sources are top ranked. When updating the smallest data sources first, even for a very small bandwidth, a great number of data sources can be fetched and consequently data changes can also be retrieved. This can be observed in the recall curve of the *SmallestFirst* strategy. When only low bandwidth is available, it is not possible to fetch data from big data sources since there is not enough bandwidth. This can be clearly seen for the *BiggestFirst* strategy, where updates are observed only when 20% or more is available. The more bandwidth is available, the more changes can be retrieved. Due to the mix of data sources sizes in the ranking list of the *Dynamics* and *ChangeRate* strategies, they are able to retrieve already data when only a very small bandwidth is available and overall are able to retrieve more modified data than the other strategies for all bandwidths. The others strategies narrow in quality when more bandwidth is available.

In general, the *single-step* experiments show that update strategies based on dynamics followed by change rate make best use of limited resources in terms of bandwidth. For very low relative bandwidth, the strategies based on data source dynamics tend to provide better results.

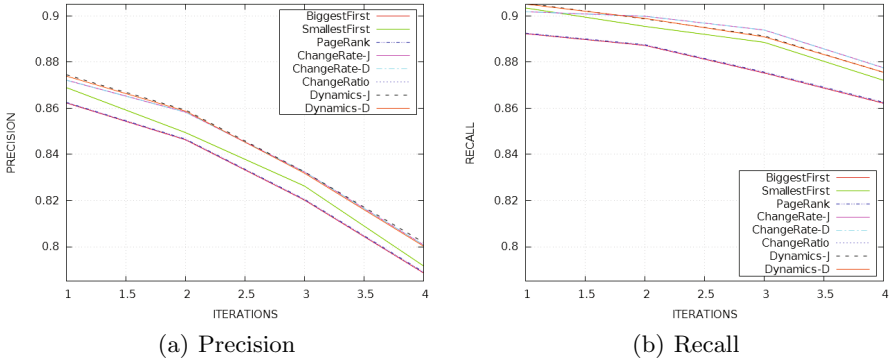


Fig. 3. Quality Outcomes for the Iterative Progression Setup at Low Bandwidth Level (5%).

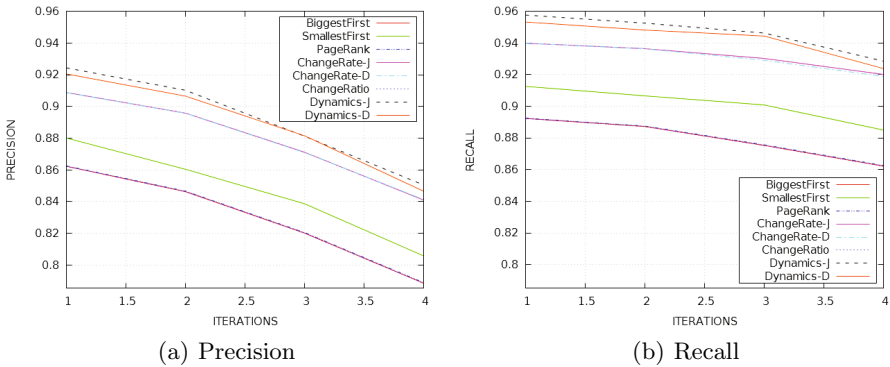


Fig. 4. Quality Outcomes for the Iterative Progression Setup at Mid-Level Bandwidth (15%).

Iterative Progression Evaluation. In this evaluation, we look at the evolving quality when considering iterative updates. This setup simulates real use case scenarios such as of a LOD search engine continuously updating its caches. In our experiments, we look how precision and recall behave over the iterations. First, we fix the bandwidth constraints. We choose a low (5%), mid (15%), and high (40%) bandwidth which provided low, average, and good outcomes based on the previous experiments (*single-step* evaluation).

Figure 3(a) and Figure 3(b) show precision and recall for bandwidth fixed at 5%. The x -axis represents the iterations (points in time) and the y -axis the quality in terms of precision and recall. Note that quality decreases along the iterations. This is expected, since only at the first iterations the update process starts from a perfect data cache. For low relative bandwidths, the impact on the (loss of) quality of iterative updates is quite similar for all strategies. Never-

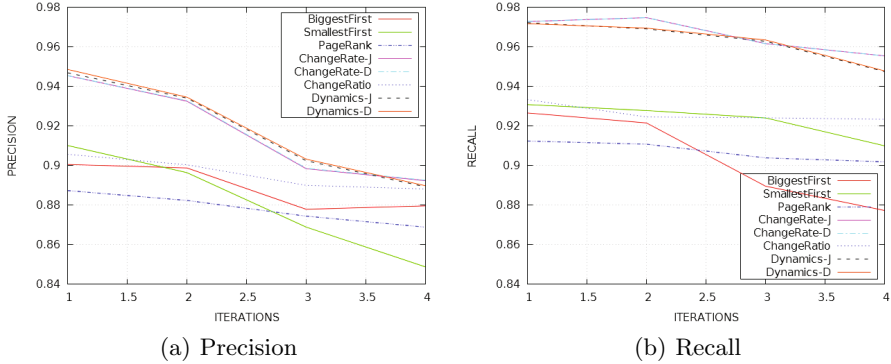


Fig. 5. Quality Outcomes for the Iterative Progression Setup at High-Level Bandwidth (40%).

theless, the plot confirms the previous discussion that the *Dynamics* strategies followed by *ChangeRate* are the more appropriate ones, if we need to predict the next best steps (and not only the first step anymore). Nevertheless, the strategies show a uniform loss of quality.

A similar output is observed for bandwidth fixed at 15% (see Figure 4(a) and Figure 4(b)). Here again, fetching data from the source that changes more than others ensure more accurate updates. Even if we can observe that the loss of quality is comparable, the *Dynamics* strategies followed by *ChangeRate* maintain a higher level of quality after the four iterations. *Dynamics* precision and recall decreases from 0.92 to 0.846 and 0.953 to 0.929 and *ChangeRate* from 0.908 to 0.841 and 0.939 to 0.918 after the four iterations, while the quality of the other strategies are mostly lower after only one or even no iteration.

Precision and recall under a relative high bandwidth (fixed at 40%) is shown in Figure 5(a) and Figure 5(b). The recall values of the *Dynamics* strategies and *ChangeRate* almost not change over the iterations (above of 0.947). The precision values decrease with a maximum of 0.889. Over all iterations, these strategies outperforms all the others even when only one-step update is applied. Interestingly, at this bandwidth level, the strategies which fetch data from the big data sources first show good performance since—up to that bandwidth level—it is possible to load a big data source entirely. As most changes concentrate in the big data sources, they are also able to fetch most of the changes. For instance, precision and recall of the *ChangeRatio* strategy reaches values of 0.888 and 0.923, respectively, after the iterations.

Overall, the results of this experiment setup confirms the discussion from the previous one, i.e, the strategies based on the dynamics features followed by the ones based on change rate are the more appropriate ones if we need to predict (iterative) updates. Certainly, the more bandwidth is available, the more changes can be grabbed (and therefore the rate of quality lost over the iterations is lower for all strategies). Still even after four iterations, *Dynamics* strategies

(followed by *ChangeRate*) were able to better maintain an up-to-date local copy for all different bandwidth levels. From our experiments and for low relative bandwidth, these strategies could definitely better support applications to fetch the most changed data (and thus to avoid to fetch unchanged data) than the other strategies.

5 Related Work

The evolution of the Web has been observed in [5] in order to obtain implications of changes on incremental Web crawlers. Incremental crawlers update local data collections if they recognize influencing changes. Likewise, the dynamics of Web pages is empirically analyzed in [3,7] with a dedicated focus on the update frequencies of search engine indices. Estimations for changes of data items and elements are proposed in [6]. Such estimations are used if the history of changes is incomplete, e.g., it is known that a Web page has changed but it is not known how often it has changed in a certain period.

Various related work have investigated the characteristics of the LOD cloud. Their goal is to apply these characteristics for the purpose of different applications such as query recommendation and indices updates. Some works conducted structural analysis of the LOD cloud such as [1,2,15] in order to obtain statistical insights into the characteristics of the data. In addition, there is related work on analyzing the LOD cloud in order to verify its compliance with established guidelines and best practices how to model and publish data as Linked Data [16,22]. Other works as by Neumann et al. [19] analyze LOD in order to obtain statistics like its distribution in the network.

Among those works that are dedicated on the study of the Linked Data dynamics, with a dedicated focus on the update frequencies of LOD search engine indices, Umbrich et al. [23] compare the dynamics of Linked Data and the dynamics of Linked datasets with HTML documents on the Web. Their change detection uses (i) HTTP metadata monitoring (HTTP headers including timestamps and ETags), (ii) content monitoring, and (iii) active notification of datasets. These three detection mechanisms are compared by several aspects like costs, reliability, and scalability of the mechanism. Similar to our approach, the content monitoring applies a syntactic comparison of the dataset content, i.e., a comparison of RDF triples (but ignoring inference). Change detection is a binary function which is activated whenever changes are found. In our evaluation, we consider more complex change metrics to allow fine-grained ranking.

The importance of caching for efficient querying linked data is analyzed by Hartig et al. [14]. Query execution is based on traversing RDF links to discover data that might be relevant for a query during the query execution itself. Data is cached and it is used for further queries. Caching show some beneficial impact to improve the completeness of the results. Additionally, Umbrich et al. [24] proposed a hybrid approach for answering SPARQL queries, i.e., deciding which parts of a query are suitable for local/remote execution. The authors estimate the freshness of materialized data using the notion of coherence for triple patterns against the live engine. Dehghanzadeh et al. [8] extend this approach by

extending the statistics of cardinality estimation techniques that are used in the join query processing phase.

The Dynamic Linked Data Observatory is a monitoring framework to analyze dynamics of Linked Data [17]. Snapshots of the Web of data are regularly collected and then compared in order to detect and categorize changes. Using these snapshots, the authors study the availability of documents, the links being added to the documents, and the schema signature of documents involving predicates and values for `rdf:type` and determine their change rate. Motivated by this work, Dividino et al. [11] analyzed the changes on the usage of the vocabulary terms in the DyLDO dataset. The authors show that the combination of vocabulary terms appearing in the LOD documents changes considerably.

6 Conclusion

In this paper, we propose and evaluate scheduling strategies for updating on a large-scale LOD dataset that is obtained from the cloud by weekly crawls over the course of three years. In a first setup, where we evaluate the quality of update strategies for a single and isolated update of a local data cache, we observe that update strategies based on dynamics or change rate make best use of limited resources in terms of bandwidth. For very low relative bandwidth, the strategies based on data source dynamics provide better results. Already only with 15% available bandwidth, we observed improvements of precision and recall for dynamics from 0.862 to 0.924 and from 0.892 to 0.957, respectively.

In a second evaluation setup, we evaluate the behavior of the strategies in a realistic scenario (e. g., a LOD search engine updating its caches) which involves measuring the quality of the local data cache when considering iterative updates over a longer period of time. Overall the results of this experiment setup confirms the discussion from the previous one. Especially for low relative bandwidth, update strategies based on dynamics or change rate are more appropriate to support applications to fetch the most changed data (and thus to avoid to fetch unchanged data) than the others strategies.

In future work, we plan to investigate the impact on the performance when combining different update strategies. We also intend to consider further evaluation setups such as the cold start setup, i. e., we measure how good is an update strategy starting from an empty cache and considering iterative updates over a longer period of time. At last, we mentioned in this paper that the implemented strategies do not differentiate whether a source is unavailable for a period of time or is deleted from the cloud. Therefore, we plan to extend these strategies to consider the availability of the LOD sources over time.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n° 610928.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets - on the design and usage of void, the ‘vocabulary of interlinked datasets’. In: LDOW. CEUR (2009)
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. Brewington, B.E., Cybenko, G.: How dynamic is the Web? *Computer Networks* (2000)
4. Cho, J., Garcia-Molina, H.: Synchronizing a database to improve freshness. In: SIGMOD (2000)
5. Cho, J., Garcia-Molina, H.: The evolution of the web and implications for an incremental crawler. In: VLDB, VLDB 2000. Morgan Kaufmann Publishers Inc. (2000)
6. Cho, J., Garcia-Molina, H.: Estimating frequency of change. *ACM Trans. Internet Technol.* (2003)
7. Cho, J., Ntoulas, A.: Effective change detection using sampling. In: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB. VLDB Endowment (2002)
8. Dehghanzadeh, S., Parreira, J.X., Karnstedt, M., Umbrich, J., Hauswirth, M., Decker, S.: Optimizing SPARQL query processing on dynamic and static data based on query time/freshness requirements using materialization. In: Supnithi, T., Yamaguchi, T., Pan, J.Z., Wuwongse, V., Buranarach, M. (eds.) JIST 2014. LNCS, vol. 8943, pp. 257–270. Springer, Heidelberg (2015)
9. Dividino, R., Kramer, A., Gottron, T.: An investigation of HTTP header information for detecting changes of linked open data sources. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 199–203. Springer, Heidelberg (2014)
10. Dividino, R., Gottron, T., Scherp, A., Gröner, G.: From changes to dynamics: dynamics analysis of linked open data sources. In: PROFILES. CEUR (2014)
11. Dividino, R., Scherp, A., Groner, G., Grotton, T.: Change-a-lod: does the schema on the linked data cloud change or not? In: COLD. CEUR (2013)
12. Gottron, T., Gottron, C.: Perplexity of index models over evolving linked data. In: Presutti, V., d’Amato, C., Gandon, F., d’Acquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 161–175. Springer, Heidelberg (2014)
13. Gottron, T., Scherp, A., Kraymer, B., Peters, A.: Lodatio: using a schema-level index to support users infinding relevant sources of linked data. In: KCAP. ACM (2013)
14. Hartig, O.: Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 154–169. Springer, Heidelberg (2011)
15. Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L., Ayers, D.: SCOVO: using statistics on the web of data. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 708–722. Springer, Heidelberg (2009)
16. Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., Decker, S.: An empirical survey of linked data conformance. *J. Web Sem.* (2012)

17. 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)
18. Konrath, M., Gottron, T., Staab, S., Scherp, A.: Schemex - efficient construction of a data catalogue by stream-based indexing of linked data. *J. Web Sem.* (2012)
19. Neumann, T., Moerkotte, G.: Characteristic sets: accurate cardinality estimation for RDF queries with multiple joins. In: *ICDE*. IEEE Computer Society (2011)
20. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999–66, Stanford InfoLab, November 1999. previous number = SIDL-WP-1999-0120, <http://ilpubs.stanford.edu:8090/422/>
21. Schaible, J., Gottron, T., Scheglmann, S., Scherp, A.: Lover: support for modeling data using linked open vocabularies. In: *EDBT/ICDT 2013 Workshops*. EDBT. ACM (2013)
22. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) *ISWC 2014, Part I*. LNCS, vol. 8796, pp. 245–260. Springer, Heidelberg (2014)
23. Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., Decker, S.: Towards dataset dynamics: change frequency of linked open data sources. In: *LDOW*. CEUR (2010)
24. Parreira, J.X., Umbrich, J., Karnstedt, M., Hogan, A.: Hybrid SPARQL queries: fresh vs. fast results. 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. 608–624. Springer, Heidelberg (2012)

CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets

Muhammad Intizar Ali^(✉), Feng Gao, and Alessandra Mileo

Insight Centre for Data Analytics, National University of Ireland, Galway, Ireland
{ali.intizar,feng.gao,alessandra.mileo}@insight-centre.org

Abstract. With the growing popularity of Internet of Things (IoT) and IoT-enabled smart city applications, RDF stream processing (RSP) is gaining increasing attention in the Semantic Web community. As a result, several RSP engines have emerged, which are capable of processing semantically annotated data streams on the fly. Performance, correctness and technical soundness of few existing RSP engines have been evaluated in controlled settings using existing benchmarks like LSBench and SRBench. However, these benchmarks focus merely on features of the RSP query languages and engines, and do not consider dynamic application requirements and data-dependent properties such as changes in streaming rate during query execution or changes in application requirements over a period of time. This hinders wide adoption of RSP engines for real-time applications where data properties and application requirements play a key role and need to be characterised in their dynamic setting, such as in the smart city domain.

In this paper, we present CityBench, a comprehensive benchmarking suite to evaluate RSP engines within smart city applications and with smart city data. CityBench includes real-time IoT data streams generated from various sensors deployed within the city of Aarhus, Denmark. We provide a configurable testing infrastructure and a set of continuous queries covering a variety of data- and application- dependent characteristics and performance metrics, to be executed over RSP engines using CityBench datasets. We evaluate two state of the art RSP engines using our testbed and discuss our experimental results. This work can be used as a baseline to identify capabilities and limitations of existing RSP engines for smart city applications.

1 Introduction

Recent advances in Semantic Technologies for IoT have created great opportunities for rendering IoT-enabled services in smart cities. As a result, an increasing number of cities have started to invest in data-driven infrastructures and services

This research has been partially supported by Science Foundation Ireland (SFI) under grant No. SFI/12/RC/2289 and EU FP7 CityPulse Project under grant No.603095. <http://www.ict-citypulse.eu>

for citizens, mostly focusing on creating and publishing a rich pool of dynamic datasets that can be used to create new services [8, 17]. Leveraging this data and semantic technologies, tools and solutions have been developed to abstract, integrate and process this distributed and heterogenous data sources.

One of the major aspects that captured the attention of the scientific community and standardisation bodies is the design of query languages, processes and tools to process RDF streams dynamically and in a scalable way¹. Despite the success of RDF Stream Processing (RSP) solutions in this direction [1, 2, 4, 12, 14], available benchmarks for their evaluation are either synthetic or mostly based on static data dumps of considerable size that cannot be characterised and broken down [13, 18]. Few of the existing RSP engines have been evaluated using offline benchmarks such as SRBench and LSBench [6, 13, 18], but none of them has been tested based on features that are significant in real-time scenarios. There is a need for a systematic evaluation in a dynamic setting, where the environment in which data is being produced and the requirements of applications using it are dynamically changing, thus affecting key evaluation metrics.

In this paper, we distinguish different characteristics of benchmarking for RSP engines with a closer look to real-time requirements of smart city applications. We use real-time datasets from the city of Aarhus and present their schema, time-dependent features and interdependencies. We provide a testing environment together with a set of queries classified into different categories for evaluation of selected application scenarios. CityBench will prove as a tool for evaluating RSP engines within smart city applications based on their dynamic features (including performance), and comparing RSP engines in terms of their ability to fulfil application-specific requirements.

Our main contributions in this paper can be summarised as follows:

- we identify a set of dynamic requirements of smart applications which must be met by RSP engines;
- we design a benchmark based on such requirements, using realtime datasets gathered from sensors deployed within the City;
- we provide a configurable benchmarking infrastructure, which allows to set up evaluation tests enabling fine tuning of various configuration parameters;
- we provide a set of queries covering broader features of the RSP Query Languages in selected scenarios;
- finally, we evaluate state of the art RSP engines on our benchmark with different configurations, and we perform an empirical analysis of the experimental results.

Structure of the Paper: Section 2 defines challenges and dynamic requirements for benchmarking of RSP engines in Smart City applications. We present the CityBench Benchmarking Suite in Section 3 and its evaluation in Section 4. Section 5 discusses state of the art, we conclude with final remarks in Section 6.

¹ <https://www.w3.org/community/rsp/> (l.a. Apr. 2015).

2 Smart City Applications: Challenges and Requirements for RSP

Challenges and requirements of smart city applications are inherently related to their dynamic nature and changing environment [9]. In this section, we identify various challenges (**Cn**) and respective requirements (**Rn**) which can potentially affect performance, scalability and correctness of smart city applications designed to query and integrate dynamic smart city datasets via RSP.

C1: Data Distribution. City data streams are instinctively distributed. Increase in the number of streams to be processed within a single query can have adverse effect over the performance of the engine. **R1:** RSP engines should be capable of addressing the challenge of high distribution and their performance should not be effected with higher degree of distribution.

C2: Unpredictable Data Arrival Rate. Data streams originated from sensor observation are mostly generated at a fixed rate. For example, a temperature sensor can be easily configured to produce each observation after a certain time period. Event data streams instead produce data at a variable rate and the observation rate for events is dependent upon the detection of a query pattern representing the event. **R2:** Applications consuming aggregated data streams at variable rates (e.g. events) should be able to cope with sudden increases in the streaming rate. Such increase or *stream burst* can potentially compromise the performance of RSP engines.

C3: Number of Concurrent Queries. Similar to the stream bursts, the number of users of an IoT-enabled smart city applications can suddenly increase. For example, a sudden increase of concurrent users of an application designed to monitor traffic conditions can be observed during traffic jams or accidents. **R3:** RSP engines should be stress tested by increasing the number of concurrent queries in their performance evaluation.

C4: Integration with Background Data. Some of the existing RSP engines have already demonstrated their capability to integrate background data. Executing queries over a larger size of such static background data may strongly affect the performance of RSP engine, and this aspect has not been thoroughly considered in current benchmarks. **R4:** RSP engines should be capable to deal with large amount of background data by applying proper data management techniques.

C5: Handling Quasi-Static Background Data. Current RSP implementations load background data before query execution over static and dynamic data and cache it for longer periods. However, some of the background data can be quasi-static (e.g. changing with irregular periodicity) such as the number of follower of a twitter user, or the price of utilities. Materialising this data at query time is not efficient, but caching might result in out-of-date results. **R5:** RSP engines should be able to efficiently update the quasi-static background data during query execution using effective strategies to determine what data is more likely to be out-of-date.

C6: On-Demand Discovery of Data Streams. In smart city environments, many applications do not have prior knowledge of the available streaming data sources that can potentially be relevant for a specific task. Therefore, discovering relevant streaming sources on the fly is a challenge. **R6:** RSP query languages should provide support for stream discovery and federation, possibly taking into account quality constraints so that the best available source is considered.

C7: Adaptation in Stream Processing. Smart city applications are operated over dynamic and distributed infrastructure, without any central control. This makes it difficult to provide efficient strategies for adapting to changing environments that are typical of smart city applications. For example availability of sensors, communication issues, changes in the environment or user needs can demand for the use of alternative data streams. **R7:** RSP solutions should be able to switch between multiple semantically equivalent data streams during query execution.

3 CityBench Benchmarking Suite

In this section, we present CityBench Benchmarking Suite consisting of, (i) *Benchmark Datasets*, designed over realtime smart city datasets, (ii) *Configurable Testbed Infrastructure*, containing a set of tools for dataset preparation and testbed set-up to benchmark RSP engines, and (iii) *Queries*, a set of continuous queries covering the query features and challenges discussed in Section 2. In what follows, we discuss each of these three components.

3.1 Benchmark Datasets

Leveraging the outcomes of the CityPulse project², we use the dataset collected from the city of Aarhus, Denmark^{3,4}. In this section, we briefly describe each of the dataset in the benchmark and elaborate on the semantic representation of the datasets.

Vehicle Traffic Dataset. This dataset contains traffic data. The City administration has deployed 449 pairs of sensors over the major roads in the city. Traffic data is collected by observing the vehicle count between two points over a duration of time. Observations are currently generated every five minutes. A meta-data dataset is also provided which contains information about location of each traffic sensor, distance between one pair of sensors and type of road where the sensors have been deployed. Each pair of traffic sensors reports the average vehicle speed, vehicle count, estimated travel time and congestion level between the two points set over a segment of road.

² <http://www.ict-citypulse.eu/>

³ We acknowledge the CityPulse consortium team for the provision of Datasets <http://iot.ee.surrey.ac.uk:8080/datasets.html> (l.a. Apr. 2015)

⁴ CityBench datasets are made publicly available by EU Project CityPulse, for use of any part of these datasets, the source must be properly acknowledged. The accuracy or reliability of the data is not guaranteed or warranted in any way.

Parking Dataset. Parking lots in Aarhus are equipped with sensors and capable of producing live data streams indicating number of vacant places. The Parking Dataset consists of observations generated by 8 public parking lots around the city.

Weather Dataset. Currently, there is only a single weather sensor available in the city to collect live sensor observations about the weather condition. Weather sensor data provides observations related to dew point ($^{\circ}\text{C}$), humidity (%), air pressure (mBar), temperature ($^{\circ}\text{C}$), wind direction ($^{\circ}$), and wind speed (kph).

Pollution Dataset. Pollution is directly related to the traffic level, however due to unavailability of the pollution sensors, a synthesised pollution data for the city of Aarhus is made available to complement the traffic dataset. Observation points for traffic sensors (446) are replicated to create mock-up sensors for pollution at the exact same location as traffic sensors. An observation for air quality index is generated every 5 minutes using a pre-selected pattern. Details regarding the procedure followed to synthesised pollution data are accessible at: <http://iot.ee.surrey.ac.uk:8080/datasets/pollution/readme.txt>.

Cultural Event Dataset. This dataset is quasi-static and contains cultural events provided by the municipality of Aarhus. The dataset is periodically updated to reflect the latest information related to planned cultural events,. Updates are available as data stream (a notification service notify of any updates in the dataset). However, due to the low frequency of updates, we consider this dataset as background knowledge and use it to demonstrate integration of the static data with the data streams.

Library Events Data. This dataset contains a collection of past and future library events hosted by libraries in the city. A total collection of 1548 events is described in this dataset. Similarly to the Cultural Events Dataset, updates in the Library Events Dataset are also not frequent, therefore the dataset is considered quasi-static.

User Location Stream. Most of the IoT-enabled smart city application are designed to be location-aware, therefore they strongly rely over updates on the location of mobile users. We synthesised a User Location Stream to mock-up a real usecase scenario of users roaming around. This data stream contains periodic observations with geo-location coordinates of a fictional mobile user.

Users of CityBench can download the existing as well as any future datasets from the CityBench website⁵. All of the above mentioned datasets are semantically annotated and interlinked using the CityPulse information model⁶.

⁵ <https://github.com/CityBench/Benchmark>

⁶ <http://iot.ee.surrey.ac.uk:8080/info.html>

configure a variety of metrics for the evaluation of RSP engine. Figure 2 provides an overview of the CTI, there are three main modules, (i) *Dataset Configuration Module*: allows configuration of stream related metrics, (ii) *Query Configuration Module*: allows configuration of query related metrics, and (iii) *Performance Evaluator*: is responsible for recording and storing the measurements of the performance metrics. We discuss configuration metrics in what follows.

Changes in Input Streaming Rate: The throughput for data stream generation can be configured in CityBench. For example, a rate $r \in [1, inf]$ can be configured to set up the streaming rate to the real interval between observations ($r = 1$ means replay at original rate), or a frequency f can be used to set a different streaming rate.

PlayBack Time: CityBench also allows to playback data from any given time period to replay and mock-up the exact situation during that period.

Variable Background Data Size: CityBench allows to specify which dataset to use as background knowledge, in order to test the performance of RSP engines with different static datasets. We also provide duplicated versions (with varying size) of two static datasets, *Cultural Event Dataset* and *Library Event Dataset*. Any version of the given background datasets can be loaded to test RSP engines with different size of background data.

Number of Concurrent Queries: CityBench provides the ability to specify any number of queries to be deployed for testing purposes. For example, any number of queries can be selected to be executed concurrently any number of times. Such situation will simulate a situation where a number of simultaneous users are executing the same query using any application.

Increase in the Number of Sensor Streams within a Single Query: In order to test the capability of the RSP engine to deal with data distribution, CityBench makes it possible to configure various size of streams involved within a single query. We achieved this by increasing the number of streams to be observed as relevant for the query. Query similar to traffic condition monitoring over a given path are best candidates for distribution test and number of streams involved within a query can be increased by simply increasing the length of the observed path.

Selection of RSP Query Engines: CityBench allows to seamlessly use different query engines as part of the testing environment. Currently, we support CQELS and C-SPARQL. However, we encourage users to extend the list of RSP engines by embedding the engine within CTI.

3.3 Smart City Applications Queries over CityBench Datasets

In this section, we present a set of 13 queries covering most of the features and challenges discussed in Section 2. Our main goal while designing the queries is to highlight and evaluate the characteristics and features of the RSP engines which are most relevant to the smart city applications requirements. Benchmark queries designed to cover query specific features of the RSP engines can be found in the state of the art [6, 13, 18]. In what follows we identify three smart city applications from the CityPulse scenarios⁸ and generate queries which are relevant for applications deployed around these scenarios.

Multi-modal Context-Aware Travel Planner: This application relies on modules that can provide one or more alternative paths for users to reach a particular location. On top of these modules, the application aims at dynamically optimising users' path based on their preferences on route type, health and travel cost. In addition to that, the application continuously monitors factors and events that can impact this optimisation (including traffic, weather, parking availability and so on) to promptly adapt to provide the best up-to-date option. Relevant queries for this application scenario are listed below.

Q1: What is the traffic congestion level on each road of my planned journey?

This query monitors the traffic congestion from all traffic sensors located on the roads which are part of the planned journey.

Q2: What is the traffic congestion level and weather conditions on each road of my planned journey?

Q2 is similar to Q1 with an additional type of input streams containing weather observations for each road at the planned journey of the user.

Q3: What is the average congestion level and estimated travel time to my destination?

This query includes the use of aggregate functions and evaluates the average congestion level on all the roads of the planned journey to calculate the estimated travel time.

Q4: Which cultural event happenig now is closest to my current location?

Q4 consumes user location data streams and integrates it with background knowledge on the list of cultural events to find out the closest cultural event happening near his current location.

Q5: What is traffic congestion level on the road where a given cultural event X is happening? Notification for congestion level should be generated every minute starting from 10 minutes before the event X is planned to end, till 10 minutes after.

Q5 is a conditional query which should be deployed at the occurrence of an event and have predefined execution duration.

⁸ <http://www.ict-citypulse.eu/scenarios/>

Parking Space Finder Application: This application is designed to facilitate car drivers in finding a parking spot combining parking data streams and predicted parking availability based on historical patterns. Additional sources such as timed no parking zones, congested hot spots and walking time from parking to a point of interest, the user can reduce circulation time and optimise parking management in the city. Queries related to this application are listed below.

Q6: What are the current parking conditions within range of 1 km from my current location?

This query represents a most common query issued by users of a parking application to easily find a nearby parking place.

Q7: Notify me whenever a parking place near to my destination is full.

Q7 is a combination of travel planner and parking application, where a user wants to be notified about parking situation close to the destination.

Q8: Which parking places are available nearby library event X?

This query combines parking data streams with the static information about the library events to locate parking spaces nearby the library.

Q9: What is the parking availability status nearby the city event with the cheapest tickets price?

Similarly to Q8, this query monitors parking availability near a city event which has the cheapest ticket price.

Smart City Administration Console: This application facilitates city administrators by notifying them on the occurrence of specific events of interest. The dashboard relies on data analytics and visualisation to support early detection of any unexpected situation within the city and takes immediate actions, but it can also be used as a city observatory for analysing trends and behaviours as they happen. Queries related to this application are listed below.

Q10: Notify me every 10 minutes, about the most polluted area in the city.

Q10 is an analytical query executed over the pollution data streams to find out which area in the city is most polluted and how this information evolves.

Q11: Notify me whenever no observation from weather sensors have been generated in the last 10 minutes.

This query helps to detect any faulty sensors which are not generating observations or networking issues.

Q12: Notify me whenever the congestion level on a given road goes beyond a predefined threshold more than 3 times within the last 20 minutes.

This query helps in early detection of areas where traffic conditions are becoming problematic.

Q13: Increase the observation monitoring rate of traffic congestion if it surpasses a pre-specified threshold.

This query provides a more frequent status update on congestion levels in critical conditions such as traffic jams or accidents.

CityBench provides all 13 queries ready to execute over CQELS and C-SPARQL, which can be downloaded from CityBench website⁹.

4 Experimental Evaluation and Empirical Analysis

In order to showcase the feasibility of CityBench and highlight the importance of configuration parameters, we conducted our experimental evaluation over CQELS and C-SPARQL engines using CityBench Benchmarking Suite. We set up a testbed with multiple configuration of *CTI* performance metrics^{10,11}. We evaluated the two RSP engines with respect to (i) Latency, (ii) Memory Consumption, and (iii) Completeness. The experiments in this paper covers requirements **R1** to **R4** (see Section 2). However there is no existing RSP engines which can meet **R5** to **R7**. It is worth mentioning that the overhead caused by the benchmark is insignificant and does not pose threats the validity of the results, i.e., for latency it costs several milliseconds to annotate a CSV row as a RDF graph, for memory consumption the benchmark uses up to 10 MB for tracking the results produced, for completeness the benchmark do not introduce any overhead.

4.1 Latency

Latency refers to the time consumed by the RSP engine between the input arrival and output generation. We evaluate the latency of RSP engines by increasing the number of input streams within a query and by increasing the number of concurrent queries executed.

Increasing the Number of Input Streams. We designed three variations of query *Q10*¹² to generate an immediate notification about polluted areas in the city with three configurations for number of input data streams (2, 5 and 8). Results shown in Figure 3 depict that the overhead for C-SPARQL was minimal with increasing number of streams, however CQELS suffer from abnormal behaviour for query with 5 input streams (secondary y-axis in Figure 3) and was unable to process 8 input streams within a single query.

⁹ <https://github.com/CityBench/Benchmark/>

¹⁰ Experiments are reproducible using *CTI* over CityBench Datasets, details are available at: <https://github.com/CityBench/Benchmark/>

¹¹ All experiments are carried out on a Macbook Pro with a 2.53 GHz duo core cpu and 4 GB 1067 MHz memory.

¹² We selected different queries for each experiment based on their suitability for the corresponding configuration metric. A comprehensive report containing complete results for all queries is available at CityBench website: https://github.com/CityBench/Benchmark/tree/master/result_log/samples

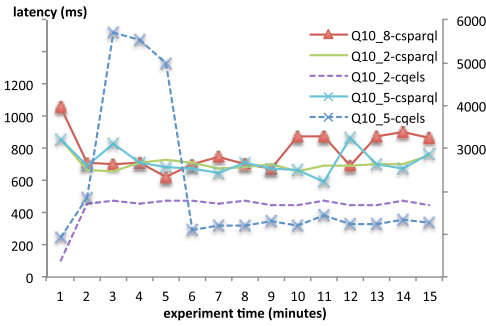


Fig. 3. Latency over Increasing Numer of Data Streams

0 to 10 for all three queries. However, CQELS performance is not much effected over subsequent increase from 10 to 20. As depicted in Figure 6 and Figure 7, C-SPARQL seems to have a constant size of overhead for delay with the increasing number of concurrent queries in contrast to CQELS.

Increasing the Number of Concurrent Queries.

We performed our scalability test by executing *Q1*, *Q5* and *Q8* over both engines. Queries are executed with three different configuration (1, 10, and 20) for number of concurrent queries. Figure 4 and Figure 5 show the effect over latency with increasing number of concurrent queries for CQELS. A closer look at the results reveals that CQELS has a substantial delay, when the number of concurrent queries is increased from

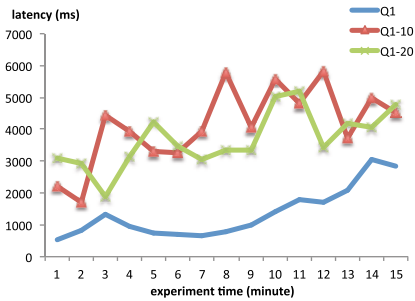


Fig. 4. Latency over Increasing Number of Concurrent Queries (*Q1* over CQELS)

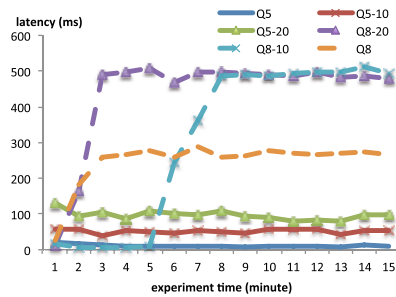


Fig. 5. Latency over Increasing Number of Concurrent Queries (*Q5* and *Q8* over CQELS)

4.2 Memory Consumption

We evaluated the two RSP engines by observing the usage of system memory during the concurrent execution of an increasing number of queries and increasing size of background data.

Increasing the Number of Concurrent Queries. We used query *Q1* and *Q5* and measure memory consumption during 15 minutes execution for each query.

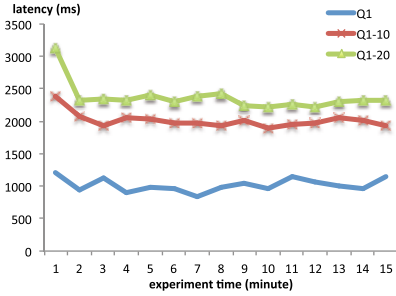


Fig. 6. Latency over Increasing Number of Concurrent Queries (*Q1* over C-SPARQL)

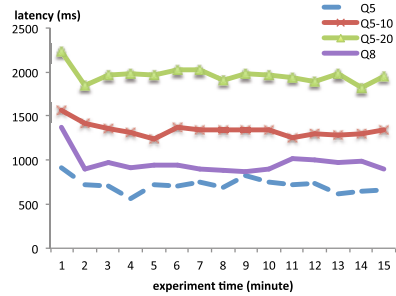


Fig. 7. Latency over Increasing Number of Concurrent Queries (*Q5* and *Q8* over C-SPARQL)

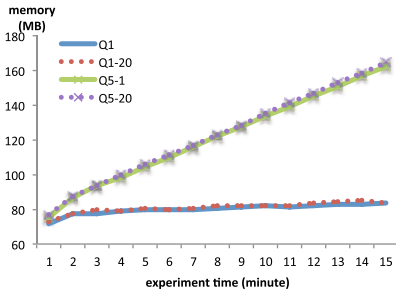


Fig. 8. Memory Consumption for Increasing Number of Concurrent Queries (C-SPARQL)

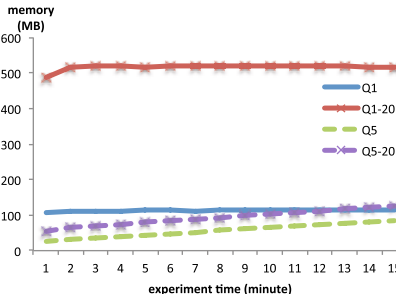


Fig. 9. Memory Consumption for Increasing Number of Concurrent Queries (CQELS)

As shown in Figure 8, with an increasing number of concurrent queries, C-SPARQL has a minimal impact on memory consumption for both queries. However, with increasing duration for query execution, there is a constant increase in memory consumption for *Q5*, rate of increase in memory is similar for both single query execution and 20 concurrent queries execution. In contrast, CQELS seems to have increasing memory consumption issue for *Q1*, there is also a substantial increase in memory consumption for *Q1* after an increase in the number of concurrent queries from 1 to 20. As depicted in Figure 9, CQELS has better performance regarding the stability of the engine over the time period of 15 minutes execution of *Q5*. Also, it is noticeable that the memory consumption of *Q5* increases linearly and it would eventually reach the memory limit and crash the engine. The reason of the abnormal behaviour is perhaps the cross-product join on the static data in *Q5* creates a lot (millions) of intermediate results and are not cleared from the cache properly in both engines.

Increasing the Size of Background Data. We analysed memory consumption while increasing the size of background data. We generated three versions of

background data required for the execution of query $Q5$, increasing the size from 3MB to 20MB and 30 MB. Figure 10 shows that CQELS seems to be better at memory management with background data of increasing size.

4.3 Completeness

We evaluated the completeness of results generated by RSP engines by executing Query $Q1$ with variable input rate of data streams. We allow each stream to produce x observations and count y unique observation IDs in the results, hence we have the completeness $c = y/x$. Note that we don't stop the streams immediately when they finished sending triples but waited for a period of time until no new results are generated, this ensured that the stream engines have enough time for query processing. Figure 11, shows that CQELS completeness level has been dropped up to 50%, while C-SPARQL continue to produce results with a completeness ratio of above 95%. The most probable cause of the completeness drop in CQELS is the complexity and concurrency of join over multiple streams.

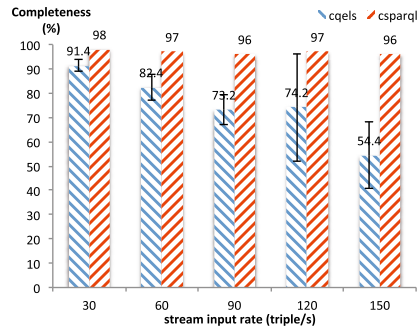
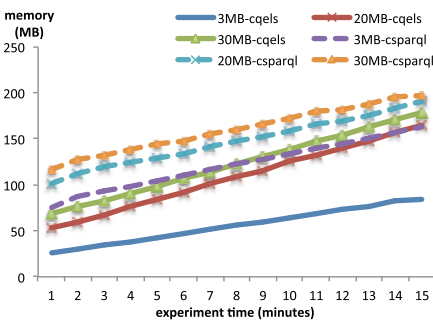


Fig. 10. Memory Consumption for Increasing Size of Background Data ($Q5$)

Fig. 11. Completeness of Results with Increasing Rate of Input Stream.eps

5 Related Work

With advances in the use of semantic technologies to process linked stream data, tools and systems for RSP started to use SPARQL benchmarking systems to test their applications. There are 3 main prominent efforts in this area [3, 11, 16]. The Berlin SPARQL Benchmark is the most widely used RDF benchmarking system which portrays a usecase scenario of e-commerce. Lehigh university benchmark is designed to test OWL reasoning and inferencing capabilities over a university domain ontology. SP² benchmark uses DBLP data¹³. All of these SPARQL

¹³ <http://dblp.uni-trier.de/>

benchmarks are inspired by traditional approaches for benchmarking relational database management systems.

Understanding the different requirements and evaluation parameters needed for RDF data, new benchmarks specifically targeting the evaluation of RSP engines have been proposed. LS Bench and SR Bench are two well known efforts for benchmarking RSP engines. SR Benchmark is defined on weather sensors observations collected by Kno.e.sis¹⁴. The dataset is part of the Linked Open Data Cloud and contains weather data collected since 2002¹⁵. All sensor observations are semantically annotated using the SSN ontology. Beside weather streams, SR contains two static datasets (GeoNames¹⁶ and DBPedia¹⁷) for integration of streaming data with background knowledge. The benchmark contains verbal description of 17 queries covering stream flow, background integration and reasoning features. However, due to the lack of a common RDF stream query language, some of the queries are not supported by the existing engines and therefore cannot be executed.

LS Benchmark is a synthetically generated dataset on linked social data streams. The dataset contains 3 social data streams, namely (i) Location (GPS coordinates) stream of a social media user, (ii) stream of micro posts generated or liked by the user, and (iii) a stream of notification whenever a user uploads an image. LS Bench also provides a data generator to synthesised datasets of varying size. LS Bench contains 12 queries, covering processing of streaming data as well as background data integration.

Both LS and SR benchmarks focus on evaluating RSP engines to demonstrate their query language support, process query operators and performance in a pre-configured static testbed. Best practices to design a benchmark are discussed in [10,15]. Real-world environment for the applications using RSP is however dynamic. In [7], authors have demonstrated that synthesised benchmark datasets do not portray the actual dataset requirements and therefore might produce unreliable results. CityBench extends the existing benchmarks and takes a new perspective on the evaluation of RSP engine which relies on the applications requirements and dynamicity of the environment to draw a picture that is closer to reality.

6 Concluding Remarks and Future Directions

CityBench is a benchmark for the evaluation of RSP solutions in real dynamic settings, with real city data. This work has been motivated by the need to benchmark RSP systems moving away from pre-configured static testbed towards a dynamic and configurable infrastructure (CTI). This comprehensive benchmarking suite includes not only streaming and static datasets but also semantic annotation tools, stream simulation capabilities, and a set of parameters that best

¹⁴ <http://knoesis.wright.edu>

¹⁵ <http://wiki.knoesis.org/index.php/LinkedSensorData>

¹⁶ <http://datahub.io/dataset/geonames>

¹⁷ <http://wiki.dbpedia.org/>

represent the set of data- and application- dependent characteristics and performance metrics that are typical of smart city applications.

This work will serve as a baseline for the evaluation of RSP engines in real application scenarios, and can be extended to accommodate additional features and datasets. Our initial evaluation of CityBench suggests the requirements identified to characterise smart city applications using streaming data provide a richer set of dimensions to evaluate RSP engines. The ability to tune these dimensions offers interesting insights on how application requirements play a key role in comparing and choosing one RSP solution over another. There are substantial differences not only in the language features but also in the windows operator and processing implemented within existing RSP engines, which is also reflected in how such engines perform on CityBench under different configurations.

Motivated by the need for a better approach to RSP, standardisation activities within the W3C RSP WG ¹⁸ have identified the need to converge towards a unified model for producing, transmitting and continuously querying RDF Streams. We believe that requirements and results presented in this paper can help guiding the roadmap towards better RSP solutions for smart cities and beyond. We are currently actively engaging with the RSP community towards this goal.

References

1. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proc. of the WWW 2011. ACM (2011)
2. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: SPARQL for continuous querying. In: Proc. of WWW, pp. 1061–1062. ACM (2009)
3. Bizer, C., Schultz, A.: Benchmarking the performance of storage systems that expose SPARQL endpoints. In: World Wide Web Internet And Web Information Systems (2008)
4. Bolles, A., Grawunder, M., Jacobi, J.: Streaming SPARQL - extending SPARQL to process data streams. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 448–462. Springer, Heidelberg (2008)
5. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The ssn ontology of the w3c semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web **17**, 25–32 (2012)
6. Dell’Aglio, D., Calbimonte, J.-P., Balduini, M., Corcho, O., Della Valle, E.: On correctness in RDF stream processor benchmarking. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 326–342. Springer, Heidelberg (2013)

¹⁸ <https://www.w3.org/community/rsp/>

7. Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In: Proc. of ACM SIGMOD 2011, pp. 145–156. ACM (2011)
8. Tönjes, R., et al.: Real time iot stream processing and large-scale data analytics for smart city applications, 2014. In: Poster presented at European Conference on Networks and Communications
9. Gao, F., Ali, M.I., Mileo, A.: Semantic discovery and integration of urban data streams. In: Proc. of S4SC @ ISWC 2014, pp. 15–30 (2014)
10. Gray, J.: Benchmark handbook: for database and transaction processing systems. Morgan Kaufmann Publishers Inc. (1992)
11. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for owl knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web **3**(2), 158–182 (2005)
12. Komazec, S., Cerri, D., Fensel, D.: Sparkwave: continuous schema-enhanced pattern matching over RDF data streams. In: Proc. of DEBS 2012, pp. 58–68 (2012)
13. Le-Phuoc, D., Dao-Tran, M., Pham, M.-D., Boncz, P., Eiter, T., Fink, M.: Linked stream data processing engines: facts and figures. 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. 300–312. Springer, Heidelberg (2012)
14. 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)
15. Scharrenbach, T., Urbani, J., Margara, A., Della Valle, E., Bernstein, A.: Seven commandments for benchmarking semantic flow processing systems. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 305–319. Springer, Heidelberg (2013)
16. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: Sp²bench: a SPARQL performance benchmark. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, pp. 222–233. IEEE (2009)
17. Nechifor, C.-S., Sheth, A., Mileo, A., Bischof, S., Karapantelakis, A., Barnaghi, P.: Semantic modeling of smart city data. In: Proc. of the W3C Workshop on the Web of Things: Enablers and Services for an Open Web of Devices, Berlin, Germany, June 2014. W3C (2014)
18. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.-P.: SRBench: a streaming RDF/SPARQL benchmark. 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. 641–657. Springer, Heidelberg (2012)

Evaluation

A Multi-reasoner, Justification-Based Approach to Reasoner Correctness

Michael Lee^(✉), Nico Matentzoglou, Bijan Parsia, and Uli Sattler

The University of Manchester, Oxford Road, Manchester M13 9PL, UK
{michael.lee-5,nicolas.matentzoglou,
bijan.parsia,uli.sattler}@manchester.ac.uk

Abstract. OWL 2 DL is a complex logic with reasoning problems that have a high worst case complexity. Modern reasoners perform mostly very well on naturally occurring ontologies of varying sizes and complexity. This performance is achieved through a suite of complex optimisations (with complex interactions) and elaborate engineering. While the formal basis of the core reasoner procedures are well understood, many optimisations are less so, and most of the engineering details (and their possible effect on reasoner correctness) are unreviewed by anyone but the reasoner developer. Thus, it is unclear how much confidence should be placed in the correctness of implemented reasoners. To date, there is no principled, correctness unit test-like suite for simple language features and, even if there were, it is unclear that passing such a suite would say much about correctness on naturally occurring ontologies. This problem is not merely theoretical: Divergence in behaviour (thus known bugginess of implementations) has been observed in the OWL Reasoner Evaluation (ORE) contests to the point where a simple, majority voting procedure has been put in place to resolve disagreements.

In this paper, we present a new technique for finding and resolving reasoner disagreement. We use justifications to cross check disagreements. Some cases are resolved automatically, others need to be manually verified. We evaluate the technique on a corpus of naturally occurring ontologies and a set of popular reasoners. We successfully identify several correctness bugs across different reasoners, identify causes for most of these, and generate appropriate bug reports and patches to ontologies to work around the bug.

Keywords: OWL · Reasoning · Debugging · Justifications

1 Introduction

A key advantage of expressive description logic ontologies (such as those encoded into OWL 2 DL) is that automated reasoners *help*. As often stated, reasoners make implicit knowledge explicit and this has benefits both at development time and at run time. One of the most obvious development time uses is for *debugging* ontologies. Reasoners *detect* faulty entailments (e.g., contradictions or unsatisfiable classes) and are a key component in *explaining* them. At runtime,

reasoners enable new sorts of functionality such as *post-coordination* [7, 11, 12] of terminologies as well as the discovery of new knowledge [17] or on the fly data integration [3].

Reasoners are complex pieces of software and their behaviour is opaque even to experts. Modern ontologies are typically too large and complex for any reasonable verification of the reasoners behaviour: Indeed, we rely on reasoners to help us manage those ontologies in the first place. Thus, we need techniques to help verify reasoner correctness.

This is not merely a theoretical issue (as bug lists for various reasoners attest). The complexity of the implementation makes a formal verification of correctness, or even the generation of a non-arbitrary set of automated unit tests, a near impossibility. Incompleteness (i.e., missing some entailments) is particularly challenging for human inspectors to detect both because the number of nonsubsumptions in any ontologies is very large (compared to the number of subsumptions) and because positive information has much higher saliency than missing information.

However, even if we can detect that there is a problem with a reasoner, coping with that problem is also difficult. Ideally, ontology engineers should be able to generate a succinct, informative bug report and a “patch” to their ontology that mitigates the problem (if switching from a buggy reasoner is not possible).

The detection problem can be mitigated by the use of multiple reasoners. If reasoners *disagree* on some entailment we know that there is at least one problem in at least one of the reasoners. Such disagreements naturally emerge in reasoner competitions such as the one conducted as part of the OWL Reasoner Evaluation workshop (ORE). Majority voting (MV) is often used in those competitions to resolve disagreements. When a disagreement occurs in the inferred class hierarchy, the verdict of the majority of reasoners is taken as truth. In case of a tie, the correct reasoner is selected at random. This technique is obviously unreliable: the majority might be wrong, in particular because some reasoners share algorithms, optimisations, and even code. Equally obviously, this resolution technique does not help with bug reports or workarounds.

In this paper, we present an extended voting method to determine reasoner correctness and narrow down potential causes of disagreement amongst a set of dissenting reasoners in an efficient manner. It is semi-automated without too heavy a dependence on human expertise.

Similarly to ORE, we first identify disagreements in class hierarchies between reasoners. A disagreement is an entailment that some reasoners infer and others not. For every disagreement, we generate a number of justifications which will be used both to provide an extra check on the reasoners to their commitment of their side of the disagreement and to provide material for debugging. We then apply a series of automated, semi-automated, and manual inspections of the justifications to determine whether they are correct. If the justification is correct, then the disagreement is conclusively resolved in favour of the positive subsumption.

We have evaluated our method using a corpus of BioPortal ontologies. We first identify cases of potential bugs, classify them according to our method, and analyse the result to identify causes of some of the bugs.

2 Preliminaries

We assume a basic familiarity with description logics, OWL, and reasoners. For those unfamiliar with the subject we suggest the Description Logic handbook [2].

Throughout this paper, we use OWL as a short form of OWL 2 DL, \models for the usual entailment relation, and \mathcal{O} for an OWL ontology, i.e., a set of axioms. We use $\tilde{\mathcal{O}}$ for the signature of \mathcal{O} , i.e., the set of class, property and individual names in \mathcal{O} . *Classification* is the process of determining, for every $A, B \in \tilde{\mathcal{O}} \cup \{\perp, \top\}$ whether $\mathcal{O} \models A \sqsubseteq B$. If $A \in \tilde{\mathcal{O}}$ and $\mathcal{O} \models A \sqsubseteq \perp$, then we call A *unsatisfiable*. We use \mathcal{R} for a description logic reasoner and $\mathcal{E}(\mathcal{O}, \mathcal{R})$ for the set of atomic subsumptions found by \mathcal{R} during classification of \mathcal{O} , i.e., $\mathcal{E}(\mathcal{O}, \mathcal{R})$ is the *inferred class hierarchy* of \mathcal{O} . \mathcal{R} is *correct on \mathcal{O}* if, for any $A, B \in \tilde{\mathcal{O}} \cup \{\perp, \top\}$, we have $A \sqsubseteq B \in \mathcal{E}(\mathcal{O}, \mathcal{R})$ if and only if $\mathcal{O} \models A \sqsubseteq B$.

Given $\mathcal{O} \models \eta$, a *justification for η* is a (subset) minimal set of axioms in \mathcal{O} which entails η . That is, \mathcal{J} is a justification for $\mathcal{O} \models \eta$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$ and there is no $\mathcal{J}' \subset \mathcal{J}$ such that $\mathcal{J}' \models \eta$.

We call a tuple $\langle \mathcal{O}, \eta, \mathcal{J}, \mathcal{R}_{just}, \mathcal{R}, v_{\mathcal{O}}, v_{\mathcal{J}} \rangle$ a *case*, where \mathcal{R}_{just} is a reasoner that generated the justification \mathcal{J} for the entailment η , \mathcal{R} is the reasoner that tested it and came to the following verdicts:

- $v_{\mathcal{O}} = 1$ if $\eta \in \mathcal{E}(\mathcal{O}, \mathcal{R})$, and 0 otherwise.
- $v_{\mathcal{J}} = 1$ if $\eta \in \mathcal{E}(\mathcal{J}, \mathcal{R})$, and 0 otherwise.

3 A Fine-Grained Justification Based Method for Verifying Reasoner Correctness

We can identify reasoner disagreements in three ways:

- *Ontology level*: reasoners have produced different inferred class hierarchies for the same ontology.
- *Entailment level*: reasoners have different verdicts whether $\mathcal{O} \models \eta$, i.e., $\eta \in \mathcal{E}(\mathcal{J}, \mathcal{R}) \setminus \mathcal{E}(\mathcal{J}', \mathcal{R}')$. An ontology level disagreement requires at least one entailment level disagreement (and vice versa). If we consider more than two reasoners, then there are only two sides to an entailment level disagreement (thus, there will always be two coalitions). This is not true at the ontology level, where a set of n reasoners can produce n distinct class hierarchies.
- *Entailment-justification level*: reasoners have different verdicts whether $\mathcal{J} \models \eta$, for a given justification \mathcal{J} for entailment $\eta \in \mathcal{E}(\mathcal{O}, \mathcal{R})$.

At each level, coalitions can occur where varying subsets of the reasoners are in agreement. By looking at more granular decisions (e.g., not just the whole hierarchy but individual entailments; not just entailments from the whole ontology but also from purported justifications), we have more “voting opportunities” which allow us to make more informed decisions about where the errors probably lie as well as justifications which are much easier for humans to verify.

The proposed method has three parts: 1) Discover disagreements and record them in the form of cases, 2) classify cases, and 3) resolve disagreements. Steps 1 and 2 are fully automated, whereas 3 involves some human intervention.

To discover disagreements, given an ontology \mathcal{O} and reasoners $\mathfrak{R} = \mathcal{R}_1, \dots, \mathcal{R}_m$ (where $m \geq 2$), we:

1. **Determine ontology agreement:** First, compute $\mathcal{E}(\mathcal{O}, \mathcal{R})$ for each $\mathcal{R} \in \mathfrak{R}$. If there is a pair $\mathcal{R}_j, \mathcal{R}_k \in \mathfrak{R}$ such that $\mathcal{E}(\mathcal{O}, \mathcal{R}_j) \neq \mathcal{E}(\mathcal{O}, \mathcal{R}_k)$, then there is a disagreement with respect to \mathcal{O} and we continue with (2).
2. **Extract entailment disagreements:** compute all entailments H that are found by some but not all reasoners, i.e., $H = \bigcup \mathcal{E}(\mathcal{O}, \mathcal{R}_i) \setminus \bigcap \mathcal{E}(\mathcal{O}, \mathcal{R}_i)$.
3. **Extract justifications:** For all $\eta \in H$ and $\mathcal{R} \in \mathfrak{R}$, we attempt to extract a justification¹ in \mathcal{O} for η using \mathcal{R} (we try all reasoners for this, including the ones for which $\eta \notin \mathcal{E}(\mathcal{O}, \mathcal{R})$). If successful, we record the pair $\langle \mathcal{J}, \mathcal{R} \rangle$. Please note that \mathcal{J} may be the same or different for the same entailment across different reasoners.
4. **Justification testing:** For each justification \mathcal{J} for η and each reasoner $\mathcal{R} \in \mathfrak{R}$ we check whether $\eta \in \mathcal{E}(\mathcal{J}, \mathcal{R})$ and record the resulting case (see above) in \mathcal{C} .

Next, we classify each case $c = \langle \mathcal{O}, \eta, \mathcal{J}, \mathcal{R}_{just}, \mathcal{R}, v_{\mathcal{O}}, v_{\mathcal{J}} \rangle \in \mathcal{C}$ in one of four categories:

1. **Consistent Yes**, where $v_{\mathcal{O}}$ and $v_{\mathcal{J}}$ are both 1. This means that \mathcal{R} found η by classification and verified the justification.
2. **Consistent No**, where $v_{\mathcal{O}}$ and $v_{\mathcal{J}}$ are both 0. This means that \mathcal{R} did not find η by classification and rejected the justification.
3. **Definite Bug**, where $v_{\mathcal{O}}$ is 0 and $v_{\mathcal{J}}$ is 1. Here, \mathcal{R} did not find η by classification but accepted the justification. This is quite problematic since monotonicity dictates that if η follows from a subset of \mathcal{O} it follows from \mathcal{O} , so *this reasoner* has an error.
4. **Possible Bug** where $v_{\mathcal{O}}$ is 1 and $v_{\mathcal{J}}$ is 0. Here, \mathcal{R} found η by classification but rejected the justification. This is either due to an error in \mathcal{R} or in \mathcal{R}_{just} which caused it to generate a spurious \mathcal{J} . Thus this case indicates a possible bug with this reasoner.

¹ Note that we attempt to extract only 1 justification (per reasoner) per entailment. While there might well be multiple putative justifications that could be extracted using a given reasoner, and the more justifications, the more cases, there is a high computational cost to extracting all justifications and the cost to human verifiers is potentially even higher. As we will see, attempting 1 has been highly successful.

Finally, we resolve disagreements: having computed and categorised cases, we still do not know which side of a disagreement is correct. However, we can do more granular comparisons. Consider the following pair of (abstract) cases:

$$\begin{aligned} c_1 &= \langle \mathcal{O}, \eta, \mathcal{J}, \mathcal{R}, \mathcal{R}_1, 1, 1 \rangle \\ c_2 &= \langle \mathcal{O}, \eta, \mathcal{J}, \mathcal{R}, \mathcal{R}_2, 0, 1 \rangle \end{aligned}$$

\mathcal{R}_1 found $\mathcal{O} \models \eta$ during classification and verified that $\mathcal{J} \models \eta$ for a justification $\mathcal{J} \subseteq \mathcal{O}$ extracted by \mathcal{R} . Contrariwise, \mathcal{R}_2 did not find $\mathcal{O} \models \eta$ during classification, but verified that $\mathcal{J} \models \eta$. We know, from our case classification, that \mathcal{R}_2 has a bug. Given that \mathcal{R}_1 answers consistently and that justifications are less likely to cause errors (being smaller and simpler than their parent ontologies), it is reasonable to bet on \mathcal{R}_1 in this case. Unless we want to seek conclusive human verification, this may be the best we can do.

To properly resolve disagreements, we need to determine whether a given justification is correct or not. If a justification is, indeed, a justification (that is, it is conclusively determined that $\mathcal{J} \models \eta$) then any reasoner which finds that $\mathcal{O} \models \eta$ is correct with respect to η . Unfortunately, the mere fact that some proposed \mathcal{J} is conclusively determined *not* to be a justification, does not show that η is a nonsubsumption. It merely shows that \mathcal{J} is not a justification and that the reasoner that extracted it has a bug. Even if we extracted “all” putative justifications for each reasoner and conclusively rejected them all, it would still be possible that all the reasoners were similarly generating spurious justifications. Of course, it seems rather unlikely that this would happen, but this is just a yet more detailed voting scheme. Of course, analysis of the spurious justifications might lead to a hypothesis about the reasoner’s buggy behaviour which then leads to a resolution. If reasoners would present putative witness countermodels, then we could get conclusive evidence for the nonsubsumption. However, non-entailment explanation is currently a wide open problem.

There are some cases where verifying the justification is automatable. For example, if the justification is a self-justification, that is, $\mathcal{J} = \eta$ then we can just check whether $\eta \in \mathcal{O}$. Of course, this would be a very odd case for a reasoner to miss, but as we will see below, it does happen.

It is well known that sets of justifications exhibit various structures and commonalities that mean that principled comprehension of the entire set can be achieved more efficiently than by individual examination of each justification in turn. As the infrastructure for these techniques is not readily available, for the purposes of this study we experiment with some ad hoc versions. It is part of future work to provide proper support.

4 Experimental Design

Note that datasets and supporting materials (including a description of an earlier version of the experiment) are available at <http://bit.ly/1Gzi7PB>.

4.1 Reasoners, and Machines, and Corpus

For the experiment the OWL API (v. 3.5.0) implementations [9] of four state of the art reasoners were used: FaCT++ 1.6.4 [16], JFact 1.2.3, Pellet 2.3.1 [13] and HermiT 1.3.8 [6].

We ran the experiment on 6 Amazon Web Service r3.large instances, 15.25 GB RAM, memory-optimised, Intel Xeon E5-2670 v2 CPU @ 2.5 GHz with a timeout of 5 hours for the overall process (including 80 minutes for each individual classification). 22 ontologies were not successfully processed: 15 failed due to timeout, 4 ran out of memory, 2 had a stack overflow (Java) and 1 caused a segmentation fault.

Corpus From a corpus of 339 BioPortal ontologies, we filtered out those that were not valid OWL DL (53), had TBoxes smaller than 50 axioms (+26), or were merely RDFS (+47) or AL (+1). This left us with 212 ontologies. Out of the 212, 190 were successfully processed according to our method in Section 3. Every process was run in a fresh Java 8 Virtual Machine, and involved generating, for 1 ontology, inferred hierarchies by four reasoners (Pellet 2.3.1, HermiT 1.3.8, JFact 1.2.3, FaCT++ 1.6.4 (snapshot)) and generating and verifying their justifications for all not agreed-upon entailments. We explicitly allowed individual reasoners to fail generating their hierarchies as long as at least two reasoners succeeded.

Identifying and Classifying Problem Cases. Of the 190 successfully processed ontologies, 181 had complete agreement at Class Hierarchy level. For the remaining 9 ontologies, we generate and verify justifications using the OWL Explanation Framework [8] to generate explanations.

Each explanation is stored in .owl format allowing us to reload them for the purposes of checking them against the reasoner and so that if needed, we can evaluate the file directly. We also generate a human readable version of the justification in DL Syntax, split into the ABox, TBox and RBox.

5 Results

The ontology level agreements are summarised in Table 1. The first thing to note is that the reasoners exhibit a great deal of agreement: they concur on 181 of the ontologies while disagreeing on only 9. While significant, it is not evidence that the current suite of reasoners are *wildly* buggy. While nearly half of the ontologies occur in polynomial profiles, all of the bug witnesses are in OWL DL. 8 out of 9 also contain datatypes. This conforms to the expectation that the more complex (i.e., OWL DL handling) or under tested (i.e., datatype handling) parts of reasoners are more likely to be buggy. For the rest of the rows the key thing to notice is that for most of them, the bug witnesses are smaller than either the successful and agreed upon cases or the timeouts (the timeouts are significantly larger in general).

Table 2 shows key statistics about the 9 problem ontologies. The first interesting point is that FaCT++ and JFact do disagree in spite of sharing a common code lineage — JFact was produced by a translation of FaCT++ from C++ to Java and the implementation follows FaCT++ to some degree.

Table 1. Filtered corpus statistics.

| | Agreement | Disagreement | Processing failed |
|-------------------------|-----------|--------------|-------------------|
| Ontologies | 181 | 9 | 22 |
| In a Polynomial Profile | 89 | 0 | 4 |
| Contain datatypes | 53 | 8 | 5 |
| TBox (mean) | 8,833 | 1,287 | 266,674 |
| TBox (max) | 415,494 | 3,547 | 2,249,883 |
| ABox (mean) | 680 | 1,331 | 10,082 |
| ABox (max) | 89,292 | 11,575 | 220,948 |
| Nr. Classes (mean) | 3,643 | 651 | 92,369 |
| Nr. Classes (max) | 110,717 | 1,851 | 517,023 |
| Nr. Object Prop. (mean) | 36 | 64 | 134 |
| Nr. Object Prop. (max) | 463 | 189 | 950 |
| Nr. Data Prop. (mean) | 5 | 13 | 2 |
| Nr. Data Prop. (max) | 117 | 37 | 11 |
| Nr. Individuals (mean) | 237 | 204 | 35 |
| Nr. Individuals (max) | 40,069 | 1,605 | 634 |

Table 2. Description of the problematic cases. CLS, OP, DP, and IN stand for number of classes, object properties, data properties and individuals, respectively. Coal. stands for ontology level agreement coalitions. Dis. stands for the number of entailment level disagreements. The last 4 columns list the number of axioms in the generated class hierarchy by a specific reasoner, either **FaCT++**, **HerMiT**, **JFaCT**, or **Pellet**. The blank entires indicate that the specific reason either timed out for this case, or rejected it due to not understanding a datatype (only FaCT++).

| \mathcal{O} | TBox | ABox | Expressivity | CLS | OP | DP | IN | Coal. | Dis. | F | H | J | P |
|---------------|------|-------|--------------|------|-----|----|------|-------|------|-------|-------|-------|-------|
| bco | 599 | 25 | SROIF(D) | 136 | 189 | 16 | 24 | 2 | 8 | 571 | 571 | 571 | 563 |
| cao | 442 | 0 | SHIQ(D) | 204 | 35 | 2 | 0 | 3 | 160 | 1249 | 1379 | 1249 | 1355 |
| dikb | 648 | 12 | ALCHOIN(D) | 125 | 70 | 37 | 9 | 2 | 5 | 282 | 282 | 279 | 282 |
| gro | 955 | 7 | ALCHIQ(D) | 507 | 24 | 6 | 4 | 2 | 3 | 3269 | 3270 | 3269 | 3270 |
| heio | 295 | 11575 | ALCHIF(D) | 124 | 11 | 37 | 1605 | 2 | 14 | 172 | 172 | 193 | 172 |
| nemo | 2686 | 182 | SHIQ(D) | 1851 | 89 | 4 | 120 | 2 | 1574 | 14665 | 14665 | 16305 | 14665 |
| obcs | 1126 | 33 | SROIQ(D) | 629 | 36 | 6 | 22 | 3 | 75 | | 4122 | 4055 | 4116 |
| obi.bcgo | 3586 | 57 | SROIN(D) | 1673 | 71 | 1 | 38 | 2 | 22 | | 16840 | 16818 | |
| stato | 1678 | 85 | SROIQ(D) | 609 | 48 | 4 | 13 | 2 | 44 | | 4102 | 4062 | |

However for gro, JFact and FaCT++ form a coalition against HerMiT and Pellet. In the ORE majority voting scheme, this would go to a coin toss, although the verdict should be weighted toward HerMiT and Pellet, as they are completely independent implementations of different underlying calculi. It is also worth noting that the entailment level differences are quite small, esp. on a percentage basis. These are not cases of large, easily detectable problems.

Table 3 shows the breakdown of cases into our four categories for each reasoner. Even this high level view is informative. Consider cao where there are three coalitions (FaCT++ and JFact vs. HerMiT vs. Pellet), so majority vote picks FaCT++ and JFact. However, FaCT++, JFact, and Pellet all have significant numbers of Definite Bugs (270, 270, and 72 resp.) while HerMiT has

none. Clearly, just because FaCT++ and JFact have (or seem to have) the same underlying bug is not a good reason to let them win! Similarly, for gro, we have two coalitions (FaCT++ and JFact vs. HerMiT and Pellet). Thus, majority voting would flip a coin, but FaCT++ and JFact are known buggy here, while the other coalition is not. We clearly should prefer the HerMiT and Pellet coalition. JFact has 12 Definite Bugs for dikb while the rest of the reasoners have none. This suggests that our efforts are best spent understanding why JFact fails to find those entailments during classification. Care must be taken with the last three rows as FaCT++ did not provide a class hierarchy for them and Pellet did not for the last two. Thus, all of their cases for those ontologies will have 0 for finding that $\mathcal{O} \models \eta$. Hence their verifying the corresponding justification might not indicate a bug, but that they would have reasoned correctly on that case if they had not failed to complete the classification. This suggests that either our procedure should be slightly modified or (better) that our case categories be. In any case, with only a bit more information, we are able to make more informed judgements about how to resolve disagreements as well as steer the manual investigation.

Table 3. Classification of all cases where the reasoner extracting the justification is different from the reasoner testing the justification, grouped by reasoner. DB = Definite bug, CY= Consistent Yes, PB = Possible Bug, and CN = Consistent No. Note that HerMiT has no cases in DB so that column was omitted.

| \mathcal{O} | FaCT++ | | | | HerMiT | | | JFact | | | | Pellet | | | |
|---------------|--------|-----|----|------|--------|----|------|-------|-----|----|----|--------|-----|----|------|
| | DB | CY | PB | CN | CY | PB | CN | DB | CY | PB | CN | DB | CY | PB | CN |
| bco | 0 | 16 | 0 | 0 | 16 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 | 24 |
| cao | 278 | 72 | 0 | 0 | 220 | 0 | 0 | 278 | 72 | 0 | 0 | 72 | 148 | 0 | 0 |
| dikb | 0 | 10 | 2 | 1 | 10 | 2 | 1 | 12 | 0 | 0 | 0 | 0 | 10 | 2 | 1 |
| gro | 6 | 1 | 0 | 0 | 5 | 1 | 2 | 6 | 1 | 0 | 0 | 0 | 5 | 1 | 2 |
| heio | 0 | 0 | 0 | 14 | 0 | 0 | 14 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 14 |
| nemo | 0 | 38 | 0 | 1273 | 38 | 0 | 1273 | 0 | 0 | 15 | 39 | 0 | 38 | 0 | 1273 |
| obcs | 167 | 0 | 0 | 5 | 111 | 4 | 1 | 150 | 12 | 4 | 1 | 16 | 99 | 0 | 0 |
| obi_bego | 52 | 0 | 0 | 0 | 38 | 0 | 0 | 52 | 0 | 0 | 0 | 38 | 0 | 0 | 0 |
| stato | 109 | 0 | 0 | 4 | 84 | 2 | 2 | 101 | 0 | 0 | 0 | 84 | 0 | 0 | 4 |
| SUM | 612 | 137 | 2 | 1297 | 522 | 9 | 1293 | 599 | 101 | 19 | 40 | 210 | 300 | 3 | 1318 |

5.1 Justification Analysis

A total of 1,622 distinct purported justifications were extracted from these 9 ontologies, which is a daunting number, but not inherently infeasible to survey. Furthermore, Table 2 showed that 7 of the problem ontologies had less than 100 suspect entailments (5 less than 50 with several having very few). Thus, resolving the disagreement w.r.t. most ontologies is a much easier task.

In addition to the total number, the verification effort is determined by their difficulty which is often proportionate to their size (Table 4). 99% of all justifications have less than 9 axioms in them, which is typically quite manageable. The min being 1 is not inherently surprising, but the fact that 13% (204) of justifications are of size 1 is a bit surprising. (We do further analysis of those

cases below.) These results suggest that crowdsourcing justifications verification (or, at least, sufficient elbow grease) is quite feasible (though perhaps not for real time applications like live competitions). Due to time constraints we do not attempt to verify all the justifications, but we did attempt some in order to see how feasible it is.

Table 4. Justification size distribution

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | 99% | Max. |
|------|---------|--------|------|---------|-----|--------|
| 1.00 | 3.00 | 3.00 | 3.33 | 4.00 | 9 | 100.00 |

A-Self Justifications. In principle, such justifications can require at least a bit of reasoning (e.g., $A \sqsubseteq (B \sqcap C)$), but all 204 such justifications in this experiment are *self* justifications, that is where $\mathcal{J} = \{\eta\}$. This is quite surprising since this means that the missing η is asserted in \mathcal{O} and if a reasoner should get any entailments right it should find all the asserted axioms. Indeed, we can (and do!) verify self justifications simply by checking whether $\eta \in \mathcal{O}$. Thus, all 204 are verified automatically.

For 378 *cases* in Definite Bug, the justification in question was a self justification which resolves the issue of *where* the reasoner error is (i.e., in the classification or in testing the justification). These cases occur for FaCT++ and JFact. We reported such problems discovered in an earlier round of experimentation (see <http://bit.ly/1Gzi7PB>) to the developer of FaCT++, who accepted the reports and attempted a fix. The current experiment used this version, but still found FaCT++ Definite Bug cases with self-justification. In personal communication with the developer, we found that the problem seems to be in the interaction between the OWL API and FaCT++ (notably, the command line version of FaCT++ does not exhibit these problems).

Note that fixing these missing entailments has a cascade effect as other justifications which depended on those lost axioms now are verified by the buggy reasoners (and, indeed, all those dependent entailments are found during classification).

This case is interesting because 1) testing for the correctness of checking the entailment of asserted axioms is not an obvious testing move 2) it highlights that problems with reasoners may stem from outside the reasoner proper, and 3) it highlights the need for analysis of the justificatory structure.

B-Manual Inspection: Pellet and BCO. We selected the 24 Consistent No cases for Pellet against the bcp ontology because Pellet was in conflict with all the other reasoners. There were 8 distinct justifications in the 24 cases, thus each non-Pellet reasoner extracted and verified each of the 8. All 8 were rather similar structurally. For example, one purported justification for $A \sqsubseteq B$ was:

$$\begin{aligned}
A &\sqsubseteq \exists S.D \\
D &\sqsubseteq \exists U.(B \sqcap \exists R.C) \\
U &\sqsubseteq P \\
S \circ P &\sqsubseteq Q \\
\exists Q.\top &\sqsubseteq B
\end{aligned}$$

All 8 of the justifications were verified independently by all the authors as correct explanations for their entailment. Thus, we verified that Pellet was incorrect.

C-Manual Inspection: FaCT++, JFact and Unsatisfiable Classes. We found two justifications produced by FaCT and JFact that asserted a particular class in the gro ontology was unsatisfiable. Since debugging unsatisfiable classes are a classic explanation target, we decided to verify them. Examination of the justifications directly showed that the class Decrease was being classified as unsatisfiable, because of the specifications of the data type.

Decrease had for its data property *polarity* a specified datatype of `rdf:PlainLiteral`. This was opposed to the specified range of the polarity datatype, which were all `xs:string`. The actual possible value was correct, but the types differed. If the types are, in fact, different, the purported justifications are correct. However, according to the W3C specifications on plain literals, such a substitution was allowed: `rdf` plain literals should be interpreted as `xs:strings` [1]. Consequently, “negative” cast as `plainLiteral` should be accepted as a `xs:string`. “negative”^{^^xsd:string} denotes the same as the plain literal “negative”. This showed that JFact and FaCT++ had problems with particular data types.

To test this, we created a minimal type casting test to run the reasoners on. We created a simple ABox and RBox:

```

a:R value “x”
a:R value “x”^^string
Functional(R)

```

Since no individual can have more than one R successor in this ontology, it is consistent just in case the plain literal “x” can be cast to a string. This was inconsistent for FaCT++ and JFact, but not Pellet or HermiT. This test case clearly shows that FaCT++ and JFact are incorrect. This example suggested other similar examples, such as:

```

a:R value “x”^^rdfs:Literal
a:R value “x”^^string
Functional(R)

```

This was found to be inconsistent by Pellet while it crashed FaCT++.

This provides specific evidence to diagnose a data type error within FaCT and JFact (and incidentally, Pellet).

D-Manual Inspection: Heio and JFact. To accumulate evidence for or against a contested assumption, Consistent Yes and Consistent No cases tend to cancel each other out. If we have only a single case (in either category) we don't have any reason to contradict the case. However, if we have a balanced set of Consistent Yeses and Nos (i.e., half of each), then we have no information to choose between them. Contrariwise, an unbalanced set of Consistent Yeses and Nos tends to verify the cases in the majority. In the case of heio, there were 14 Consistent Yeses for JFact, and 14 Nos for each of the other reasoners, and JFact was the only reasoner to produce a justification. In this case, the smart money is against JFact. Given that there were only 14 justifications to verify and they were all of size 2, we decided to completely verify this set.

The justifications were all structurally similar:

$$\begin{aligned} A &\equiv C \sqcap \exists P.\text{integer}[>3] \\ B &\equiv C \sqcap \exists P.\text{integer}[\leq 3] \end{aligned}$$

In each case, two classes (e.g., A and B) are modelled as non-equivalent by having incompatible ranges as the value of some datatype property. Clearly, these are not real justifications for $A \sqsubseteq B$. Moreover, JFact will infer from such justifications (and thus from heio) that $A \equiv B$. Clearly, JFact is not appropriately coping with the data range facets.

A Modest Generalisation. While we exhausted all the self justifications (A), the remaining case studies are suggestive: B involves role chains, while C and D involve datatypes. Both these features are comparatively new (in their current form) as of OWL 2 and thus comparatively little used or tested. Of all our justifications, 1363 involve a datatype, of those 1347 use facets, while only 24 use role chains. (Those 24 were fully verified.) As FaCT++ and Pellet both rejected ontologies on the basis of datatypes, we have further confirmation that datatypes need special attention. Modellers using elaborate datatype modelling would be well advised to test their ontologies against a set of reasoners, not just one, and compare the results.

6 Discussion

One important consequence of our justification based method is its implications for crude methods such as majority voting. Recall that majority voting considers if there is total, partial or no consensus in the inferred class hierarchies. In the case of partial consensus, any given reasoner is correct if it agrees with the majority. In the case of no consensus, some tie-breaking method is applied (this can be as simple as flipping a coin).

In two of the cases we found reasons to suspect that the majority (or lack of) might lead to wrong reasoners being classified as “correct”. Moreover, we could produce information to justify the choices picked through our method and those cases where our method agreed with MV. The evidence shows a clear difference between the method stipulated and MV.

With respect to the reliability of MV, we have produced evidence to suggest that its reliability is questionable. Given that we can find a clear instance of a tie-breaking method picking from a set of reasoners that are suspected to exhibit problems, this undermines MV's effectiveness. We believe that a more granular voting system paralleling some of our analyses above would produce a more satisfactory mechanisms. If a contestant wanted to dispute the voting results, they would have the comparatively easy task of verifying some justifications.

It is clear that the system is informative with respect to how the errors are produced by the reasoners. Although we were unable to diagnose the source of each error in every evaluation, we were fairly successful for comparatively minor effort. In particular, we were able to generate minimal test cases with detailed explanation of the erroneous behaviour very suitable for bug reports. Our initial bug reports have been well received. Providing this information and a strict methodology to narrow down these minimal patches should provide basis for future work. It can also be seen that for Ontology Engineers, we are either able to provide effective minimal patches to the Ontology, to allow one to work around the problem with respect to reasoning, or provide an indication that no such work around is available such as in the case of FaCT++ and JFact handling of some self-justifications.

A general shortcoming of this system is that it does not catch all errors generated by the reasoners. We will not catch errors where the reasoners are all in agreement. Either they all infer an entailment that is not correct or they all fail to infer an entailment that should follow from the ontology. Against this, we argue that as this test can be performed with multiple reasoners, it is unlikely that this will occur. The more reasoners that are in accord with each other, the more confidence we may have that such inferred entailments are correct.

Another potential source of problems might be our restriction to a single justification per reasoner entailment pair. There are only two cases where this could be a problem: (1) The reasoner generates some justifications and some non-justifications for a given entailment and (2) a reasoner is able to verify some, but not all justifications. Without formally verifying this conjecture, we have not encountered any cases like this in some preliminary experiments. It might be necessary to rule this problem out in the future with more experiments. However, the restriction does not seem to hurt us with any of the examples we have pursued.

7 Related Work

Our initial motivation was the problem of reasoner disagreement in the context of ORE. We found Majority Voting to be a very unsatisfying disagreement resolution especially when it degrades to random choice. In prior ORE competitions known approximate reasoners, such as TrOWL, have been deployed over logic levels for which their incompleteness is a known fact (so for instance TrOWL approximates OWL DL by converting statements into a level of expressivity lower than OWL DL). The balance between approximation and speed is

an understandable one and it is unlikely that incomplete reasoners will ever form a majority for MV, as most reasoners within these competitions are complete. However, a method to adjudicate against this possibility is very useful. Moreover, within the competition, it would be useful to know the degree to which an automated reasoner is incomplete with respect to the majority (of presumably complete reasoners), because this can allow us to discern the degree of penalisation to those incomplete reasoners. Finally, a key goal of ORE is to improve reasoners. While it has proved helpful in pushing the performance bound, correctness is similarly important.

Justifications have been used in benchmarking, where their analysability was held up as a virtue. In order to verify that the justifications being used as benchmarks were reliable [4, p.38], every justification generated was checked against every reasoner. Additionally for completeness purposes, the authors use a number of reasoners to generate all entailments and their justifications. They note that this "...is very time consuming and infeasible for some ontologies". This problem with generating justifications means it is necessary to have some form of hard limit on generation in order to ensure the process is not overly time consuming.

Similar work for automated reasoners has been performed with respect to queries [14]. The authors create test units (A-Boxes representations of the query) to form a test base. For certain benchmark ontologies that are used to assess reasoners, this provides a metric that evaluates the completeness of the reasoners. Importantly they also stress the need for such methods to be invariant or independent of the ontology being tested against. Our work is complementary to this. [14] is evaluating an expected feature of the reasoners, in so much as the implementations of them are incomplete in order to have a satisfactory level of efficiency. By comparison, our method is concerned with error, with unexpected levels of incompleteness. It is also important to note that our method fulfils the invariant condition that [14] stipulate. This method can be used with any corpus of ontologies.

In general in Automated Reasoning there are a variety of techniques deployed for debugging. We note however, that a good deal of the techniques used as detailed in the literature refer to debugging of SAT or SMT solvers, rather than any reasoner that underlies semantic web technology. For instance, fuzzing is used by Brummayer to debug SAT and SMT solvers [5]. Briefly stated fuzzing is the use of deliberate random input for bug generation and is typically used in software engineering. Moreover, there exist libraries of test cases such as "The Thousands of Problems for Theorem Provers" (TTFP) for First Order Logic and typed higher order formed logic [15]. TTFP helpfully contains solutions to these problems, allowing reasoner developers to verify the output of their solvers. The library of problems forms a benchmark for solver developers as well as a nexus for the community as a whole.

8 Conclusions and Future Work

In this paper, we have presented a justification based method to identify bugs in OWL reasoners. Furthermore, our method allows us to narrow down possible sources of bugs, providing a starting point for reasoner debugging. Ideally, we would like to ensure that reasoners are as correct as possible for key reasoning services such as classification to avoid confusions caused by wrongly missing or spurious entailments.

A limitation of the work is that it is blind in terms of exhaustiveness of error. That is, we have produced evidence to show that there is something wrong with particular reasoners on particular ontologies, axioms and entailments. However, we do not know the extent to which this is exhaustive of erroneous behaviour. While this is not within the scope of the work, it means that we must apply a conservative mindset to the results of the method. It is not that the method has shown that reasoning software is in fact sound and complete. It has only shown that it has certain errors and certain agreements.

It is known that there is a good deal of diversity between reasoner implementation, in terms of code base, calculus and use of modularity and that this does not even take into account differences in degrees of completeness. We may naively assume that such a diversity influences the effectiveness of our method, in the sense that multiple pieces of complex software that agree should grant a certain confidence in that result. However, we note that a certain degree of care needs to be made when making claims regarding the diversity of software acting as a guarantee of reasoner soundness and being a principle factor in the effectiveness of the method. This is largely because of prior research on N Version Software Development. N Version Software Development, is the concept that a way of developing reliable software is to have multiple parallel developments attempt to produce the same piece of software. Certain communication restrictions are placed on the programmers developing the software in parallel, with the intention of enforcing a diversity of methods in its creation and hence a reliability in the overall product, when they are evaluated against each other.

There are parallels with our method - we are using multiple reasoners against each other, each from separate developers. This means that certain criticisms that were made of N Version Software Development need to be kept in mind. These principally come from [10]. They test the idea that the failures that occur within strictly separated programmers (in their case, students) can be considered as independent events, that is that failure does not occur in correlation (note that they do not specify how the failure occurs). The authors conclude that such a thing is statistically unlikely to be independent. They are keen to stress that this conclusion only applies to the application they used in their experiment. While there is no straightforward application of their results in this scenario, what can be learnt is a measure of caution about how diverse we may assume the reasoners are and whether diversity of implementation itself is playing a key role in our method. This is a possible avenue for further research. Our situation is even less restricted than the one that took place in their experiment, given the vitality of the semantic web community as a whole.

Our process involves a great deal of duplication of human evaluation work, in situations where justifications are duplicated, either for the same entailment or in the cases involving unsatisfiable classes, offered elsewhere as justifications for an aspect of the class hierarchy. Compounding this problem are situations where malign justifications are generated for correct entailments. Moreover, it can be seen that with axiom swallowing, justifications for one entailment can be subsets for another justification for a different entailment (if a reasoner misses an explanation for entailments, it may miss it for situations where the entailment is used as a step elsewhere). Hence, knowing how justifications interact either as subsets or in duplication and being able to collapse these cases down into a single human readable case would make the process of analysis far more efficient.

This provides an intention for future work. We could use tools for diffing, isomorphism detection, detecting root and derived justifications and pattern detection. Deployment of anyone of these tools could greatly speed up the analysis: focusing on root justifications, for example, reduces both the number of justifications one needs to inspect and their average size. Isomorphism could easily show that a similarity of pattern has occurred across justifications. At the moment these tools are not integrated together into a suite of tools. An ideal end goal would be to produce such a suite alongside an implementation of our method to allow end users to test reasoners and ontologies for bugs and then analyse such results.

In the future, we would like to provide a web service supporting our methods. Developers would be able to test their reasoners against a set of standard reasoners and then obtain cases that pinpoint potential bugs consisting of justifications, missing entailments and ontology patches (that make the problem disappear). Ontology engineers would be able to verify that their ontologies are treated consistently by all reasoners. In both cases, we would attempt to facilitate crowdsourcing of the justification verification task.

Furthermore, standard testing methodologies such as test case minimisation and mutation seem worth exploring in this context. While independently useful, they might also complement the other comprehension tools.

Perhaps the most important extension is the generation of a potentially conclusive witness for non-entailment. The classic technique for non-entailment explanation is the generation and display of counter models. Current implementations, being mostly of a model construction flavour, seem well suited for this. However, the structures that reasoners use internally are not directly a model, nor, in complex logics, are they particular close to sharable structures like aBoxes (which can completely capture certain sorts of model and then be tested by other reasoners). Furthermore, the models (and internal structures) can be quite large and complex potentially defying human inspection.

However, our reasoner correctness task provides a good test scenario for non-entailment explanation. One of the great barriers to such research is the difficulty of finding “interesting” non-entailments to experiment with. Our technique generates interesting non-entailments with no need for domain or ontology familiarity and with a strong motivation for understanding them.

References

1. W3C Recommendations RDF Semantics (2004). [Online; accessed April 15, 2015]
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook*, 2nd edn. Cambridge University Press (2007)
3. Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodríguez-Muro, M., Slusnys, M., Xiao, G.: The ontop framework for ontology based data access. In: Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., Pan, J.Z. (eds.) *CSWS 2014. CCIS*, vol. 480, pp. 67–77. Springer, Heidelberg (2014)
4. Bail, S., Parsia, B., Sattler, U.: JustBench: a framework for OWL benchmarking. 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. 32–47. Springer, Heidelberg (2010)
5. Brummayer, R., Biere, A.: Fuzzing and delta-debugging SMT solvers. In: *Proceedings of the 7th International Workshop on Satisfiability Modulo Theories, SMT 2009*, New York, NY, USA, pp. 1–5. ACM (2009)
6. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 Reasoner. *J. Autom. Reasoning* **53**(3), 245–269 (2014)
7. Hedeler, C., Parsia, B., Brandt, S.: Estimating and analysing coordination in medical terminologies. In: *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, New York, NY, USA, May 27–29, 2014, pp. 357–362 (2014)
8. Horridge, M.: *Justification Based Explanation in Ontologies*. PhD thesis, University of Manchester (2011)
9. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. *Semantic Web* **2**(1), 11–21 (2011)
10. Knight, J.C., Leveson, N.G.: An Experimental Evaluation of the Assumption of Independence in Multiversion Programming. *IEEE Trans. Software Eng.* **12**(1), 96–109 (1986)
11. Rector, A.L., Iannone, L.: Lexically suggest, logically define: Quality assurance of the use of qualifiers and expected results of post-coordination in SNOMED CT. *Journal of Biomedical Informatics* **45**(2), 199–209 (2012)
12. Rogers, J.E.: *Development of a methodology and an ontological schema for medical terminology*. PhD thesis, University of Manchester (2004)
13. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *J. Web Sem.* **5**(2), 51–53 (2007)
14. Stoilos, G., Grau, B.C., Horrocks, I.: How incomplete is your semantic web reasoner? In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010*, Atlanta, Georgia, USA, July 11–15, 2010 (2010)
15. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning* **43**(4), 337–362 (2009)
16. Tsarkov, Dmitry, Horrocks, Ian: FaCT++ description logic reasoner: system description. In: Furbach, Ulrich, Shankar, Natarajan (eds.) *IJCAR 2006. LNCS (LNAI)*, vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
17. Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Turi, D., Stevens, R.: A little semantic web goes a long way in biology. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) *ISWC 2005. LNCS*, vol. 3729, pp. 786–800. Springer, Heidelberg (2005)

Introducing Defeasibility into OWL Ontologies

Giovanni Casini^{1,2,4}, Thomas Meyer^{3,4}, Kody Moodley^{4,5} (✉), Uli Sattler⁶,
and Ivan Varzinczak^{4,7}

¹ University of Luxembourg, Luxembourg City, Luxembourg

² Department of Philosophy, University of Pretoria, Pretoria, South Africa

³ Department of Computer Science, University of Cape Town,
Cape Town, South Africa

⁴ Centre for Artificial Intelligence Research, CSIR Meraka, Pretoria, South Africa

⁵ School of Mathematics, Statistics, and Computer Science,
University of KwaZulu-Natal, Durban, South Africa
kody.moodley@gmail.com

⁶ University of Manchester, Manchester, UK

⁷ Universidade Federal Do Rio de Janeiro, Rio de Janeiro, Brazil

Abstract. In recent years, various approaches have been developed for representing and reasoning with exceptions in OWL. The price one pays for such capabilities, in terms of practical performance, is an important factor that is yet to be quantified comprehensively. A major barrier is the lack of naturally occurring ontologies with defeasible features - the ideal candidates for evaluation. Such data is unavailable due to absence of tool support for representing defeasible features. In the past, defeasible reasoning implementations have favoured automated generation of defeasible ontologies. While this suffices as a preliminary approach, we posit that a method somewhere in between these two would yield more meaningful results. In this work, we describe a systematic approach to modify real-world OWL ontologies to include defeasible features, and we apply this to the Manchester OWL Repository to generate defeasible ontologies for evaluating our reasoner DIP (Defeasible-Inference Platform). The results of this evaluation are provided together with some insights into where the performance bottle-necks lie for this kind of reasoning. We found that reasoning was feasible on the whole, with surprisingly few bottle-necks in our evaluation.

1 Introduction

Reasoning with *exceptions* has been a major topic in AI since the 80s. Classical *monotonic* formalisms such as OWL, assume that represented knowledge is infallible and do not admit exceptions; such systems generally cannot accommodate the addition of new information which contradicts what is known. For example, if a monotonic system is told that “*Students do not pay taxes*” then, upon encountering an exception (a student who works), it will still conclude that this student is exempt from taxes [10].

Defeasible reasoning is concerned with the development of formalisms which are able to represent and reason with defeasible (non-strict) facts: “Typically,

students do not pay taxes” is the defeasible counterpart of “*Students do not pay taxes*”.

Key approaches for defeasible reasoning in KR formalisms have been through adaptations and combinations of the following systems: Circumscription [4], Default Logic [22], Negation as failure [15], Probabilistic logic [17] and Preferential reasoning [6, 8, 10].

The theoretical foundation of our work is a Description Logic (DL) [1] adaptation of the preferential reasoning approach by Lehmann et al. [16]. DLs form the logical underpinning of OWL and so our approach is applicable in this setting as well.

The motivation for focusing on the preferential approach is that it derives intuitive inferences using procedures that reduce to classical OWL reasoning. This gives the advantage of being able to use “off-the-shelf” OWL reasoners such as FaCT++ (owl.man.ac.uk/factplusplus) and HerMiT (hermit-reasoner.com), to perform defeasible inference. In particular, we have implemented a defeasible entailment regime called *Rational Closure* (RC) in our reasoner DIP (Defeasible-Inference Platform) [21]. This implementation is a variant of the one by Casini and Straccia [8].

However, there is a lack of insight into the expected practical performance of implementations such as DIP. A major barrier is the lack of tools for representing defeasibility in OWL, which in turn leads to the absence of naturally occurring data using defeasible features - the ideal candidates for testing performance. Currently, the majority of datasets for testing defeasible extensions of OWL, are automatically generated with the only mature attempt at a standardisation being LoDEN (loden.fisica.unina.it).

Our main goal in this paper is to take the next step from completely synthetic data, to a systematic approach for introducing defeasible features into naturally occurring ontologies that do not contain such features. We apply this approach to construct a dataset for evaluating our RC implementation in DIP. First, we introduce \mathcal{ALC} , the DL of choice for our implementation and a defeasible notion of subsumption that we introduce into the logic. We then give a concise description of RC for this logic and sketch the algorithms for computing the construction. Section 3 is the heart of the paper, here we detail a procedure for introducing *defeasible subsumption* into real-world ontologies and apply it to the Manchester OWL Repository to generate data for evaluating the performance of RC. We present the results and compare these with related work. Finally, we conclude by mentioning future work to be undertaken in the area.

2 Preliminaries

2.1 Description Logics

DLs are decidable fragments of first-order logic with a variety of applications, notably the formalisation of ontologies. They are very popular since they represent the logical underpinning of the Web Ontology Language (w3.org/TR/owl-features). In this paper we focus on \mathcal{ALC} , a representative member of the

family of DLs, although our algorithms are applicable to a wide class of DLs, in particular *SHIQ* [14].

Let $N_{\mathcal{C}} = \{A_1, A_2, \dots\}$ (resp. $N_{\mathcal{R}} = \{r_1, r_2, \dots\}$) be a finite set of *class names* (resp. *role names*) s.t. $(N_{\mathcal{C}} \cap N_{\mathcal{R}} = \emptyset)$. The language, \mathcal{L} , of complex classes is:

$$C ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C \mid \perp \mid \top$$

\mathcal{ALC} has a standard set-theoretic semantics defined in the provided reference [1].

A classical DL ontology consists of a TBox \mathcal{T} (and optionally an ABox \mathcal{A}), \mathcal{T} contains the terminology describing the domain of discourse, i.e., \mathcal{T} is a finite set of *inclusion axioms* $C \sqsubseteq D$; such an axiom is read as “ C is subsumed by D ”, that is, every individual that falls under the class C , falls also under the class D . \mathcal{A} is a finite set of *instance axioms* (called assertions) of the form $C(a)$ or $R(a, b)$, where the former represents that a is an instance of the concept C , and the latter that a is related to b via the role R . \mathcal{ALC} has a classical monotonic relation of entailment, and we use \models to indicate this standard entailment relation, i.e., $\mathcal{T} \cup \mathcal{A} \models \alpha$ indicates that all the interpretations satisfying all the axioms contained in \mathcal{T} and \mathcal{A} also satisfy the axiom α . There are efficient tools for deciding \mathcal{ALC} entailment. More details on DLs (\mathcal{ALC} in particular [1]) and the relationship between OWL and DL [9] can be found in the provided references.

2.2 An Algorithm to Compute Rational Closure in \mathcal{ALC}

RC has a series of desirable properties from a formal perspective: the consequence relation has a solid logical foundation, is characterised by a set of structural properties that should be satisfied by any nonmonotonic formalism [5, 16], and its computation can be reduced to classical monotonic decision steps.

The applicability of RC to \mathcal{ALC} is predicated on the ability to represent defeasible information. To model such information, we introduce a type of inclusion, i.e., a *defeasible inclusion* $C \sqsubset D$, which is read as “*Typically* an instance of C is also an instance of D ”, that is, if we know that an object x is in the set referred to by C , we can conclude that x is in the set referred to by D , unless we have knowledge to the contrary. For the semantics of such axioms, we refer the reader to the work by Britz et al. [5, 6].

We consider knowledge bases (KBs) of the form $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, where \mathcal{T} is a DL TBox and \mathcal{D} is known as a *defeasible TBox* (DTBox) which is a finite set of defeasible inclusions. We are not considering ABoxes here - the algorithm we introduce (based on the one by Casini and Straccia [8]), computes RC only considering classes. Given $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, it decides if $C \sqsubset D$ or $C \sqsubseteq D$ is a defeasible consequence of \mathcal{K} .

Example 1. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, with $\mathcal{T} = \{\text{BactMen} \sqsubseteq \text{Men}, \text{VirMen} \sqsubseteq \text{Men}\}$ and $\mathcal{D} = \{\text{Men} \sqsubset \neg \text{Fatal}, \text{BactMen} \sqsubset \text{Fatal}\}$. \mathcal{K} is about meningitis (Men), bacterial meningitis (BactMen), viral meningitis (VirMen), and their fatality (Fatal).

If all axioms in \mathcal{K} were classical inclusion axioms, we derive $\mathbf{BactMen} \sqsubseteq \perp$ because the facts lead to bacterial meningitis being both fatal ($\mathbf{BactMen} \sqsubseteq \mathbf{Fatal}$) and non-fatal ($\mathbf{BactMen} \sqsubseteq \mathbf{Men}$, $\mathbf{Men} \sqsubseteq \neg\mathbf{Fatal}$). We would rather “relax” some strict facts to cater for the atypicality of $\mathbf{BactMen}$ (meningitis is usually not fatal, but bacterial meningitis is an exceptional type of meningitis because it usually *is* fatal).

We shall indicate the set of *materialisations* of the axioms in \mathcal{D} by $\overline{\mathcal{D}}$, where the *materialisation* of an axiom $C \sqsupseteq D$ denotes the class expressing the same subsumption relation of the axiom (i.e., $\neg C \sqcup D$). Hence $\overline{\mathcal{D}} = \{\neg C \sqcup D \mid C \sqsupseteq D \in \mathcal{D}\}$.

The *classical translation* of $C \sqsupseteq D$ is $C \sqsubseteq D$. Similarly, for a set $\mathcal{D} = \{C_1 \sqsupseteq D_1, \dots, C_n \sqsupseteq D_n\}$, the classical translation of \mathcal{D} is $\mathcal{D}' = \{C_1 \sqsubseteq D_1, \dots, C_n \sqsubseteq D_n\}$.

The RC algorithm consists of a main procedure and two sub-procedures. The first sub-procedure is called **Exceptional**. Its aim is to determine which of the left-hand side (LHS) classes in our inclusions are *exceptional*. Intuitively, a class is exceptional in a KB if it is atypical w.r.t. one of its superclasses (e.g. $\mathbf{BactMen}$ in the example above). The exceptionality of a class can be decided using \models , since a class C is exceptional in $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$ if and only if $\mathcal{T} \models \bigcap \overline{\mathcal{D}} \sqsubseteq \neg C$. A defeasible axiom $C \sqsupseteq D \in \mathcal{D}$ is considered exceptional if its antecedent (LHS-concept) C is exceptional. Given a finite set \mathcal{E} of defeasible inclusion axioms, Procedure **Exceptional** gives back the subset of \mathcal{E} containing the exceptional axioms.

Procedure Exceptional(\mathcal{T}, \mathcal{E})

Input: $\mathcal{T}, \mathcal{E} \subseteq \mathcal{D}$

Output: $\mathcal{E}' \subseteq \mathcal{E}$ such that \mathcal{E}' is exceptional w.r.t. \mathcal{E}

- 1 $\mathcal{E}' := \emptyset$;
 - 2 **foreach** $C \sqsupseteq D \in \mathcal{E}$ **do**
 - 3 **if** $\mathcal{T} \models \bigcap \overline{\mathcal{E}} \sqsubseteq \neg C$ **then**
 - 4 $\mathcal{E}' := \mathcal{E}' \cup \{C \sqsupseteq D\}$;
 - 5 **return** \mathcal{E}' ;
-

Since, in general, there may be “exceptions-to-exceptions” (perhaps a strain of bacterial meningitis that is usually *not* fatal), we can compute this *exceptionality ranking* by recursive application of Procedure **Exceptional**. I.e., we associate a value to each axiom in the KB representing its degree of exceptionality. **ComputeRanking** is the second sub-procedure that, using **Exceptional**, partitions the set \mathcal{D} into $\mathcal{R} = \{\mathcal{D}_0, \mathcal{D}_1, \dots\}$, where each set \mathcal{D}_i contains the defeasible axioms having i as ranking value.

ComputeRanking receives KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$ as input and outputs an *equivalent* KB $\mathcal{K}^* = \langle \mathcal{T}^*, \mathcal{D}^* \rangle$ (i.e., satisfied in the same interpretations as \mathcal{K} [5]), but in which *implicit* strict knowledge contained in the DTBox has been moved to the TBox, and all the information in the DTBox has been ranked w.r.t. its exceptionality. We call axioms which are implicitly strict and yet concealed in the

Procedure ComputeRanking(\mathcal{K})

Input: KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$
Output: KB $\langle \mathcal{T}^*, \mathcal{D}^* \rangle$ and the partitioning (ranking) $\mathcal{R} = \{\mathcal{D}_0, \dots, \mathcal{D}_n\}$ for \mathcal{D}^*

- 1 $\mathcal{T}^* := \mathcal{T}; \mathcal{D}^* := \mathcal{D}; \mathcal{R} := \emptyset;$
- 2 **repeat**
- 3 $i := 0; \mathcal{E}_0 := \mathcal{D}^*;$
- 4 $\mathcal{E}_1 := \text{Exceptional}(\mathcal{T}^*, \mathcal{E}_0);$
- 5 **while** $\mathcal{E}_{i+1} \neq \mathcal{E}_i$ **do**
- 6 $i := i + 1; \mathcal{E}_{i+1} := \text{Exceptional}(\mathcal{T}^*, \mathcal{E}_i);$
- 7 $\mathcal{D}_\infty^* := \mathcal{E}_i; \mathcal{T}^* := \mathcal{T}^* \cup \{C \sqsubseteq D \mid C \sqsubset D \in \mathcal{D}_\infty^*\}; \mathcal{D}^* := \mathcal{D}^* \setminus \mathcal{D}_\infty^*;$
- 8 **until** $\mathcal{D}_\infty^* = \emptyset;$
- 9 **for** $j = 1$ **to** i **do**
- 10 $\mathcal{D}_{j-1} := \mathcal{E}_{j-1} \setminus \mathcal{E}_j; \mathcal{R} := \mathcal{R} \cup \{\mathcal{D}_{j-1}\};$
- 11 **return** $\langle \mathcal{T}^*, \mathcal{D}^* \rangle, \mathcal{R};$

DTBox, *totally exceptional* axioms. Consider the KBs: $\mathcal{K}_1 = \{C \sqsubset D, C \sqsubset \neg D\}$ and $\mathcal{K}_2 = \{E \sqsubset D, C \sqsubseteq E, C \sqsubset \neg D\}$. C is unsatisfiable w.r.t. \mathcal{K}'_1 and \mathcal{K}'_2 . In \mathcal{K}'_1 , the unsatisfiability does not indicate an exception, whereas in \mathcal{K}'_2 , it does. The former case is a knowledge engineering problem, indicating logical incoherence in defeasible KBs, while the latter is not considered such.

Sub-procedure [Exceptional](#) reaches a fixed point $\mathcal{E}_i = \mathcal{E}_i + 1$ (\mathcal{E}_i is possibly empty) during Procedure [ComputeRanking](#). If \mathcal{E}_i is not empty, then it represents a set of totally exceptional axioms, and we assign ∞ as the ranking value to each of these axioms (indicated by \mathcal{D}_∞^*). We move such information to the TBox (that is, if $C \sqsubset D$ is in \mathcal{D}_∞^* , we eliminate it from \mathcal{D} and add $C \sqsubseteq \perp$ to \mathcal{T}^*). We repeat the procedure (Lines 2-8) until *all* implicit strict facts in \mathcal{D} are moved to the TBox. Consider the example:

Example 2. Consider $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, with $\mathcal{T} = \{E \sqsubseteq D\}$ and $\mathcal{D} = \{F \sqsubset \exists r.C, C \sqsubset \neg D, C \sqsubset E\}$. Applying Procedure [ComputeRanking](#), we obtain that C is exceptional, i.e., $\mathcal{E}_2 = \mathcal{E}_1 = \{C \sqsubset \neg D, C \sqsubset E\}$. This means that $\mathcal{D}_\infty^* = \{C \sqsubset \neg D, C \sqsubset E\}$, and therefore, $\mathcal{T}^* = \{E \sqsubseteq D, C \sqsubseteq \perp\}$ and $\mathcal{D}^* = \{F \sqsubset \exists r.C\}$. Repeating Lines 2-8 of the procedure we get $\mathcal{E}_1 = \mathcal{E}_0 = \{F \sqsubset \exists r.C\}$, hence $\mathcal{D}_\infty^* = \{F \sqsubset \exists r.C\}$ and we end up with a TBox $\mathcal{T}^* = \{E \sqsubseteq D, C \sqsubseteq \perp, F \sqsubseteq \perp\}$ and an empty DTBox.

Once the procedure has identified $\langle \mathcal{T}^*, \mathcal{D}^* \rangle$, it ranks the remainder axioms in the DTBox (if any): an axiom has ranking value i if i is the highest label for which it turns out to be exceptional. The result is a partition of \mathcal{D} into $\mathcal{R} = \{\mathcal{D}_0, \dots, \mathcal{D}_n\}$.

Example 3. Consider \mathcal{K} in Example 1. [ComputeRanking](#) takes \mathcal{K} as input, executes Lines 2 - 8 to obtain the sequence $\mathcal{E}_0 = \mathcal{D}$, $\mathcal{E}_1 = \{\text{BactMen} \sqsubset \text{Fatal}\}$, $\mathcal{E}_2 = \emptyset$ and the TBox $\mathcal{T}^* = \mathcal{T}$. Finally, applying Lines 9 - 10, we obtain the partition \mathcal{D}^* of $\mathcal{D}_0 = \{\text{Men} \sqsubset \neg \text{Fatal}\}$, $\mathcal{D}_1 = \{\text{BactMen} \sqsubset \text{Fatal}\}$.

Given our computed ranking, we can ask queries of the form $C \sqsubset D$. Note that if we are confronted with a strict query (classical inclusion axiom $C \sqsubseteq D$), one can

Procedure RationalClosure(\mathcal{K}, δ)

Input: KB $\mathcal{K} = \langle \mathcal{T}^*, \mathcal{D}^* \rangle$ with no implicit strict facts, $\mathcal{E}_0, \dots, \mathcal{E}_n$, query $\delta = C \sqsubseteq D$.

Output: true iff $C \sqsubseteq D$ is in the RC of \mathcal{K}

```

1  $i := 0$ ;
2 while  $\mathcal{T}^* \models \prod \overline{\mathcal{E}_i} \sqcap C \sqsubseteq \perp$  and  $i \leq n$  do
3    $i := i + 1$ ;
4 if  $i \leq n$  then
5   return  $\mathcal{T}^* \models \prod \overline{\mathcal{E}_i} \sqcap C \sqsubseteq D$ ;
6 else
7   return  $\mathcal{T}^* \models C \sqsubseteq D$ ;

```

determine if it is in the RC of the KB by checking if it is classically entailed by the strict facts alone in \mathcal{K} (that is, \mathcal{T}^*). This was implemented as an optimisation.

However, for simplicity, Algorithm [RationalClosure](#) considers only the case in which the query is a defeasible inclusion axiom. The algorithm takes the ranking \mathcal{R} and query $C \sqsubseteq D$ as input, determines which portion of \mathcal{R} is compatible with the class C , i.e., which portion of defeasible knowledge does not imply that C is exceptional, starting from the most normal situations up to increasing levels of exceptionality. By example:

Example 4. Consider the ranking \mathcal{R} in [Example 3](#) and the query $\text{VirMen} \sqsubseteq \neg \text{Fatal}$. The RC algorithm checks if $\mathcal{T}^* \models \prod \overline{\mathcal{E}_0} \sqsubseteq \neg \text{VirMen}$, which is not the case. Hence, we have to check if $\mathcal{T}^* \models \prod \overline{\mathcal{E}_0} \sqcap \text{VirMen} \sqsubseteq \neg \text{Fatal}$, which is true. However, if our query is $\text{BactMen} \sqsubseteq \neg \text{Fatal}$, we obtain a different result: since $\mathcal{T}^* \models \prod \overline{\mathcal{E}_0} \sqsubseteq \neg \text{BactMen}$ but $\mathcal{T}^* \not\models \prod \overline{\mathcal{E}_1} \sqsubseteq \neg \text{BactMen}$, BactMen is an exceptional class of level 1, compatible with \mathcal{D}_1 , and we have to move one level higher in the ranking eliminating the facts in \mathcal{D}_0 from consideration. It turns out that $\mathcal{T}^* \not\models \prod \overline{\mathcal{E}_1} \sqcap \text{BactMen} \sqsubseteq \neg \text{Fatal}$, and that is the right conclusion since we have $\text{BactMen} \sqsubseteq \text{Fatal}$ in our KB.

The correctness of Algorithm [RationalClosure](#) follows from the procedure by Casini and Staccia [\[8\]](#) as it is a reformulation thereof. The computational complexity of the entire procedure is the same as that of the underlying monotonic entailment relation \models , i.e., it is an EXPTIME-complete problem ([\[5\]](#) and [\[8, Corollary2\]](#)). Moreover, note that the defined procedures can be applied to DLs more expressive than \mathcal{ALC} , still preserving the computational complexity of the decision problem w.r.t. the underlying monotonic entailment relation, and, is still sound and complete for logics up to \mathcal{SHIQ} . Using a more expressive DL than \mathcal{ALC} , the defeasible information will still be represented only by defeasible inclusion axioms $C \sqsubseteq D$, while the strict information different from \mathcal{ALC} inclusion axioms (role inclusion axioms, role transitivity, etc.) must be considered as background knowledge at each step of the decision procedure.

3 Performance Evaluation

An important question about RC is: how much do we pay for the additional expressivity, in terms of practical reasoning performance? We have shown that the worst case computational complexity of RC in \mathcal{ALC} is not higher than reasoning with classical \mathcal{ALC} . This is good news, but does not guarantee good performance in practice. As illustrated in our algorithms (Section 2.2), we have to perform some additional computation over and above the classical decision checks. In general, we perform multiple classical entailment checks to answer a single defeasible entailment question. The question is how much more work are we doing. We aim to investigate this in order to provide evidence of the feasibility of adding defeasible features to ontologies.

In terms of data, the norm until now has been the use of automatically generated ontologies with defeasible features (the most notable attempt at a benchmark of synthetic defeasible ontologies is LoDEN). Indeed, we have also used synthetic data in the past as a preliminary indicator of performance [7]. Naturally, there are obvious shortcomings with such an approach, such as possible biases in the ontology generation methodology. However, there is no question of finding *representative* data because there are virtually no naturally occurring ontologies with intended defeasible features.

We instead choose a middle-ground approach, taking advantage of the rich set of (classical) OWL ontologies on the Web in various repositories and corpora. The basic idea is to modify selected subsumptions in these ontologies to be *defeasible*, thereby making them useful as data to evaluate our defeasible reasoner. Of course, this has to be done with care to generate cases which are challenging for the reasoner. For example, we need to ensure that there are cases where there are multiple ranks in the ranking of the ontology (see Procedure [ComputeRanking](#)). Our method is described in Section 3.2, together with a discussion about its strengths and weaknesses. Now, we describe the curation process used for sampling our initial set of *unmodified* OWL ontologies.

3.1 Non-defeasible Dataset

For our initial data, we sample some classical OWL ontologies which we can later pass through our procedure for the introduction of defeasible features. The natural choice is to select the same data that is traditionally used to evaluate the performance of existing *classical* OWL reasoners. However, even in such a setting, there is no precise consensus on what data to use. The result is that data is generally curated manually by choosing “well-known” ontologies and corpora from which to sample, or arbitrarily selecting from the variety of respectable corpora on the web.

Choice of Corpora: While there are bona fide ontology benchmarks available such as LUBM [11] and its extensions, it was pointed out that there are shortcomings in manual selection of ontologies and ontology corpora for evaluation [18]. In particular, the main limitation with such selections is that they lack

sufficient *variety*. Thus the results of evaluations can be heavily biased towards the selected benchmarks. The Manchester OWL Repository [19] is an effort to address this issue. The Repository is a framework for sharing ontology datasets for OWL empirical research. The current version of the repository contains three core datasets, namely versions of NCBO Bioportal (bioportal.bioontology.org), The Oxford Ontology Library or OOL (cs.ox.ac.uk/isg/ontologies) and MOWL-Corp [18]. While Bioportal and OOL are established corpora actively used in reasoner evaluations, MOWL-Corp is a recent gathering of ontologies through sophisticated web crawls and filtration techniques [18].

We obtain a recent snapshot of the Manchester OWL Repository as the base dataset for our evaluation. There are 344, 793 and 20,996 ontologies in the Bioportal, OOL and MOWL-Corp corpora respectively.

Filtration Process and Choice of OWL Reasoner: For loading and analysing ontologies of our dataset, we use the popular and well-supported Java-based OWL API [13]. The API contains parsers for a wide variety of different syntaxes of ontologies such as RDF, Turtle and OBO. As we have shown in Section 2.2, our algorithms are built upon classical entailment checks. Thus, we would need to select an existing OWL 2 DL reasoning implementation to perform these classical entailment checks from within our defeasible reasoner. While running our evaluation with multiple reasoners would have been interesting, such an investigation is not necessary to ascertain the price we pay for reasoning with defeasible (in addition to classical) subsumption. We chose to utilise a single OWL 2 DL reasoner for our evaluation. In particular, we would ideally like to use the fastest and most robust implementation.

Consulting the latest results of the OWL Reasoner Evaluation Workshop (dl.kr.org/ore2014/results.html), we identified the top three OWL 2 DL reasoners for the standard reasoning tasks of: *classification*, *consistency checking* and *satisfiability testing* (in terms of performance and robustness). Robustness was measured as the number of ontologies that were successfully processed in the allotted time. The top reasoners were Konclude (derivo.de/produkte/konclude.html), HermiT, MORE (cs.ox.ac.uk/isg/tools/MORE), Chainsaw (chainsaw.sourceforge.net), FaCT++ and TrOWL (trowl.org). As we shall see in Section 3.2, we require to check incoherence of ontologies before introducing defeasible subsumptions into them. Modern OWL reasoners are optimised for classification (computing the subsumption relationship between each pair of class names in the ontology), and identifying unsatisfiable class names (incoherence) is usually performed by first classifying the ontology, and then “reading” the unsatisfiable class names from the results. Thus, we chose to focus on the reasoners which performed best in OWL 2 DL classification. These were respectively, Konclude, HermiT and MORE. Konclude, unfortunately, does not yet support the OWL API. Therefore, our choice was to select the next best reasoner - HermiT.

Given our choice of tools for manipulating and reasoning with the ontologies in our dataset, we filtered out the ontologies that could be loaded and parsed by the OWL API (each within an allotted 40 minutes). The resulting ontologies were then tested to determine if they were classifiable by HermiT within an

additional 40 minutes each. Ontologies which did not pass this test were removed from the data. In order to remove some of the cases which are very likely to be easy for our reasoner, we elected to remove ontologies with less than 100 logical axioms (ignoring annotations and other axioms carrying meta-information). This is justifiable because ontology size is proven to be an overwhelmingly dominant factor in reasoning performance [24]. Finally, we stripped the ontologies of ABox data because our defeasible reasoner is currently purely equipped with (D)TBox entailment procedures. This leaves us with 252, 440 and 2335 ontologies in Biportal, OOL and MOWLCorp respectively.

3.2 Defeasible Dataset

In this section, we describe a systematic technique to introduce defeasible subsumptions into ontologies, thereby making them amenable to defeasible reasoning evaluation.

Methodology: Our approach hinges upon an important correspondence between *class exceptionality* (as described in Section 2.2) and classical class unsatisfiability:

Lemma 1. *If a class C is exceptional w.r.t. a defeasible KB $\langle \mathcal{T}, \mathcal{D} \rangle$ then C is unsatisfiable w.r.t. $\mathcal{T} \cup \mathcal{D}'$, where \mathcal{D}' is the classical translation of \mathcal{D} .*

Lemma 1 states that if a class is exceptional in a defeasible ontology then it will necessarily be unsatisfiable in the classical translation of the ontology. This result is useful because we can use it to identify possible exceptional classes in classical ontologies. Taking the contrapositive of Lemma 1, we obtain the result that if a class is satisfiable w.r.t. a classical ontology then it is necessarily *not* exceptional w.r.t. any defeasible translation of the ontology. Therefore, we can eliminate ontologies from our dataset without LHS-classes of subsumptions that are unsatisfiable, because these could never become exceptional by turning classical subsumption into defeasible ones.

The following definition is a generalisation of standard incoherence to axioms with complex left hand side (LHS) concepts:

Definition 1. *A classical TBox \mathcal{T} is LHS-coherent if each $C \sqsubseteq D \in \mathcal{T}$ is s.t. $\mathcal{T} \not\models C \sqsubseteq \perp$. \mathcal{T} is LHS-incoherent if it is not LHS-coherent.*

Eliminating all ontologies from our dataset that are LHS-coherent leaves us with 11, 46 and 77 ontologies in the Biportal, OOL and MOWLCorp corpora respectively. Figure 1 provides some average properties of the ontologies in our dataset.

Thus, in total we have 134 ontologies for our performance evaluation. Now, the task is to relax some of the subsumptions of our ontologies to be defeasible. The obvious naïve approach to introducing defeasibility would be to convert *all* subsumptions to defeasible ones. Naturally, this is not likely to be the general approach of defeasible-ontology engineers in practice. The other extreme would

| Corpus | Classes | | | Roles | | | TBox size | | | RBox Size | | | Nested Classes | | Conjuncts | | Disjuncts | |
|----------|---------|--------|--------|-------|--------|-------|-----------|--------|--------|-----------|--------|------|----------------|-----|-----------|-----|-----------|-----|
| | avg | median | max | avg | median | max | avg | median | max | avg | median | max | max | avg | max | avg | max | avg |
| Biportal | 17992 | 422 | 187515 | 50 | 33 | 152 | 41309 | 754 | 439208 | 30 | 17 | 149 | 28 | 1 | 10 | 2 | 5 | 2.1 |
| OOL | 22203 | 16306 | 89926 | 55 | 44 | 194 | 41389 | 32928 | 142996 | 27 | 9 | 123 | 136 | 1 | 68 | 2 | 16 | 2.5 |
| MOWLCorp | 4621 | 216 | 89926 | 466 | 13.5 | 16586 | 8719 | 691 | 137021 | 214 | 5 | 7749 | 34 | 1 | 17 | 2 | 16 | 2.7 |

Fig. 1. Ontology metrics for the LHS-incoherent cases in the dataset.

be to develop an approach to identify the *minimal* (for some defined notion of minimality) amount of defeasibility to introduce into the ontology in order to successfully “cater for all the exceptions”. The latter approach would be ideal, and we are currently investigating such an approach; however, we propose that a reasonable approximation of such an “ideal” procedure yields meaningful data for *performance* evaluation. The approach that we discuss here is in the spirit of such an approximation.

Example 5. Consider the following TBox \mathcal{T} about different types of mechanics (Mech), general (GenMech), car (CarMech) and mobile (MobileMech):
 {1. $\text{Mech} \sqsubseteq \exists \text{hasWorkshop}.\top$, 2. $\text{Mech} \sqsubseteq \exists \text{hasSpecialisation}.\top$,
 3. $\text{MobileMech} \sqcup \text{GenMech} \sqcup \text{CarMech} \sqsubseteq \text{Mech}$, 4. $\text{MobileMech} \sqsubseteq \neg \exists \text{hasWorkshop}.\top$, 5. $\text{MobileMech} \sqcap \neg \exists \text{status.OnStandBy} \sqsubseteq \exists \text{hasWorkshop}.\top$,
 6. $\text{GenMech} \sqsubseteq \neg \exists \text{hasSpecialisation}.\top$, 7. $\text{CarMech} \sqsubseteq \exists \text{hasSpecialisation.Car}$ }

The classes MobileMech, GenMech and the class expression $\text{MobileMech} \sqcap \neg \exists \text{status.OnStandBy}$ are unsatisfiable w.r.t. \mathcal{T} . An intuitive analysis of \mathcal{T} tells us that the ontology engineer probably intended to model that mechanics *usually* have a workshop ($\text{Mech} \sqsubseteq \exists \text{hasWorkshop}.\top$) and *usually* specialise in certain types of equipment that they repair ($\text{Mech} \sqsubseteq \exists \text{hasSpecialisation}.\top$). This translation of Axioms 1 and 2 in Example 5, is a minimal and intuitive way to introduce defeasibility into \mathcal{T} , catering for exceptional types of mechanic - i.e., mobile and general mechanics.

However, we also have an exceptional type of mobile mechanic in \mathcal{T} (an “exception-to-an-exception”). That is, mobile mechanics who are no longer “on standby” or “on call” ($\text{MobileMech} \sqcap \neg \exists \text{status.OnStandBy}$). These mechanics would then be assigned a workshop for their repair tasks. To cater for such mechanics we would have to relax Axiom 4 of Example 5 as well, to express that mobile mechanics *usually* don’t have a workshop ($\text{MobileMech} \sqsubseteq \neg \exists \text{hasWorkshop}.\top$).

We now define a general *defeasible translation function* (DTF) for converting classical subsumptions to defeasible subsumptions in classical ontologies.

Definition 2. (DTF) Let \mathcal{T} be a set of classical subsumptions of the form $C \sqsubseteq D$, then $\mathcal{F} : \mathcal{T} \rightarrow \{C \sqsubseteq D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \mathcal{T}$ is a DTF for \mathcal{T} .

We also have to formalise what we mean when a particular DTF “caters for all exceptions” in the TBox. We call such a function a *safe* DTF.

Definition 3. (*safe DTF*) Let \mathcal{T} be a set of classical subsumptions, let \mathcal{F} be a DTF for \mathcal{T} and let \mathcal{D} be the special DTF that translates all subsumptions in \mathcal{T} to defeasible ones. Then, \mathcal{F} is a safe DTF for \mathcal{T} if C is totally exceptional w.r.t. $\mathcal{D}(\mathcal{T})$ if and only if C is totally exceptional w.r.t. $\mathcal{F}(\mathcal{T})$, for each $C \sqsubseteq D \in \mathcal{T}$.

We define a safe DTF that places a small upper bound on the subset of axioms to relax using the well-known notion of *justification* [12]. A justification for an entailment α of an ontology is a minimal (w.r.t. set inclusion) subset of the ontology that entails α . If we compute the justifications for $\mathcal{T} \models \text{MobileMech} \sqsubseteq \perp$ (concise reasons for *MobileMech* being unsatisfiable and possibly exceptional) we obtain a single justification $\{1, 3, 4\}$. Relaxing these axioms is sufficient for catering for mobile mechanics (in fact, it is only necessary to relax Axiom 1 as mentioned earlier). Similarly, we arrive at $\{2, 3, 6\}$ to cater for general mechanics and $\{4, 5\}$ for mobile mechanics no longer on call.

The basic idea is thus to take the union of the justifications for the unsatisfiable LHS-classes and relax these axioms to defeasible ones. We obtain that $\{1, 2, 3, 4, 5, 6\}$ should be relaxed in Example 5, which is admittedly a large proportion of our TBox. However, as we discover in Section 3.4, the proportion is much smaller in practice on larger real-world ontologies. However, while computing all justifications has been shown to be feasible in general on real-world ontologies, *black-box* (reasoner-independent) procedures are known to be exponential in the worst case [12]. To avoid this potential computational blowup, we obtain a small upper bound of the union of justifications by extracting a *star locality based module* [23] for the ontology in question, w.r.t. the set of unsatisfiable LHS-classes. A module of an ontology w.r.t. a signature (set of terms from the ontology) is a small subset of the ontology that preserves the meaning of the terms in the signature. We specifically choose star locality based modules because of two key properties: (i) they preserve all justifications in the ontology for all entailments (or axioms) that can be constructed with the given signature (*depleting* property [23, Section3]), and (ii) they are smaller in size relative to other modules which have the depleting property. The pseudocode of our procedure is given in Algorithm 1.

Algorithm 1. *relaxSubsumption*

Input: LHS-incoherent TBox \mathcal{T} , $\mathcal{C} = \{C \mid (C \sqsubseteq D \in \mathcal{T} \text{ for some } D) \wedge (\mathcal{T} \models C \sqsubseteq \perp)\}$

Output: Defeasible ontology $\langle \mathcal{T}, \mathcal{D} \rangle$

- 1 $\mathcal{T} := \emptyset$; $\mathcal{D} := \emptyset$; $\mathcal{M} := \text{extractStarModule}(\mathcal{O}, \text{sig}(\mathcal{C}))$; $\mathcal{T} := \mathcal{O} \setminus \mathcal{M}$;
 - 2 **foreach** $X \sqsubseteq Y \in \mathcal{M}$ **do**
 - 3 $\mathcal{D} := \mathcal{D} \cup \{X \sqsubseteq Y\}$;
 - 4 **return** $\langle \mathcal{T}, \mathcal{D} \rangle$;
-

Theorem 1. (*safety of our DTF*) Let \mathcal{F} be the DTF defined by Algorithm 1 and let \mathcal{T} be a set of classical subsumptions, then \mathcal{F} is a safe DTF for \mathcal{T} .

Discussion: There are two conflicting issues with the procedure we have presented for introducing defeasibility into OWL ontologies: (i) minimality of modification to the original ontology and (ii) the representative quality of the resulting defeasible ontology as something that might be built by an ontology engineer. While (i) and (ii) would be the ultimate goal for a methodology automating the introduction of defeasible features into OWL ontologies, our approach does not yet meet such desiderata. It is clear that the minimal axioms to relax in Example 5 would be $\{1, 2, 4\}$, yet we relax $\{3, 5, 6\}$ as well. On a related note, relaxing $\{1, 2, 3, 4, 5, 6\}$ does not capture a “natural” defeasible translation. For instance, it does not make sense, intuitively, to relax $\text{MobileMech} \sqsubseteq \text{Mech}$ (all mobile mechanics are mechanics) to $\text{MobileMech} \sqsubset \text{Mech}$ (*typical* mobile mechanics are mechanics). Such constraints should ideally remain strict.

Furthermore, a critical observation is that incoherence in classical ontologies may be caused by erroneous modelling. In ontology development tools, large emphasis has been placed on debugging incoherence by making modifications to the ontology to remove the “unwanted” entailments such as $C \sqsubseteq \perp$. This is likely to have prevented many developers publishing incoherent ontologies.

Given the above main shortcomings of our approach, we do not argue that our approach is the ideal methodology. Rather, we hope that it serves as a stepping stone from purely synthetic approaches to investigate and develop more suitable methodologies.

Hypotheses: Our general predictions for the evaluation are that (i) the ranking computation will be dramatically more performance intensive than testing entailment, (ii) entailment testing will be feasible for on-demand use and (iii) the number of incoherent LHS-classes (and the number of defeasible subsumptions) will affect the performance significantly, (iv) we anticipate the occurrences of totally exceptional LHS-classes to be rare and minimal, (v) since these cases also require recursive execution of the ranking procedure, we also anticipate such cases to be significantly harder for reasoning and (vi) in terms of the ranking of the defeasible subsumptions in the ontologies, we expect there to be not more than 2 levels of exceptionality (or 3 ranks in total). I.e., we expect exceptions-to-exceptions in the data, although we anticipate very few of these cases. We expect the majority of cases to have either no exceptions or 1 level of exception (2 ranks in total). Of course, we also predict a general trend of the higher the number of (logical) axioms in the ontology, the longer to compute inferences.

3.3 Experiment Setup

Our setup, methodologies and design choices for the experimental evaluation can be summarised as follows:

Data Summary: The input data for our experiments are 134 LHS-incoherent ontologies (curated as described in Section 3.1) from the Manchester OWL

Repository. The ontologies are divided across three corpora: 11, 46 and 77 in Biportal, OOL and MOWL Corp respectively. The average ratio of defeasible to strict axioms in each ontology is 8%, the median being 1.5%, the minimum ratio being 0.01% and the maximum being 98%. The DL expressivity distribution of the data ranges from variants of \mathcal{ALC} all the way up to \mathcal{SROIQ} . There are 35 DL variants in total represented in the data.

Additionally, we generated a set of entailment queries (defeasible subsumptions of the form $C \sqsupseteq D$ and strict subsumptions of the form $C \sqsubseteq D$) for each ontology to present to our defeasible reasoner. For the C 's, we focus on all LHS-incoherent classes in the ontology. The motivation is two-fold: (i) if we instead focus on C 's that are satisfiable, we would not require execution of Lines 2-3 of Algorithm [RationalClosure](#) in Section 2.2 because C could never be exceptional (see Lemma 1). Thus, we focus on incoherent C 's since these are the only ones which could possibly be exceptional and result in harder cases for reasoning. Instead of generating such incoherent C 's, we use the existing LHS-concepts that are unsatisfiable in the ontology as a preliminary strategy. Admittedly, generating incoherent C 's might also be interesting for future evaluation.

For the D 's we first take the \perp -syntactic locality module for the ontology w.r.t. to the signature of C (including \perp), and then take all nested class expressions present in the axioms of this module. The reason being that we want to preserve entailments over the signature of C in the module. We collect all LHS-incoherent classes C from the ontology and then collect all class expressions in the \perp -module for C to be the consequents D . We then test if $C \sqsupseteq D$ (and $C \sqsubseteq D$) is in the RC of the defeasible ontology. All our data is available for download in ZIP format (cair.za.net/ontologies).

Tasks: The first task is precompiling exceptionality rankings for each ontology in the dataset. Rankings are then stored on file for later use in entailment testing. It is important to note that the computation of the ranking is considered as an offline, precompilation process for each stable version of an ontology. Such a task is not meant to be executed on-demand during defeasible entailment tests. Lemma 1 is used as an optimisation in the ranking procedure. We only need check exceptionality of $C \sqsupseteq D$'s where C is unsatisfiable w.r.t. the classical translation of $\langle \mathcal{T}, \mathcal{D} \rangle$ (see Lines 2 to 4 of procedure [Exceptional](#) in Section 2.2).

The entailment tests are then performed on the stored rankings and results of both tasks are recorded. We recorded the average time it took to compute the rankings, and to answer entailment questions, with some additional metrics which we present in Section 3.4. For entailment tests, we made no use of any optimisation.

Equipment: The evaluation was carried out on an Intel Core i7 2.5Ghz processor running MacOSX 10.10. 8GB of memory is allocated to the Java Virtual Machine (Java version 1.6 is used). Hermit is the classical OWL 2 DL reasoning implementation.

Reproducible Steps: Assuming we have already obtained our dataset (set of ontologies) and generated set of queries for each ontology (as described in

the above data summary), the steps required to conduct the evaluation are as follows: (1) Compute the ranking (according to Procedure [ComputeRanking](#)) of each ontology, record the time required and store it to file (use optimisation described in Lemma 1 to avoid checking if satisfiable concepts are exceptional), (2) for each ontology, execute its set of queries on its stored ranking (using Procedure [RationalClosure](#)) and record timings.

3.4 Results and Analysis

The overall results for computing the rankings and testing entailment have proven to be extremely promising. Figure 2 gives an overview of some of the more pertinent results w.r.t. the computation of rankings.

| Corpus | TBox size | | | DTBox size | | | LHS-incoherent classes | | | Totally Exceptional classes | | Number of ranks | | Ranking time (s) | | |
|-----------|-----------|--------|--------|------------|--------|------|------------------------|--------|------|-----------------------------|-----|-----------------|-----|------------------|-------|--------|
| | avg | median | max | avg | median | max | avg | median | max | avg | max | avg | max | avg | max | median |
| Bioportal | 41309 | 754 | 439208 | 627 | 81 | 6010 | 477 | 13 | 4716 | 38 | 322 | 1.2 | 2 | 1705 | 18742 | 0.81 |
| OOL | 41389 | 32928 | 142996 | 415 | 39 | 7842 | 160 | 2 | 4018 | 2.7 | 122 | 1.6 | 2 | 139 | 3173 | 2.34 |
| MOWLCorp | 8719 | 691 | 137021 | 208 | 31.5 | 5002 | 76 | 6.5 | 1461 | 13.4 | 322 | 1.4 | 3 | 60 | 2070 | 0.16 |

Fig. 2. Ontology metrics and ranking computation results for the dataset.

Ranking Performance: Examining the ranking times in Figure 2, we notice that on average over the entire dataset, it takes 10 minutes to rank a single ontology. However, looking at the “median” column of the ranking shows the majority of rankings were computed in less than a second. There are just four “outlier” ontologies which breach the 2000 second mark, while the maximum ranking time for the remainder of the data is 1000 seconds. The reason is that these four ontologies have the most number of unsatisfiable LHS classes in the data (requiring more exceptionality checks). It must be stressed that the ranking computation is concerned with stratifying only the *defeasible* axioms in the ontologies. Therefore, in general, the ranking times increase with the number of defeasible axioms (see Figure 3a). A key insight is when the number of entailment checks *and* ontology size (number of subsumptions we are checking entailment w.r.t.) is maximised, then performance will be worst for these cases.

The most challenging cases, in theory, for our reasoner are those with totally exceptional LHS-classes in the ontology. These cases are more intensive because we have to recursively apply the ranking procedure (see Lines 2-8 of Procedure [ComputeRanking](#) in Section 2.2), until all the totally exceptional information is added to the TBox. In our dataset of 134 ontologies, roughly 30% of them have totally exceptional LHS-classes.

Figure 3b shows that, even restricted to cases with totally exceptional LHS-classes, the ranking performance is well inside 100 seconds for the vast majority of the cases. The reason, we conjecture, is that the numbers of defeasible axioms in these ontologies stay relatively low allowing the performance to stay in check. Figure 4a illustrates the number of recursions required in Procedure [ComputeRanking](#) for these cases.

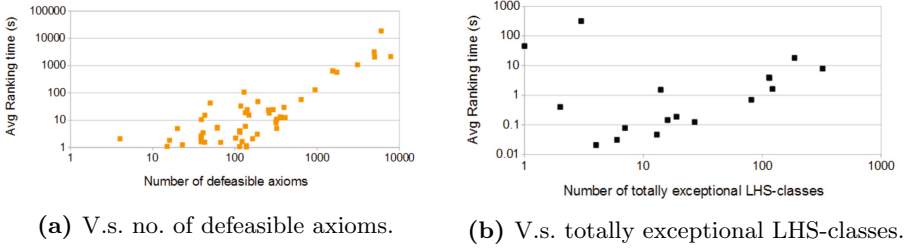


Fig. 3. Ranking computation performance (ranking time).

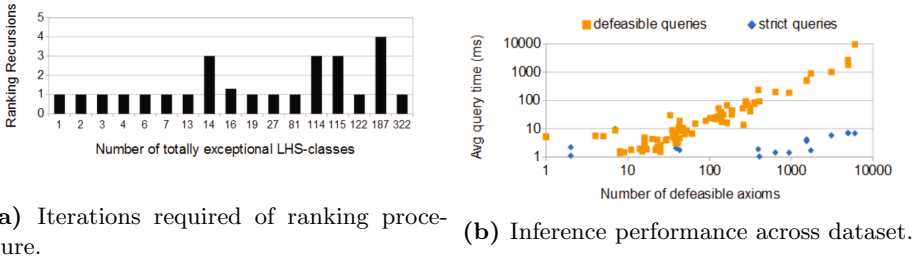


Fig. 4. Recursion in ranking procedure (4a) and defeasible inference performance (4b).

While there are some cases with 3 and 4 repetitions, the majority of cases require just one repetition. This, together with the fact the average number of defeasible axioms for these ontologies is just 127, explains the very low impact on performance that these “hard” cases have. In fact, the average number of defeasible axioms for these cases is significantly lower than that of any of the 3 general corpora in the dataset (see the table in Figure 2). We also notice that a key factor in performance is the number of LHS-classes that are unsatisfiable. Because of the optimisation represented by Lemma 1, we only need check exceptionality for these classes. Therefore the ratio of these incoherent classes to the ontology size will have a major impact on performance. Finally, we note that our hypothesis turns out to be correct concerning the number of ranks in the computed rankings. The average is 2 (single level exceptions), while there are a sprinkling of cases in which there are 3 ranks.

Entailment Performance: The performance of defeasible entailment in the data is also encouraging. It seems that once the ranking of an ontology is obtained, the majority of defeasible entailment queries can be answered instantly. The average time to decide a defeasible entailment was 145ms.

The median is just 4ms highlighting that most of the entailments can be computed almost instantly. As in the case of the ranking performance behaviour, there are a few “outlier” cases which prove much harder than in general. In one particular BioPortal ontology (the most difficult ontology in the data), it takes on average 9.6 seconds to compute a defeasible entailment. There are, however,

421,268 logical axioms in this ontology, of which 6010 are defeasible and 4716 have LHS-concepts that are classically unsatisfiable.

94% of the ontologies in our dataset take less than 200ms on average to decide defeasible entailment. It is not surprising that the prevalence of totally exceptional concepts does not significantly impact the performance of defeasible inference. This is likely because such information was moved to the TBox in the ranking step and therefore of little importance performance-wise during inference.

The dominant factor in performance (for both ranking compilation and query performance) remains ontology size and number of defeasible axioms (see Figure 4b). Because of the low variance in the number of ranks in ontology rankings (between 0 and 3), it is not surprising that this does not significantly impact performance, although we omit the graph illustrating this due to space constraints. It must be stated, though, that in theory the number of ranks will affect performance especially in the case where the exceptionality of the antecedent of the query is high. I.e., in such cases the number of repetitions of the while loop in Procedure [RationalClosure](#) will be higher.

4 Related Work

From a practical standpoint, the most closely related work is that of Bonatti et al. [2]. They introduce a new DL called \mathcal{DL}^N for handling exceptions by allowing or blocking inheritance of certain properties to these exceptions. While the extra features of \mathcal{DL}^N can be built on top of any classical DL, the authors apply their evaluation to ontologies of \mathcal{EL} -variants [1]. They also use an underlying classical OWL reasoner *ELK* (cs.ox.ac.uk/iscg/tools/ELK) which is highly optimised for such logics. In addition, they exploit incremental reasoning capabilities of *ELK* so unnecessary repetition of computations is not required with small changes to the ontology.

The authors use two approaches for extending the Gene Ontology (GO) (geneontology.org), with defeasible features. The first, is a principled injection of purely synthetic defeasible inclusions into GO, and the second is the translation of a *random* subset of classical inclusions in GO to defeasible ones. A direct comparison of their results with ours is non-sensical. (i) Their data is derived through synthetic modifications of GO which are expressed in variants of \mathcal{EL} (rather than our \mathcal{ALC} variants of different ontologies), (ii) they generate cases with between 5 and 25 percent of subsumptions being defeasible (whereas we utilise our DTF with no direct control of this percentage), (iii) their experiment machine is allocated 18GB of memory (whereas ours is restricted to 8GB) and (iv) they exploit the incremental reasoning of *ELK* which greatly increases the performance of reasoning (whereas we do not make use of such facilities). Nevertheless, for interests sake, their KB precompilation times (query time is negligible after precompilation) roughly vary between 25 and 115 seconds under these conditions.

We have, ourselves, also employed synthetic generation of data [7] in the past. All of the data in this evaluation were \mathcal{ALC} ontologies and had between

150 and 5150 axioms. We varied the percentage of defeasible to strict axioms in the ontologies, in increments of 10, between 10 and 100 percent. A direct comparison with our results is also not suitable here, although our ranking times ranged between 0.5 seconds in the 10 percent case, to roughly 8 seconds in the 100 percent case. Query times, thereafter, ranged between 3 and 18 milliseconds for these respective cases.

We also know of some mature Circumscriptive [20] approaches that have been implemented [3,4]. However, the performance results of such implementations remain unpublished to the best of our knowledge.

5 Conclusions and Future Work

We have presented a systematic and intuitive approach to introduce defeasible subsumption into real-world OWL ontologies. Applying this to the Manchester OWL Repository, we were able to generate test cases to evaluate the performance of Rational Closure implemented in DIP (Defeasible-Inference Platform). We report that this kind of defeasible reasoning is quite feasible on our principally generated data. While there are some mentioned limitations of our approach, we argue that the data we generate can give meaningful insight into the performance of RC for real-world ontologies. In conclusion, we hope that our approach provides a stepping stone to developing more sophisticated methodologies for introducing defeasible features into real-world ontologies, and that it will spur more investigations into the performance of defeasible reasoning in general.

Acknowledgments. Part of the work of Giovanni Casini has been supported by the Fonds National de la Recherche, Luxembourg, and cofunded by the Marie Curie Actions of the European Commission (FP7-COFUND) (AFR/9181001).

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., (eds.): *The Description Logic Handbook*. Cambridge Univ. Press (2003)
2. Bonatti, P., Faella, M., Petrova, I., Sauro, L.: *A New Semantics for Overriding in Description Logics*. *Artificial Intelligence* (2015)
3. Bonatti, P., Faella, M., Sauro, L.: *Defeasible Inclusions in Low-Complexity DLs*. *JAIR* **42**, 719–764 (2011)
4. Bonatti, P., Lutz, C., Wolter, F.: *Description logics with circumscription*. In: *Proc. of KR*, pp. 400–410 (2006)
5. Britz, K., Casini, G., Meyer, T., Moodley, K., Varzinczak, I.J.: *Ordered Interpretations and Entailment for Defeasible Description Logics*. Technical report, CAIR, CSIR Meraka and UKZN, South Africa (2013)
6. Britz, K., Meyer, T., Varzinczak, I.: *Semantic foundation for preferential description logics*. In: Wang, D., Reynolds, M. (eds.) *AI 2011. LNCS*, vol. 7106, pp. 491–500. Springer, Heidelberg (2011)
7. Casini, G., Meyer, T., Moodley, K., Varzinczak, I.: *Towards practical defeasible reasoning for description logics*. In: *Proc. of DL* (2013)

8. Casini, G., Straccia, U.: Rational closure for defeasible description logics. In: Janhunen, T., Niemelä, I. (eds.) JELIA 2010. LNCS, vol. 6341, pp. 77–90. Springer, Heidelberg (2010)
9. Cuenca-Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The Next Step for OWL. *Web Semantics: SSAWWW* **6**(4), 309–322 (2008)
10. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Preferential description logics. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 257–272. Springer, Heidelberg (2007)
11. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. *Web Semantics: SSAWWW* **3**(2), 158–182 (2005)
12. Horridge, M.: Justification Based Explanation in Ontologies. PhD thesis, University of Manchester (2011)
13. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL Ontologies. *Semantic Web* **2**(1), 11–21 (2011)
14. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: Ganzinger, H., McAllester, D., Voronkov, A. (eds.) LPAR 1999. LNCS, vol. 1705, pp. 161–180. Springer, Heidelberg (1999)
15. Ke, P., Sattler, U.: Next steps for description logics of minimal knowledge and negation as failure. In: *Proc. of DL* (2008)
16. Lehmann, D., Magidor, M.: What Does a Conditional Knowledge Base Entail? *Art. Intell.* **55**(1), 1–60 (1992)
17. Lukasiewicz, T.: Expressive Probabilistic Description Logics. *Art. Intell.* **172**(6), 852–883 (2008)
18. Matentzoglou, N., Bail, S., Parsia, B.: A snapshot of the OWL web. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 331–346. Springer, Heidelberg (2013)
19. Matentzoglou, N., Tang, D., Parsia, B., Sattler, U.: The manchester OWL repository: system description. In: *Proc. of ISWC*, pp. 285–288 (2014)
20. McCarthy, J.: Circumscription - A Form of Non-Monotonic Reasoning. *Art. Intell.* **13**(1–2), 27–39 (1980)
21. Meyer, T., Moodley, K., Sattler, U.: DIP: a defeasible-inference platform for OWL. In: *Proc. of DL* (2014)
22. Reiter, R.: A Logic for Default Reasoning. *Art. Intell.* **13**(1), 81–132 (1980)
23. Sattler, U., Schneider, T., Zakharyashev, M.: Which kind of module should i extract? In: *Proc. of DL* (2009)
24. Sazonau, V., Sattler, U., Brown, G.: Predicting performance of OWL reasoners: locally or globally? In: *Proc. of KR* (2014)

Empirical Studies

Timely Semantics: A Study of a Stream-Based Ranking System for Entity Relationships

Lorenz Fischer¹ (✉), Roi Blanco², Peter Mika², and Abraham Bernstein¹

¹ Department of Informatics, University of Zurich, Zurich, Switzerland
{lfischer,bernstein}@ifi.uzh.ch

² Yahoo Labs, London, UK
{roi,pmika}@yahoo-inc.com

Abstract. In recent years, search engines have started presenting semantically relevant entity information together with document search results. Entity ranking systems are used to compute recommendations for related entities that a user might also be interested to explore. Typically, this is done by ranking relationships between entities in a semantic knowledge graph using signals found in a data source as well as type annotations on the nodes and links of the graph. However, the process of producing these rankings can take a substantial amount of time. As a result, entity ranking systems typically lag behind real-world events and present relevant entities with outdated relationships to the search term or even outdated entities that should be replaced with more recent relations or entities.

This paper presents a study using a real-world stream-processing based implementation of an entity ranking system, to understand the effect of data timeliness on entity rankings. We describe the system and the data it processes in detail. Using a longitudinal case-study, we demonstrate (i) that low-latency, large-scale entity relationship ranking is feasible using moderate resources and (ii) that stream-based entity ranking improves the freshness of related entities while maintaining relevance.

1 Introduction

In the past years, one of the major developments in the evolution of search engines has been the move from serving only document results to providing entity-based experiences. In contrast to the document results that are crawled from the Web, these experiences are typically built on top of a knowledge base assembled by the search engine provider from various sources of general and domain knowledge. All three major US search engines (Bing, Google, and Yahoo) have developed features that make use of such a knowledge base, and in particular to provide large information boxes which, at the time of writing, appear on the rightmost column of the interface for all three search engines. In all three cases, the displays also provide recommendations for related entities that the user may also want to explore.

Knowledge bases are typically organized in the form of an entity-relationship graph with additional facts attached to the entities and relationships. While the

facts represented in the graph rarely change, the timeliness of relationships can be significantly impacted by real world events. For example, in the domain of entertainment, a movie release could drive significant interest towards the collaborations of actors, or news of an impending celebrity divorce may raise interest into a couple. Similarly, in the domain of sports, a game could drive searches toward the players that participated in certain actions during the game, etc. The features that are used for measuring the importance of these relationships thus also need to be reassessed as a result of these events.

Entity recommender systems typically work by exploiting query logs for predicting the relevance of a related entity, as query logs provide an accurate reflection of current interests. Traditionally, such logs are collected and processed using offline, distributed batch processing systems such as Hadoop MapReduce.¹ These systems are designed to handle large volumes of data but at the cost of significant processing latency. More recently, a new class of distributed systems based on stream processing have become available, opening up the potential for new or improved applications of semantic technologies.

In this work, we describe *Sundog*, a stream processing based implementation of an entity-recommender system and show that by exploiting the temporal nature of search log data, we are able to significantly improve the quality of recommendations compared to static models of relevance, in particular with respect to freshness. The architecture of Sundog is based on a system that has previously been presented at ISWC – Spark [2]. To understand the differences in technology, we provide a comparison to the architecture of the batch-processing based predecessor. We then describe a longitudinal study that evaluates the relevance and freshness of the results computed by the system over a number of consecutive days, using different window sizes and temporal lag in computing the model. We show the benefits of using increasing amounts of data and reducing the lag in processing, namely a relevance and *freshness* increase of over 24% with respect to approaches that use stale data, in the best case. We conclude by discussing improvements and other potential applications of our work.

2 Related Work

Our *Sundog* system is an entity ranking system facilitating semantic search through the application of supervised machine learning techniques to features extracted from query log data. Hence, this section succinctly reviews the related work on (i) semantic search & entity ranking and (ii) temporal information retrieval.

With the introduction of entity-based experiences such as infoboxes, direct answers and *active objects* [16], the disambiguation of query intent and search results have gained in importance, because in these applications mistakes in query interpretation are immediately obvious to the user. However, the semantic gap between the words in user query and the descriptions of entities in the entity-graph can be significant [18]. Entity ranking, or ad-hoc object retrieval is aimed

¹ <http://hadoop.apache.org>

at finding the most relevant entity related to the user’s query, and it has been the focus of many recent studies [2, 14, 19, 20, 26]. Pound et al. provide a query classification of entity-related search queries and define evaluation metrics for the entity retrieval task [20]. This task has also been the focus of evaluations in TREC [1] and other venues such as the SemSearch challenges [3]. A variant of the ad-hoc object retrieval task is the recommendation of related entities, where the focus is on ranking the relationships between a query entity and other entities in the graph, see Kang et al. [14] and van Zwol et al. [26]. More recently, Blanco et al. [2] present their work on the *Spark* system, which is a continuation of the work of Kang et al.

Temporal aspects have gained traction in information retrieval (IR) over the last couple of years and have found applications in document ranking [6, 7, 9], query completion [22], query understanding [8, 15, 17], and recommender systems [5, 21, 24]. Shokouhi et al. [22] analyse temporal trends and also use forecasted frequencies to suggest candidates for auto completion in web search. Kulkarni et al. analyse different features to describe changes in query popularity over time, to understand the intent of queries [15]. In [7], Dai et al. use temporal characteristics of queries to improve ranking web results using machine learned models. They use temporal criteria for their page authority estimation algorithms in [6]. More specifically, they propose a temporal link-based ranking scheme, which also incorporates features from historical author activities. Dong et al. identify *breaking news queries* by training a learning to rank model with temporal features extracted from a page index such as the time stamp of when the page was created, last updated, or linked to [8]. Elsas et al. analyzed the temporal dynamics of content changes in order to rank documents for navigational queries [10]. More related to the topic of query intent analysis, Metzler et al. [17] propose to analyse query logs in order to find base queries that are normally qualified by a year, in order to improve search results for *implicit year qualified* queries. The work that is probably most closely related to our study is by Dong et al. [9]. The authors use realtime data of the micro-blogging website Twitter to extract recency information and train learning to rank models, which in turn are used to rank documents in web search. The recency information from Twitter was then successfully used to rank documents, which promotes documents that are both more fresh and more relevant.

3 System Description

The entity ranking system employed at Yahoo – Spark – is implemented as a batch-processing based pipeline. For this study, we present Sundog, which implements parts of the Spark pipeline using a distributed stream processing framework. A full system description of the production system is beyond the scope of this paper and we refer to [2, 27] for details. However, for the sake of reproducibility and to understand the various design decisions made when building the system for our experiments, it is necessary to have an understanding of Spark. For this reason, we are first going to introduce the most important parts

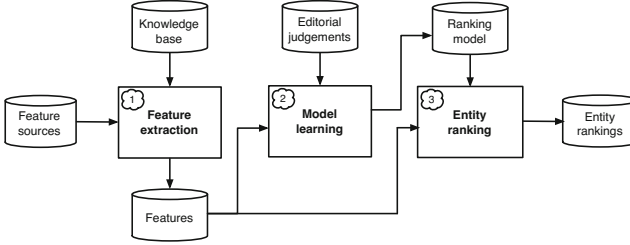


Fig. 1. High-level architecture of the Spark entity ranking system.

of the Spark processing pipeline before we elaborate in detail, how and in what aspects Sundog is different from the original system. We then describe the various performance optimizations we applied. We end the section with performance statistics of the system.

3.1 The Spark Processing Pipeline

Figure 1 gives a high level overview over the Spark ranking pipeline. The ranking essentially happens in three steps: Volatile data sources are used to generate co-occurrence features of entity pairs that are part of the relationships found in a semantic knowledge base (1). Data sources used for this step are Yahoo search logs, tweets from Twitter, and Flickr image tags. Note that for Sundog, we limited ourselves to only use search logs as input data. Next to features extracted from these volatile sources, semantic information such as the entity types and relationship types are leveraged as features. The next step involves the training of a decision tree model using editorial judgements that have previously been collected for a limited set of entity pairs (2). The resulting ranking model is then used to generate entity rankings for all the entity pairs for which features were extracted (3). Disambiguation is conducted in a post-processing step and it is based on a popularity measure derived from Wikipedia. For more information on pre- and post-processing as well as the serving facility, we refer to [2].

Model Learning and Ranking. Spark employs learning to rank approaches in order to derive an efficient ranking function for entities related to a query entity.

Formally speaking, the goal of the Spark ranking system is to learn a function $h(\cdot)$ that generates a score for an input query q_i and an entity e_j that belongs to the set of entities related to the query $e_j \in \mathcal{E}^{q_i}$. Together, q_i and e_j are represented as a *feature vector* \mathbf{w}_{ij} that contains one entry per feature extracted. The input of the learning process consists of *training data* of the form $\{T(q_i) = \{\mathbf{w}_{ij}, l_{ij}\}\}_{q_i \in \mathcal{Q}}$, where $l_{ij} \in L$ is a manually assigned label from a pre-defined set. Spark uses a 5-level label scale ($l \in \{Bad, Fair, Good, Perfect, Excellent\}$) and the assignment from examples (q_i, e_j) was done manually by professional editors, according to a pre-defined set of judging guidelines. The query set \mathcal{Q} is comprised of editorially picked entities and random samples from query logs. This is expected to mimic the actual entity and query distribution of the live

system. The training set might also contain *preference* data, that is, labels that indicate that an entity is preferred over another one for a particular query. The ranking function has to satisfy the set of preferences as much as possible and at the same time is has to *match* the label in the sense that a particular loss function is minimized, for instance square loss $\frac{1}{|Q|} \sum_{q_i \in Q} \frac{1}{|E^{q_i}|} \sum_{e_j \in E^{q_i}} (l_{ij} - h(\mathbf{w}_{ij}))^2$, for a set of test examples.

Similarly to [27], Spark uses Stochastic Gradient Boosted Decision Trees (GBDT) for ranking entities to queries [12, 13]. GBRank is a variant of GBDT that is able to incorporate both label information and pairwise preference information into the learning process [25] and is the function of choice we adopted for ranking in Spark. The system was trained using $\sim 30\text{K}$ editorially labelled pairs and ten fold cross-validation. Each time a model is learned the system sweeps over a number of parameters (learning rate, number of trees and nodes, etc.) and decides on their final value by optimizing for *normalized discounted cumulative gain* (NDCG).

The features included in the system comprise a mixture of *co-occurrence features* and *graph-based features*. Co-occurrence features compute several statistics on mentions of pairs of entities appearing together in the data sources (conditional and joint probabilities, Kullback-Leibler divergence, mutual entropy, etc.). Other features include the types of entities and types of their relationships. In contrast to Spark, Sundog does not currently include graph-based features such as PageRank or the number of shared vertices (common neighbors) between two entities. It does, however, create features using various linear combinations of the features mentioned before as well as make use of the semantic features (type annotations). For a detailed description of these features we refer to [2].

3.2 The Sundog System

In this section we present the implementation details of the Sundog system. First, we describe the programming framework used to build the system. Next, we describe Sundog itself, before presenting a series of optimizations that were implemented to achieve the necessary performance.

Storm and Trident. Sundog was implemented using the *Storm*² realtime computation framework. Storm is best described as the Hadoop MapReduce for stream processing. Similar to MapReduce, data is partitioned, distributed amongst, and processed by multiple compute nodes concurrently. A Storm application—a *topology*—is a directed graph consisting of spout and bolt nodes.

Trident is a higher level programming framework that is part of the Storm distribution. Trident offers higher level concepts such as aggregates, joins, merges, state queries, functions, filters, and methods for state handling. As the Sundog system relies heavily on the computation of state in the form of feature statistics that have to be computed continuously, we chose to use *Trident* for the

² <http://storm-project.net>

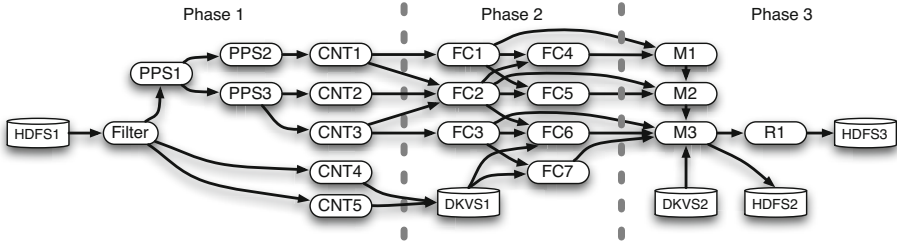


Fig. 2. The *Sundog* topology.

implementation. In Trident, tuples are processed and accounted in mini-batches, offering consistency guarantees on a per-batch basis.

Topology Design. The topology of Sundog can be roughly divided into three phases as depicted in Figure 2. Please note that for the sake of clarity, we show less nodes in the schematic depiction than the actual topology has. Each phase has to fully finish processing, before the next phase can start. For example, before we can compute the probability features of one mini-batch in phase 2, we first have compute the counter values for that mini-batch in phase 1. In the following paragraphs we are going to explain each of these phases in more detail.

In the first phase, query log data is read from the *Hadoop Distributed Filesystem* (HDFS). For the sake of reproducibility of our experiments and because we needed to be able to process historic log data, we chose to read from HDFS, rather than reading from a volatile message queue. We then filter out all search queries that do not contain at least one known entity name. The basis for this filtering is the same knowledge base (KB) that is used in the Spark system. For the experiments in this paper, we relied on the KB that was in production when we ran our experiments. From this reduced data stream, we already count certain events such as the number of distinct users or the number of search events. For other counters, we first have to build entity pairs from the query terms in a series of preprocessing steps (PPS1-3). We count the number of search events and unique users overall (for each entity pair and for each entity). As most of these counters count events and unique users per entity or entity pairs, the relevant data can readily be partitioned and the computation therefore run in parallel on multiple compute nodes. Some counters, however, have a global character and their value has to be aggregated across all data partitions. Their values are stored in an external distributed key-value store (DKVS1) to enable access from all compute tasks in later phases of the processing pipeline. The second phase consists of computing the actual feature values from the counter values (FC1-7). In the final phase (phase 3), the computed features are merged together and complemented with semi-static features that are read from a table in the distributed key-value store (DKVS2 in Figure 2). Semi-static features are features that do not change often, or not at all. For example the semantic type of an entity or a relationship between two entities is assumed to be mostly static. After all features have been merged together, a score is computed for each entity pair

using a GBDT model. Both, the feature values and the final scores are written back to HDFS (HDFS2 and HDFS3 in Figure 2).

Optimizations. A major difference between a Storm topology and a MapReduce-based data pipeline is that while between every MapReduce job data necessarily needs to be written to and read from disk. Storm does not have this requirement. Tuples can pass between bolt instances without ever being written to disk. As long as there is enough memory, state can be kept in memory for fast access. In its current implementation, Sundog only stores two counter values persistently in an external storage. All other information is kept in memory. For this reason, we implemented several optimizations that reduce the memory footprint and the network traffic incurred by the system. In this section, we list the most important ones.

HyperLogLog. For some of the features, we need the number of unique users that searched for a given entity or a pair of entities. These counts are necessary to reduce the impact any single user can have on the ranking of an entity. For example, a fan of a football team may search for the name of his team very often together with the names of several other related entities. Normalizing with the number of unique users (rather than the number of search events) reduces the impact that any single search user has on the ranking.

The naïve way of counting uniques is to create a hash-set for each entity and entity pair – the values we want to count uniques for. For every search event, we could then add the user identifier to the hash-sets of the corresponding entity or entity pair. As hash-sets prevent duplicates from being stored, the size of the hash-set automatically represents the number of unique users that searched for a given entity or entity pair. The disadvantage of this method is, that we need one hash-set for every entity and entity pair and store the user identifiers of each user who searched for the given entity/entity pair. This has a worst case space complexity of $(e + ep) * u$, where e is the number of entities, ep is the number of entity pairs, and u is the number of users. With millions of users and millions of entity pairs, this number can become prohibitively large. To circumvent this, we used an implementation of the *HyperLogLog* algorithm proposed by Flajolet et al.[11]. More specifically, we used the *stream-lib*³ library for approximate counting. The fact that approximate counting - or cardinality estimation - does not provide us with exact counts, should not have a great impact on our results, as we normalize all values with the same “imprecise” counts for unique users. For the experiments presented in this paper, we chose a relative standard deviation of 1% as the target accuracy for the *HyperLogLog* estimator.

Dictionary Encoding and Bloom Filters. As we only have to work with a limited set of entity and relationship types, we use dictionaries to encode both of these values. The compressed dictionary file is so small, that we were able

³ <https://github.com/addthis/stream-lib>

to include it in the jar file deployed on the servers to make it available on all machines.

As described in Section 3.2 the information about semi-static features is stored in a distributed key-value store. The data in this database is also used to filter invalid entity pairs early in the processing pipeline. This ensures that we do not compute feature values for entity pairs that eventually turn out to be invalid. For example the pair “*Brad Pitt - Zurich, Switzerland*” may be found in the search logs, because a user searched for these two entities in the same search query. However, the KB may not contain a relationship entry for the two entities “*Brad Pitt*” and “*Zurich, Switzerland*”. In fact, the vast majority of potential entity pairs are invalid as there are certain geographical locations that have names that (after the text normalization process) are lexicographically equivalent to some words in the English dictionary. For example, there are several villages in Norway with the name “*Å*”. This leads to many candidate entity pairs between “*a*” and other entities that have no semantic relationship with each other. In order to filter these, we need to check against the KB. To reduce the number of requests to the KB, and hence the number of network requests in the process, we use Bloom filters [4]. For the experiments presented in this paper, we added all entity relations in the KB into a Bloom filter. The resulting data structure is included in the application deployment.

3.3 Runtime Characteristics and Performance

There are many factors that influence the performance of a distributed system. For the sake of reproducibility, it may be of interest to the reader to learn more about the setup of our cluster and the configuration parameters that we used for the evaluation of Sundog. For this reason, we present the setup of our system in this section by first describing the general setup of the Storm cluster, before giving some insight into how we configured our topology in order to get better performance out of the hardware that we were able to use.

Cluster Configuration. We ran our evaluations on a cluster of machines that are connected to a 1 Gbit/s network, each having 24 2.2GHz cores and 96GB of RAM. The service that starts and stops Storm worker instances is called the *supervisor* service. There is one supervisor per machine in the cluster. All supervisors are centrally managed by a master server, the *nimbus* node. Communication between the nimbus service and the supervisors happens over a *Zookeeper*⁴ cluster. The *nimbus* server schedules work among the available supervisor nodes. For our experiments, we were given exclusive access to 40 machines. We configured *Storm* to start 8 worker instances (JVMs) per supervisor, each having 12GB of RAM.

Job Configuration and Performance. Table 1 lists setup parameters we used to configure Storm and Trident: We ran our experiments on 40 supervisors,

⁴ <http://zookeeper.apache.org>

Table 1. Sundog configuration parameters and performance numbers of a typical evaluation run.

| Parameter | Value |
|---------------------------------|--------------|
| Workers | 320 |
| Spouts (Spout Tasks) | 1 (40) |
| Bolts (Bolt Tasks) | 30 (2087) |
| Total Task Instances | 2449 |
| Concurrent Batches | 1 |
| Batch Size (log lines) | 100'000 |
| Average Batch Time (Seconds) | \approx 40 |
| Log Lines per Second | 100'000 |
| Transferred Messages per Second | 2.5 mio |

each having 8 workers which resulted in 320 workers in total. These workers were executing 2449 task instances, of which 40 were spout instances and 2087 were regular bolt instances. The remaining bolt instances are acker-instances or Trident coordinator bolts. We ran one spout instance for each machine. Each of these read and emitted 100,000 log lines per batch. One batch took on average 40 seconds to complete, which means that the system ingested about 100,000 log lines per second. We found that neither increasing nor decreasing the batch-size led to increased throughput. Most likely a result of the bookkeeping overhead of Storm becoming proportionally more expensive with smaller batches, while larger batches just increased the processing time per batch. The system transferred about about 2.5 million messages per second within the topology. As running multiple batches concurrently did not yield higher throughput and only increased the chances for batches to time out, we always only processed one batch at a time. This led to the situation that we barely used all of the available resources, because, as mentioned in Section 3.2, certain parts of the computation need to wait for other parts to complete. This suggests that there may be further potential improvements in terms of resource utilization.

While the underlying platforms of Sundog and Spark are vastly different and the performance indicators can therefore not easily be compared directly, it is interesting to note, that even though Spark is running on a cluster that has two orders of magnitude more machines, Sundog is still able to process comparable amounts of data in about $\frac{3}{4}$ of the time used by Spark.

4 Evaluation

Sundog allows us to compute feature values and entity rankings in much less time compared to the old Spark system. This in turns enables us to use more recently collected data for the ranking process. Hence, we are interested in three things: First, we investigate the impact of data recency on the entity rankings. For this we are interested in measuring the quality of the rankings in terms of

freshness and relevance. Secondly, we analyze if fresh rankings are more useful to users. Lastly, we evaluate how the amount and the age of data used to train the system impact performance.

4.1 Experimental Setup

We evaluated the rankings that Sundog produces on four different days over the course of a week. We had human editors assess the rankings with regards to relevance and freshness on each day. In this section, we present the experimental setup of our evaluation. First, we describe the data sets that we collected, before we describe in detail how our editors assessed the generated rankings.

The Data. For our experiments, we produced entity rankings using search log data of three time periods. For each time period we grouped the log files in sets of different sizes (windows). Each log file set $s_i \in S$ has a window of size w_i and an end date d_i . The window size is inclusive. Hence, for a set s_i with a window size $w_i = 7$ and end date $d_i = 2014/01/12$, the respective start date is defined as $d_i - w_i + 1 = 2014/01/06$. We differentiate between three different time periods or epochs, so three collections of *old*, *recent*, and *new* sets of log files. As we ran our experiments on 4 different days, the values for the *new* epoch changed. The end date for the *new* epoch is defined as:

$$d_n \in \{2014/01/20, 2014/01/21, 2014/01/22, 2014/01/23\}$$

For the *recent* and the *old* epoch we chose $d_r = 2014/01/12$ and $d_o = 2013/12/31$, respectively, to simulate the situation in which the rankings would be used during a period of two to three weeks. For each period we compiled a data set of three different sizes $w \in \{1, 7, 30\}$. Table 2 lists the resulting data sets.

Table 2. Data sets collected for the evaluation.

| Epoch | Window | Dates |
|--------|---------|----------------------------------|
| Old | 1 Day | 2013/12/31 |
| | 7 Day | 2013/12/25 – 2013/12/31 |
| | 30 Days | 2013/12/2 – 2013/12/31 |
| Recent | 1 Day | 2014/1/12 |
| | 7 Day | 2014/1/6 – 2014/1/12 |
| | 30 Days | 2013/12/14 – 2014/1/12 |
| New | 1 Day | 2014/1/20...23 |
| | 7 Day | 2014/1/14...17 – 2014/1/20...23 |
| | 30 Days | 2013/12/22...25 – 2014/1/20...23 |

For each of the data sets we first had Sundog generate the feature values which are stored in files. We then used these feature files to train *Gradient*

Boosted Decision Tree (GBDT) models (see 3.1 for details). The resulting models were then used to generate the entity rankings, again stored in files. For each feature file and its corresponding model, we generated one ranking file. In addition, to test the performance of a model that has been generated with an old feature file on freshly generated feature values, we also generated some ranking files using models trained on old data and feature files extracted from new search log data. Note that for all rankings where we used models trained with historic data, we only scored the feature files on models of the corresponding window size, as the feature values in the feature files would otherwise be incompatible with the models.

The resulting ranking files contained a ranking score for each entity pair that at least one user searched for within the corresponding time window. As the number of such rankings can be quite large, we restricted ourselves to evaluate only a subset of all pairs. As we are mostly interested in evaluating for freshness, we selected the top-60 of all queries that matched the label of entities in the KB. This ensured, (i) that we only select queries for which related entities would actually be shown on the search page, and, as the entities in question were “trending”, (ii) increased the likelihood that recency would be a factor for the relationships. We then took all entity pairs that we could find from all 15 ranking files for that day and pooled the query-entity pairs. With at most 10 related entities on the result page, this yielded a pool of at most $60 \times 15 = 900$ entity pairs per day - which was the upper limit of entity pairs that our human editors could evaluate in respect to relevance and freshness in a day. Table 4 lists the exact numbers of query-pairs for each day.

Editorial Judgement. We asked a group of expert search editors working for Yahoo to judge entity pairs in terms of relevance and freshness. Table 3 lists the categories from which the editors could select. The editors were trained and instructed to judge each relationship from the viewpoint of “today”. We asked the editors to research the relationships which they did not know about, in order to provide a well founded judgment.

Table 3. Available recency and relevance categories and their description.

| Recency Categories | | Relevancy Categories | |
|--------------------|-------------------------------|----------------------|---------------------------------------|
| Super Recent | Is current today or yesterday | Super Related | Most interesting factual relationship |
| Very Recent | Was current the past week | Closely Related | Related in a meaningful or useful way |
| Recent | Relevant in the past year | Mostly Related | A little off, but makes sense |
| Reasonable | A bit old, but still popular | Somewhat Related | Not a meaningful or useful suggestion |
| Outdated | There are better connections | Embarrassing | Does not make sense |
| NA or NJ | Freshness is not a factor | N/J | No judgement possible |

Table 4. The number of query pairs evaluated on each day with the corresponding number of pairs that were judged *Super-Recent* or *Super-Related*, respectively.

| Date | Pairs | Super-Recent | Super-Related |
|------------|-------|--------------|---------------|
| 2014/01/20 | 819 | 57 | 290 |
| 2014/01/21 | 696 | 34 | 184 |
| 2014/01/22 | 865 | 54 | 171 |
| 2014/01/23 | 785 | 34 | 196 |

We measure the performance on both *relevance* and *freshness* using standard metrics such as normalized discounted cumulative gain (NDCG), precision, and mean average precision (MAP). Given that we are considering freshness as a discrete, graded variable we report on the same metrics as relevance, but using the editorial labels for recency.

4.2 Results and Discussion

Fresh Data Is Better. In Figure 3 we present our findings on the impact of data freshness on relevance and freshness scores: In the two charts in the upper row we plotted the NDCG scores using the top-10 and the top-5 results. We chose top-10, because Spark always shows the top-10 ranked related entities. Sundog may not be able to always find 10 related entities. This has several reasons: Firstly, Sundog only uses one of the four data sources that Spark uses. Secondly, while Spark uses default values for all features and entity pairs that could not be found in the data, Sundog only computes features values, and hence rankings, for entity pairs that we were able to find in the data. As we are currently mostly interested in freshness of relationships it makes sense not to include relationships that were not of any importance to our users during the time we collected the data. For this reason, Figure 3 also shows the NDCG values for the top-5 ranked entities.

It is apparent, that for sufficiently large time windows, *new* data always produces entity-rankings that are both fresher and also more relevant in general. If the data only contains log data from a single day, we see that while the data that was collected most recently still consistently produces superior rankings, the difference between the rankings of the *recent* data compared to the *old* data is negative. Looking at the numbers in Table 5a we can see a similar picture: Using window sizes of 7 and more days, we observe a significant improvement in terms of freshness when using more recent data.

In the graphs in the lower row of Figure 3, we compare the relevance and freshness measured using several metrics such as precision (P), MAP, and NDCG using a cutoff of 5 and 10, respectively. These charts confirm that our hypothesis also holds for this analysis: Rankings produced from *new* data score higher than rankings produced from historic data.

Fresh is Relevant. Table 6a shows the probabilities of different freshness labels conditioned to observing relevance and non-relevance, respectively. Relevance

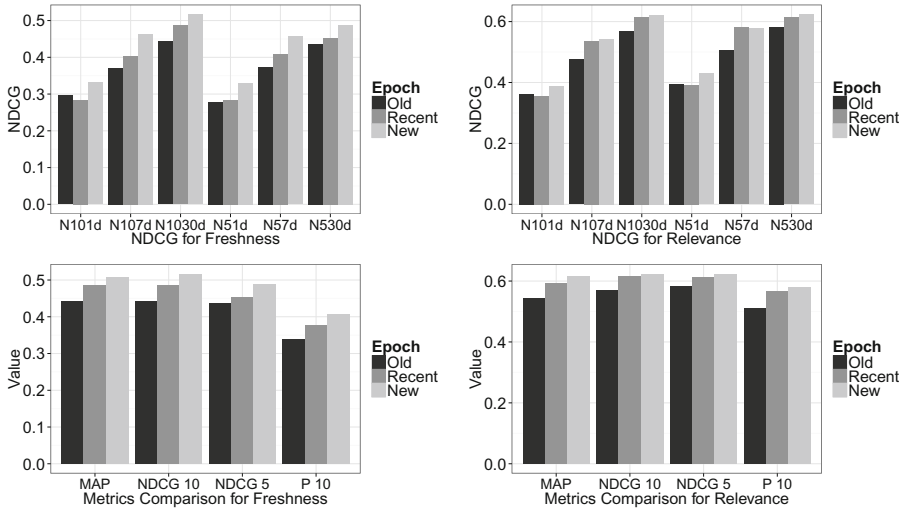


Fig. 3. Top: NDCG for the top 10/5 ranked entities for freshness and relevance Bottom: Comparing several metrics for freshness and relevance for a 30 day window

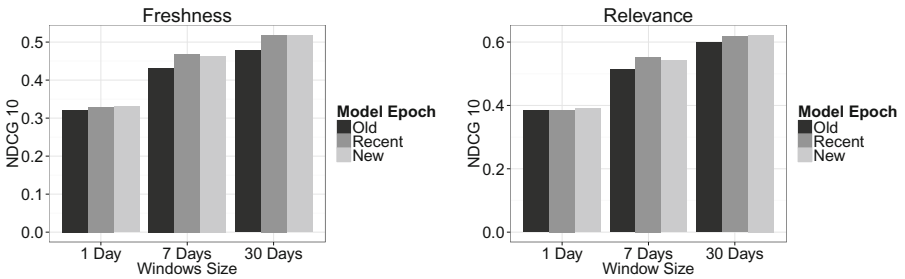


Fig. 4. Comparing the effect of the model age on freshness (left) and relevance (right). NDCG values obtained by applying new feature values on models of varying age.

labels have been collapsed in the table, this is, we deemed the labels *Super Related* and *Closely Related* as relevant and all other labels as not-relevant. The results suggest that freshness is a good indicator for relevance for the *Super Recent* and *Very Recent* categories. On the other hand, looking at Table 6b, we observe that relevance is not a good indicator for freshness. Intuitively this makes sense as it seems logical to assume that there are many more relationships between entities that may, although being relevant, not be of immediate importance in terms of recency. Overall, Pearson's correlation coefficient between labels is 0.28, which indicates that there is only a slight correlation between them.

More Data Is Better. In all but one cases, having more data available to generate the rankings resulted in better performance in terms of relevance

Table 5. NDCG-10 improvements reported over *old* baseline. * indicates a significant improvement over *old*, and † over *recent* (p-value < 0.05, paired two-sided t-test). Values for *new* are averaged over all four days.

| (a) Freshness | | | | |
|---------------|--------|---------|----------|--------------------|
| | Old | Recent | | New |
| 1 | 0.3600 | 0.3446 | (-4.20%) | 0.3868* (+11.13%) |
| 7 | 0.3945 | 0.4317* | (+9.44%) | 0.4870*† (+24.38%) |
| 30 | 0.4569 | 0.4994* | (+9.30%) | 0.5335*† (+16.75%) |

| (b) Relevance | | | | |
|---------------|--------|---------|-----------|-------------------|
| | Old | Recent | | New |
| 1 | 0.4499 | 0.4522 | (+0.66%) | 0.4958† (+10.20%) |
| 7 | 0.5107 | 0.5913* | (+15.80%) | 0.6123* (+19.90%) |
| 30 | 0.6041 | 0.6588* | (+7.71%) | 0.6589* (+9.05%) |

Table 6. Distribution of recency across freshness and relevance values. “Super Related” and “Closely Related” collapsed into “Relevant” (Rel./Not Rel.).

| (a) Relevance | | | (b) Freshness | | |
|---------------|----------------------------|------------------------|---------------|----------------------------|------------------------|
| | $p(\text{Not Rel.} \cdot)$ | $p(\text{Rel.} \cdot)$ | | $p(\cdot \text{Not Rel.})$ | $p(\cdot \text{Rel.})$ |
| Super Recent | 0.15 | 0.85 | Super Recent | 0.02 | 0.11 |
| Very Recent | 0.21 | 0.79 | Very Recent | 0.05 | 0.14 |
| Recent | 0.50 | 0.50 | Recent | 0.23 | 0.17 |
| Reasonable | 0.31 | 0.69 | Reasonable | 0.14 | 0.25 |
| Outdated | 0.57 | 0.43 | Outdated | 0.52 | 0.41 |
| NA or NJ | 0.45 | 0.55 | NA or NJ | 0.05 | 0.03 |

(top-right in Figure 3). Looking at the freshness evaluation, we observe a similar behavior (top-left in Figure 3) with the exception of the NDCG values computed using *new* data for the 7-day window, that for both, the top-10 and the top-5 ranks scored higher than the corresponding NDCG values for the 30-day window computed using “old” data. While this observation is consistent with the machine learning literature, it also shows that data that is more fresh can compensate in situations where only very little historic data can be collected.

Performant Recent Models. In order to assess how well the GDBT models we employed generalize for unknown data, we used models of varying age to rank feature data generated from the most recent log data. The results of this comparison are shown in Figure 4: Using 20 day old data to train the models (Model Epoch = Old) yields worst performance for both freshness and relevance for all time windows, which suggests that the age of a trained model has an impact on performance. Comparing the performance achieved when using 10 day old (Model Epoch = recent) and current (Model Epoch = new) data, however, we

can see that freshly trained models do not necessarily deliver better performance. This suggests, that while fresh data is important for ranking entities, the training of models is less time critical.

5 Conclusions and Future Work

We presented an evaluation of Sundog, a system for ranking relationships between entities on the web using a stream processing framework. Sundog is able to ingest large quantities of data at high rates (orders of magnitude more than a legacy batch-based system) and thus allows for adapting ranking into a live setting, where the ordering of elements (entities) displayed to the user might change with small time delays. This can be accomplished by inspecting relevance signals coming from query logs and updating feature values on the fly.

This architecture enabled us to investigate the tradeoff between data recency and relevance in a live setting, where rankings can change every day. We ran live experiments on four different days using real queries, generating rankings that were evaluated by professional human editors with regards to their *relevance* and *recency*. We trained different models, using old, recent and new data and reported their performance. It is apparent, that for sufficiently large time windows, *new* data always produces entity-rankings that are both, more fresh and also more relevant, with improvements reaching up to 24% in NDCG. We observed, that recency of input data can even compensate for reduced amounts of data, which is traditionally thought of as being a primary factor for the performance of machine learning models. Additionally, the ranking models we deployed are robust enough to be able to generalize well 10 days after they have been trained, even when feature values for query-entity pairs have changed over time. This suggests that while being able to process recent data is crucial, realtime re-learning does not impact performance as much (if at all).

While the source code of the system as well as the search log data used in the study are proprietary to Yahoo or cannot be released to the public for privacy reasons, we do provide a detailed description of the system and the data, which does allow for reproducibility of our results. For example, similar data sets that could be used are tweets from Twitter or image tags from Flickr. In addition, Sundog is built using open source software, e.g. Apache Storm.

In future work, we will explore adaptations to the ranking model in more depth and also investigate, how freshness and relevance can be combined into one objective function. Currently, the models are learned by trying to maximize the relevance score. While recency and relevance are not necessarily two orthogonal performance characteristics, they can differ. The way in which one could combine these two aspects is not clear, yet. This, as well as an investigation of techniques with which recency and relevance can be independently measured without trained editors, remains future work. Additionally, we are interested in equipping the system with an online learner in order to make use of user feedback information (clicks) in real time. Finally, additional work on recency features is also necessary in order for the ranking models to be able to capture time dependent characteristics as for example concept shifts [23].

Acknowledgments. We would like to thank B. Barla Cambazoglu, Alice Swanberg and her team, Derek Dagit, Dheeraj Kapur, Bobby Evans, Balaji Narayanan, and Karri Reddy for their invaluable support.

References

1. Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the trec 2011 entity track. In: Voorhees, E.M., Buckland, L.P. (eds.) TREC. National Institute of Standards and Technology (NIST) (2011)
2. Blanco, R., Cambazoglu, B.B., Mika, P., Torzec, N.: Entity recommendations in web search. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 33–48. Springer, Heidelberg (2013)
3. Blanco, R., Halpin, H., Herzig, D.M., Mika, P., Pound, J., Thompson, H.S., Tran, T.: Repeatable and reliable semantic search evaluation. *J. Web Sem.* **21** (2013)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* **13**(7) (1970)
5. Chen, W., Hsu, W., Lee, M.L.: Modeling user’s receptiveness over time for recommendation. In: SIGIR (2013)
6. Dai, N., Davison, B.D.: Freshness matters: in flowers, food, and web authority. In: SIGIR 2010 (2010)
7. Dai, N., Shokouhi, M., Davison, B.D.: Learning to rank for freshness and relevance (2011)
8. Dong, A., Chang, Y., Zheng, Z., Mishne, G., Bai, J., Zhang, R., Buchner, K., Liao, C., Diaz, F.: Towards recency ranking in web search. In: WSDM (2010)
9. Dong, A., Zhang, R., Kolari, P., Bai, J., Dias, F., Chang, Y., Zheng, Z.: Time is of the essence: improving recency ranking using twitter data. In: WWW (2010)
10. Elsas, J.L., Dumais, S.T.: Leveraging temporal dynamics of document content in relevance ranking. In: WSDM (2010)
11. Flajolet, P., Fusy, E., Gandouet, O., Meunier, F.: HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In: DMTCS Proceedings (2008)
12. Friedman, J.H.: Stochastic gradient boosting. *Computational Statistics and Data Analysis* (1999)
13. Friedman, J.: Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001)
14. Kang, C., Vadrevu, S., Zhang, R., Zwol, R.V., Pueyo, L.G., Torzec, N., He, J., Chang, Y.: Ranking related entities for web search queries (2011)
15. Kulkarni, A., Teevan, J., Svore, K.M., Dumais, S.T.: Understanding temporal query dynamics. In: WSDM. ACM Press (2011)
16. Lin, T., Pantel, P., Gamon, M., Kannan, A., Fuxman, A.: Active objects: actions for entity-centric search. In: WWW (2012)
17. Metzler, D., Jones, R., Peng, F., Zhang, R.: Improving search relevance for implicitly temporal queries. In: SIGIR. No. 1, ACM Press (2009)
18. Mika, P., Meij, E., Zaragoza, H.: Investigating the semantic gap through query log analysis. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 441–455. Springer, Heidelberg (2009)
19. Pehcevski, J., Thom, J.A., Vercoustre, A.M., Naumovski, V.: Entity ranking in Wikipedia: utilising categories, links and topic difficulty prediction. *Information Retrieval* **13**(5) (2010)

20. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: WWW (2010)
21. Ren, Z., Liang, S., Meij, E., de Rijke, M.: Personalized time-aware tweets summarization. In: SIGIR (2013)
22. Shokouhi, M., Radinsky, K.: Time-sensitive query auto-completion. In: SIGIR. ACM Press (2012)
23. Vorburger, P., Bernstein, A.: Entropy-based concept shift detection. In: ICDM. IEEE Computer Society (2006)
24. Yuan, Q., Cong, G., Ma, Z., Sun, A., Thalmann, N.M.: Time-aware point-of-interest recommendation. In: SIGIR (2013)
25. Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., Sun, G.: A general boosting method and its application to learning ranking functions for web search. In: Neural Information Processing Systems (2008)
26. van Zwol, R., Garcia Pueyo, L., Muralidharan, M., Sigurbjörnsson, B.: Machine learned ranking of entity facets. In: SIGIR. ACM Press (2010)
27. van Zwol, R., Pueyo, L.G., Muralidharan, M., Sigurbjörnsson, B.: Ranking entity facets based on user click feedback. In: ICSC. IEEE, September 2010

Link Analysis of Life Science Linked Data

Wei Hu¹(✉), Honglei Qiu¹, and Michel Dumontier²

- ¹ State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, China
whu@nju.edu.cn, hlqiu@smail.nju.edu.cn
- ² Stanford Center for Biomedical Informatics Research,
Stanford University, Stanford, USA
michel.dumontier@stanford.edu

Abstract. Semantic Web technologies offer a promising mechanism for the representation and integration of thousands of biomedical databases. Many of these databases provide cross-references to other data sources, but they are generally incomplete and error-prone. In this paper, we conduct an empirical link analysis of the life science Linked Data, obtained from the Bio2RDF project. Three different link graphs for datasets, entities and terms are characterized using degree distribution, connectivity, and clustering metrics, and their correlation is measured as well. Furthermore, we analyze the symmetry and transitivity of entity links to build a benchmark and preliminarily evaluate several entity matching methods. Our findings indicate that the life science data network can help identify hidden links, can be used to validate links, and may offer the mechanism to integrate a wider set of resources for biomedical knowledge discovery.

Keywords: Link analysis · Bio2RDF · Life sciences · Linked data

1 Introduction

Semantic Web (SW) technologies such as Linked Data provide a salient mechanism by which human and machine can navigate across large and heterogeneous data sources [6]. Links in distributed datasets [14] usually occur between entities (a.k.a. instances) or terms (i.e. classes and properties), and can be not only manually curated but also automatically generated [28]. Due to their complexity and descriptive nature, the life science and health care domains have long been used to assess the feasibility of advanced knowledge management systems. With over 1,500 published biomedical databases, numerous efforts have been directed towards establishing Linked Data for the life sciences, including Bio2RDF [5, 8], Chem2Bio2RDF [9], Neurocommons [25], the EBI RDF Platform [22], and W3C HCLS Linked Open Drug Data.¹ They contain millions of links (e.g. `owl:sameAs` relations) over hundreds of datasets that partially overlap in content. Such rich networks can yield insights into the basic structures demanded to express data

¹ <http://www.w3.org/wiki/HCLSIG/LODD>

types, facilitate large-scale data integration, and help improve the overall quality of biomedical data. To the best of our knowledge, however, there is no such study at present.

In this paper, we conduct an empirical link analysis of the life science Linked Data, obtained from the Bio2RDF project, in three perspectives:

- *Dataset link analysis*, which provides the statistics of datasets and their links to other datasets based on the RDF data model;
- *Entity link analysis*, which captures the status and intended semantics of links between entities using a special kind of cross-references in Bio2RDF;
- *Term link analysis*, which measures the overlap of topics between terms by ontology matching.

For each perspective, we investigate the graph features of Bio2RDF vis-à-vis what has been previously reported, e.g. [12,18]. Specifically, we represent datasets (entities and terms respectively) and their links by a graph, and measure the degree distribution, connectivity and clustering metrics. Furthermore, we examine the symmetry and transitivity of entity links, and establish a benchmark to preliminarily evaluate several entity matching approaches. In addition to study each perspective alone, we also analyze their correlation. The data and results shown in this paper are available at <http://ws.nju.edu.cn/bio2rdf-analysis/>.

Our analytical results and findings are expected to be useful in many areas. For biomedical data exploration [4], our entity link analysis can help create multiple sets of links according to different equivalence criteria and interpretations, e.g. “truly identical” or “close match”. Our dataset link analysis can help identify hidden links between hundreds of biomedical datasets and enable federated SPARQL query processing. Our analysis can also be used to identify error links and poorly annotated datasets, which require more manual or automated curation. Moreover, our empirical analysis of Bio2RDF may reveal some widespread trends in the life sciences and even in the SW, which provide evidences for applications using Linked Data and guide future research.

The rest of this paper is organized as follows. Section 2 provides the preliminaries used in the paper. In Section 3, we introduce the dataset link analysis. In Section 4, we describe the entity link analysis and evaluate entity matching approaches. In Section 5, we present the term link analysis. Section 6 measures the correlation between the three different link structures. We introduce related work in Section 7 and discuss our findings in Section 8. Finally, we conclude this paper with future work in Section 9.

2 Preliminaries

Let \mathbf{U} be the set of URIs, \mathbf{L} be the set of literals, and \mathbf{B} be the set of blank nodes. A triple $\langle s, p, o \rangle \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{L} \cup \mathbf{B})$ is called an *RDF triple*. Following VoID [2], an *RDF dataset* is a set of RDF triples that are published, maintained or aggregated by a single provider. Typically, a dataset is accessible

on the Web, for example through resolvable HTTP URIs or through a SPARQL endpoint, and is identified by a namespace.

In a dataset, named classes, properties and instances are uniquely identified using URIs. Classes and properties together are referred to as *terms*, and terms sharing a common namespace constitute a vocabulary. In this paper, instances are particularly referred to as *entities*.

A *graph* comprises nodes and edges, and edges can be either ordered (a.k.a. arcs) for a directed graph or unordered for an undirected graph. The *degree* of a node is the number of edges incident to it. For a directed graph, we distinguish between the outgoing degree and incoming degree of a node. The outgoing degree of a node is the number of edges directed from it, while the incoming degree of a node is the number of edges directed to it. A *sink* node is a node with outgoing degree equal to 0, while a *source* node has its incoming degree equal to 0. The degree of a node in a directed graph is the sum of its outgoing and incoming degrees. A node with a degree of 0 is called an *isolated* node.

A random variable x is distributed according to a *power law* when its probability density function $p(x)$ is in the form of $p(x) \propto x^{-\alpha}$, where α is a positive constant called *power law exponent*. Power law functions are *scale-free*, in the sense that if x is rescaled by multiplying it by a constant, $p(x)$ would still be proportional to $x^{-\alpha}$. Clauset *et al.* [11] designed a well-known maximum-likelihood method to estimate α for both discrete and continuous values.

A *weakly connected component* (WCC) for a directed graph is a subgraph in which any two nodes can reach each other through some undirected path and to which no more nodes or edges can be added while still preserving its reachability. The number of nodes in a connected component is called its *size*.

The *average distance* for a WCC is the average shortest path length between all nodes in the WCC. The clustering coefficient for a node in a WCC quantifies how close its neighbors are to be a clique (complete graph), while the *clustering coefficient* for the WCC is the average of the clustering coefficients of all nodes. A graph demonstrates the *small world* phenomenon, if its clustering coefficient is significantly higher than that of a random graph on the same node set, and if the graph has a shorter average distance. Degree distribution, average distance and clustering coefficient are considered as the three most robust measures of network analysis.

3 Dataset Link Analysis

Bio2RDF [8] is an open source project that uses SW technologies to build and provide the largest network of life science Linked Data. Particularly, Bio2RDF defines a set of convention scripts to create RDFS compatible Linked Data from a diverse collection of heterogeneously formatted sources obtained from multiple data providers. In this analysis, we use Bio2RDF Release 3 (July 2014), which is the latest version of Bio2RDF and contains about 11 billion RDF triples, 1 billion entities, 2 thousand classes and 4 thousand properties from 35 datasets. For more information, please visit <http://download.bio2rdf.org/release/3/release.html>. To

conduct the dataset link analysis of Bio2RDF, we define the dataset link graph as follows:

Definition 1 (dataset link graph). *A dataset link graph, denoted by (\mathbf{D}, \mathbf{A}) , is a directed graph, where \mathbf{D} is the node set, and each node $D_i \in \mathbf{D}$ denotes a dataset; \mathbf{A} is the arc set, and each arc $(D_i, D_j) \in \mathbf{A}$ exists iff there are at least k RDF triples $(s, p, o) \in D_i$, where s, o are two URIs in D_i and D_j respectively. k is a non-negative integer to adjust the sparseness of arcs in the graph.*

Since Bio2RDF has assigned unique names to the data lacking a source identifier, blank nodes are not existent in the datasets. The original dataset of a URI is obtained by dereferencing the URI, because Bio2RDF includes a few datasets, e.g. BioPortal and iRefIndex, which are themselves aggregates of other datasets [8]. Also, meta-level URIs in RDF(S) and OWL are excluded as every Bio2RDF dataset has RDF triples involving them.

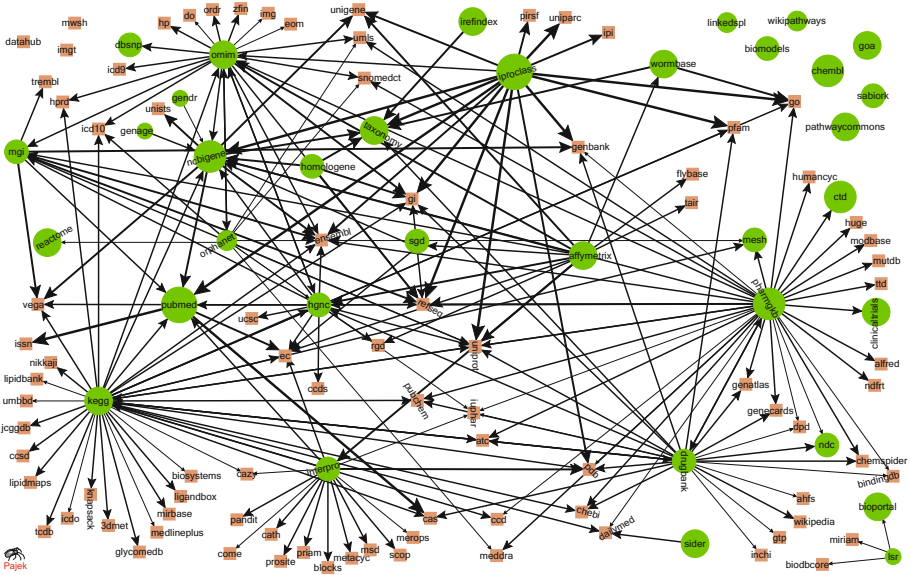
Fig. 1(a) shows the generated dataset link graph for Bio2RDF. We observe that the majority of the datasets is well linked and the largest connected component contains 28 Bio2RDF datasets and 81 *external* datasets that have not been converted in Bio2RDF. The upper right corner depicts seven isolated Bio2RDF datasets that have not linked with others yet, while the upper left corner shows three isolated external datasets linked by less than five triples. In consideration of at least thousands of URIs in each dataset, we regard this little number as a mistake. The lower right corner lists four connected datasets.

Due to most externally linked datasets do not support SPARQL queries, it may be more fair to not consider the directionality of dataset links. The average distance of the largest WCC in the figure is 2.77 and the clustering coefficient is 0.22. The average distance and clustering coefficient for a random graph with the same numbers of nodes and edges are 6.6 and 0.013, respectively. Thus, it indicates very good connectivity among the datasets and reveals the small world phenomenon. Additionally, Fig. 1(a) gives us several hints about the external datasets that are direly needed in the next release of Bio2RDF, such as UniProt and Ensembl, due to many Bio2RDF datasets linking to them (a.k.a. authorities on the Web [7]).

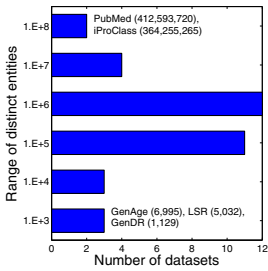
More specifically, Fig. 1(b) illustrates that entities in the Bio2RDF datasets are approximately normally distributed, where 23 datasets have hundred thousands to millions of entities. We also show in this figure the datasets with most or least entities.

Fig. 1(c) illustrates the link distribution between the Bio2RDF datasets only, where OMIM has the most links with the other datasets (including 6 outgoing links and 7 incoming links), followed by NCBI Gene (12 links) and KEGG (11 links). If we took the external datasets into account, the three datasets with most out-going links (a.k.a. hubs [7]) would be KEGG (42 outlinks), PharmGKB (36 outlinks) and DrugBank (24 outlinks).

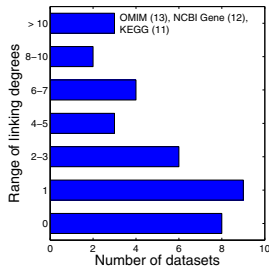
Fig. 1(d) shows the size variation of the largest WCC by keep removing the datasets holding most links. The sequence of removal is KEGG, PharmGKB, OMIM, DrugBank, InterPro and iProClass. We find that the size of the largest WCC decreases slowly, which demonstrates good resilience among the datasets.



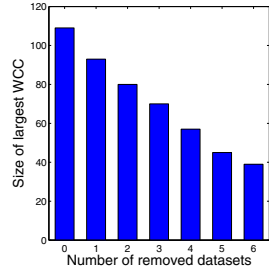
(a) Bio2RDF dataset link graph: (i) the cycles denote the datasets in Bio2RDF Release 3, while the squares represent the externally linked datasets (including BioPortal hosted datasets such as GO). The size of each cycle indicates the number of entities contained in the dataset; and (ii) the arcs constituted by at least five RDF triples are drawn in the figure. The thickness of each arc indicates the number of RDF triples linking one dataset to the other.



(b) Entity distribution



(c) Link distribution betw. Bio2RDF datasets only



(d) WCC size variation by keep removing datasets with most links

Fig. 1. Bio2RDF datasets and their links

In overall, this analysis characterizes a landscape of the current Bio2RDF datasets and provides the basis for analyzing entity and term links in the next two sections.

4 Entity Link Analysis

During the dataset link analysis, we observe that the majority of dataset links is generated from a special kind of RDF triples in the form of $\langle s, x\text{-relation}, o \rangle$.² X-relations contribute to more than 76% entity links, followed by `article` (12%), `gene` (4.3%) and `disease` (1.8%), but they have under-specified semantics.

As an example, `kegg:x-drugbank` links a KEGG entity (e.g. `kegg:D03455`) to a DrugBank entity (e.g. `drugbank:DB00002`) and its intended meaning is to specify that these two entities are “truly identical” (e.g. both refer to the same drug “Cetuximab”), but `kegg:x-drugbank` is not defined as a sub-property of `owl:sameAs`. In another case, `kegg:x-pubmed` signifies a reference to a scientific article that is indexed in the PubMed dataset. Other meanings that we observed include “part of” and “close match”. Actually, due to the design principles of Bio2RDF [18], `owl:sameAs` would be only used when the URI is precisely another name for an entity in the original dataset, for instance, where Bio2RDF URIs for DrugBank entries coincide with URIs assigned by DrugBank itself.

Since x-relations are key to link entities in Bio2RDF, we seek to examine its role in link structure and determine the extent to which we can use x-relations to create entity links. We define the entity link graph using x-relations:

Definition 2 (entity link graph). *An entity link graph, denoted by (\mathbf{E}, \mathbf{X}) , is a directed graph, where \mathbf{E} is the node set, and each node $e_i \in \mathbf{E}$ represents an entity; \mathbf{X} is the arc set, and each arc $(e_i, e_j) \in \mathbf{X}$ exists iff there is an x-relation linking e_i to e_j , in other words, there exists an RDF triple $\langle e_i, x\text{-relation}, e_j \rangle$.*

4.1 Degree Distribution

We generate the entity link graph for Bio2RDF. In Fig. 2, we depict the link distributions (incoming and outgoing) and related statistics for three different types of entities from three datasets: OMIM, NCBI Gene, and KEGG. These three datasets exhibit the most links with the other Bio2RDF datasets (as shown in Fig. 1(c)). The selected types, namely Gene, Phenotype and Drug, have the largest numbers of entities in the corresponding datasets.

We observe that the outgoing/incoming degree distributions of entity links in the three datasets do not exhibit the power law pattern characteristic of scale-free networks (except the outgoing degree distribution for `ncbigene:Gene`). We find that there are fewer entities with an outgoing/incoming degree of 10 than one would expect from a power law distribution. This may be a consequence of overlap among the Bio2RDF datasets such that entities in one dataset are likely to link with at least a certain number of entities in the remaining datasets. Also, the exponents are large (close to 5) and p -values are very small (close to 0).³ In particular, only four datasets link to KEGG and there is no many-to-one links

² These cross-references are created by the original data owners, while Bio2RDF just uniformly converts them to x-relations.

³ The power law hypothesis should be rejected for p -values below 0.01 [11].

between their entities, thus the incoming degree distribution for `kegg:Drug` is sparse. Our results therefore differ from the calculated in-degree distribution of `owl:sameAs` on the 2010 Billion Triples Challenge (BTC) dataset [12]. We argue that this may be the result of link bias from the life science data providers.

In Table 1, we observe that a few entities link to hundreds of other entities, and most of them are widely studied genes and have many related publications or images. Due to the size of NCBI Gene, many entities are not linked by other entities, resulting in a large number (162,018) of source nodes. A direct outcome of our analysis is that we identified one super-connected node (linked to 75,000 nodes), which turned out to be the result of wrong parsing. This bug was fixed immediately by the authors and an updated dataset was released.

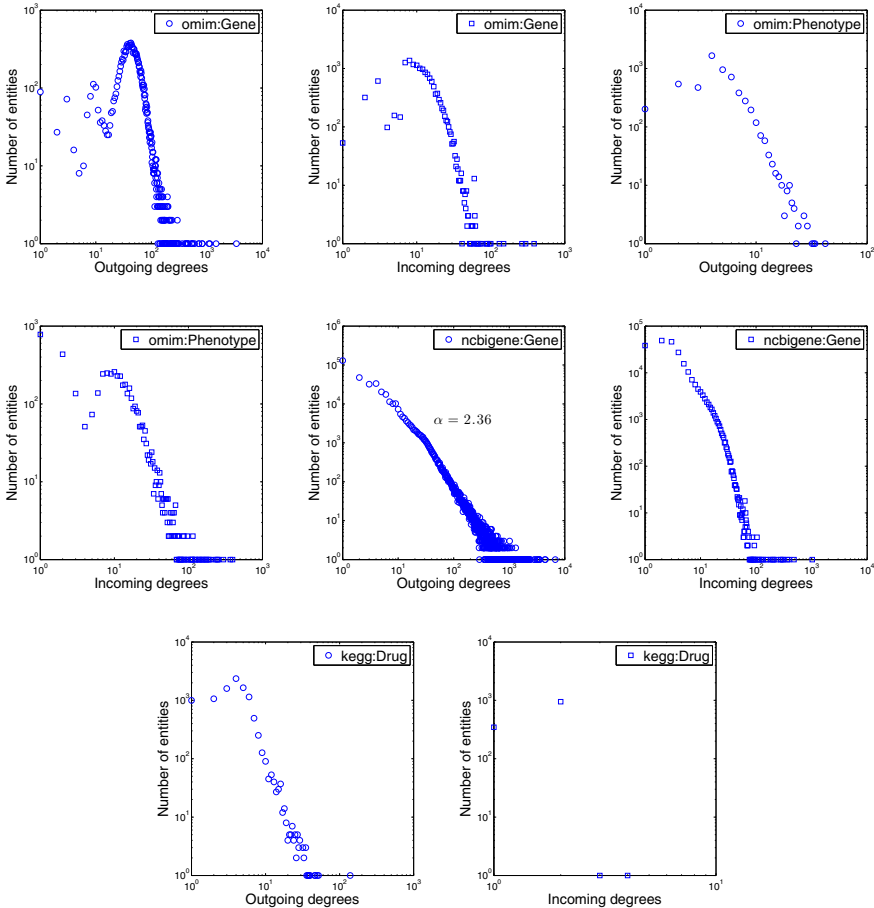


Fig. 2. Bio2RDF entity link distribution: (i) the figures are presented in log-log scale; and (ii) only the datasets in Bio2RDF are considered for computing incoming degrees.

Table 1. Degree analysis of entity links

| Entity types | Entity number | Avg. outdegree | Avg. indegree | Max. degree | Isolated nodes | Sink nodes | Source nodes |
|----------------|---------------|----------------|---------------|-------------|----------------|------------|--------------|
| omim:Gene | 14,609 | 50.3 | 12.8 | 3,409 | 12 | 15 | 118 |
| omim:Phenotype | 5,825 | 5.2 | 10.5 | 414 | 34 | 38 | 1,027 |
| ncbigene:Gene | 394,479 | 10.8 | 2.9 | 6,798 | 0 | 0 | 162,018 |
| kegg:Drug | 10,082 | 4.5 | 0.2 | 139 | 0 | 0 | 8,785 |

4.2 Symmetry and Transitivity of Entity Links

As the entity link graph is directed, we seek to examine the symmetry of entity links. We find that only four pairs of datasets link to each other bi-directionally in Bio2RDF, which are DrugBank—KEGG, DrugBank—PharmGKB, OMIM—HGNC and OMIM—Orphanet.

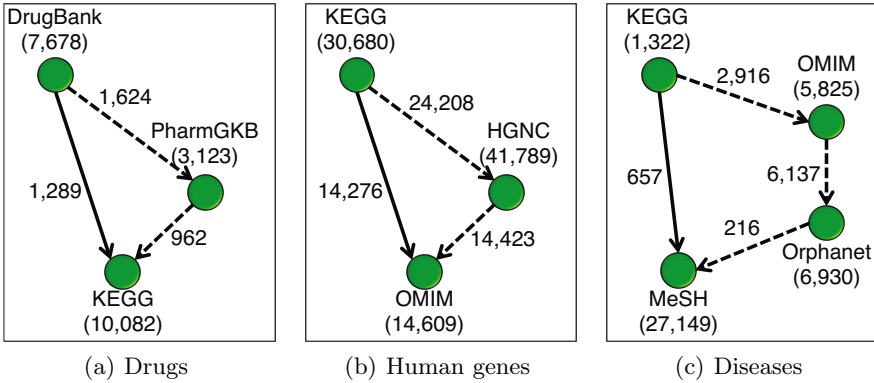
Table 2 lists the results on the symmetry of entity links in the four dataset pairs, where a reciprocal link indicates that two entities e_i, e_j are linked from both directions, a malposed link represents that e_i, e_j are linked in one direction (e.g. $e_i \rightarrow e_j$) but in the other direction e_j links to someone else (e.g. $e_h \leftarrow e_j$), and a missing link implies that either of the two directions is missing.

We observe that the symmetry of entity links varies between different pairs of datasets. For DrugBank—PharmGKB and OMIM—HGNC, a large proportion (99%) of entity links are reciprocal. A possible explanation is that one dataset just borrows the links from the other dataset and simply reverses them. On the other hand, DrugBank—KEGG and OMIM—Orphanet have different numbers of entity links from different directions and are mainly caused by their modeling divergence. For example, OMIM only creates the class `omim:Phenotype` instead of “Disease” and use it to link to `orphanet:Disorder`, which causes many links lost in the other direction, since a disorder may have many different phenotypes.

Also, we analyze the transitivity of entity links, which means that a direct entity link $e_i \rightarrow e_j$ may also be inferred from a transitive path through entity links $e_i \rightarrow e_k \rightarrow \dots \rightarrow e_j$. We find three transitive examples in the Bio2RDF datasets and show them in Fig. 3, where an identical (or different) ending entity indicates that the same entity (different entities) can be achieved through a direct link and a transitive path from the same beginning entity. If the ending entity from the direct link is missing, it is called “missing direct”, while the ending entity from the transitive path is missing, it is called “missing transitive”.

Table 2. Symmetry analysis of entity links

| | Forward | Backward | Reciprocal | Malposed | Missing | Total |
|-------------------|---------|----------|------------|----------|---------|--------|
| DrugBank—KEGG | 1,289 | 2,155 | 1,964 | 485 | 995 | 3,444 |
| DrugBank—PharmGKB | 1,624 | 1,619 | 3,210 | 4 | 29 | 3,243 |
| OMIM—HGNC | 14,274 | 14,423 | 28,514 | 6 | 177 | 28,697 |
| OMIM—Orphanet | 6,137 | 2,600 | 4,464 | 2,523 | 1,750 | 8,737 |



| | Direct links | Transitive paths | Identical ending entities | Different ending entities | Missing direct | Missing transitive | Total |
|-------------|--------------|------------------|---------------------------|---------------------------|----------------|--------------------|--------|
| Drugs | 1,289 | 954 | 946 | 6 | 2 | 343 | 1,297 |
| Human genes | 14,276 | 14,250 | 14,236 | 5 | 9 | 40 | 14,290 |
| Diseases | 657 | 33 | 8 | 18 | 7 | 649 | 682 |

Fig. 3. Transitivity analysis of entity links: (i) the value in each parenthesis denotes the number of entities given a specified topic; and (ii) the solid arcs represent direct links between entities while the dashed arcs form transitive paths. The value on each arc denotes the number of entity links from one dataset to the other.

Our analysis reveals that most links are confirmed through transitivity among the human gene link network only. In the other two examples, there are some intermediate datasets, such as Orphanet, which affect the transitivity. To improve connectivity in the future, these datasets should be enhanced. Also, the number of links may decrease significantly with the increase of transitive path length. Therefore, the transitivity of entity links is often topic-dependent, and its accuracy varies in different contexts.

We take a deeper look at these transitive entity links. Fig. 4 exemplifies two different ending entities from DrugBank to KEGG, where one is from a direct link and the other is from a transitive path. The two drugs have different names but highly similar chemical structures (a.k.a. isomers), and their medical functions are similar as well. The DrugBank provider thinks that the two drugs are the same, while the KEGG provider uses different URIs to identify them without any equivalence relation. This example illustrates the difficulty of linking entities in the life sciences, caused by modeling divergence.

4.3 Evaluation of Entity Matching Approaches

According to our analysis above, we observe that an *x*-relation probably represents the *owl:sameAs* relation between two entities if they have the same or very similar types. Furthermore, although *owl:sameAs* is not a necessarily symmetric

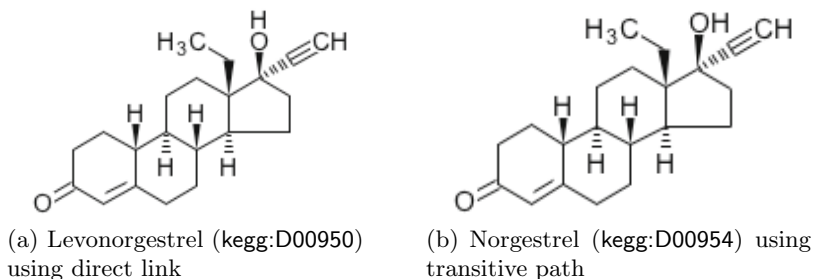


Fig. 4. Different ending entities from starting entity drugbank:DB00367

property [12], it is considered strongly equivalent only when reciprocal links exist. These observations guide us to use the reciprocal links between similarly-typed entities to build a benchmark and evaluate entity matching approaches.

For this purpose, we reuse the four pairs of datasets in Table 2. A commonly-used approach to entity matching in the life sciences is by comparing the labels of entities [17]. We develop four different string comparison algorithms based on Levenshtein, Jaro-Winkler, N-gram ($N = 2$) and Jaccard distances respectively to compute the similarity of labels. For each algorithm, we change the similarity threshold from 0.1 to 0.95 (step by 0.05) to achieve the highest F1-score, where $F1\text{-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$. In overall, the best threshold for each algorithm falls into [0.5, 0.8]. For instance, the best threshold for Jaro-Winkler is achieved at 0.75 when matching DrugBank and KEGG.

Linear regression and logistic regression are often employed to make use of more properties in entities. We re-implement the approach in [28] to identify five matched property pairs by 10-fold cross-validation and combine them using linear or logistic regression for similarity computation. The threshold is set to 0.25, which achieves the best F1-score.

Our experimental results are shown in Fig. 5. We observe that N-gram and Jaro-Winkler algorithms obtain the best F1-score among the string comparison algorithms. But their results are far from perfect, because there are many other useful properties. For example, by considering the property “chemical formula”, the F1-scores achieved by logistic regression consistently rise up on all the drug datasets. For OMIM—Orphanet, the low F1-scores are caused by many-to-one links between the entities in `omim:Phenotype` and `orphanet:Disorder`.

Moreover, four small-scale drug datasets are provided in OAEI2010 and two entity matching systems participated in the test [15]. However, due to the reliability of reference links, the test did not make clear conclusions. We published our benchmark on our website and expect that it can help both researchers and practitioners in biomedicine and the SW verify their entity linking approaches and tools in future.

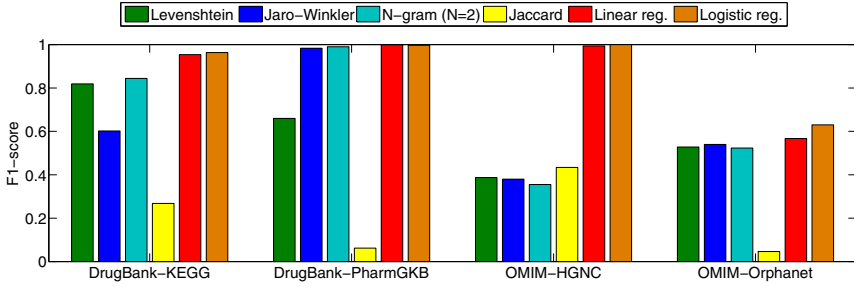


Fig. 5. F1-scores of entity linking approaches

5 Term Link Analysis

Ontology matching aims at creating mappings between terms (classes and properties) from different vocabularies [14], which has already been used for the term link analysis on the SW [17, 20, 24]. In order to investigate the link structure of terms in Bio2RDF, we define the term link graph as follows:

Definition 3 (term link graph). A term link graph, denoted by (\mathbf{T}, \mathbf{M}) , is an undirected graph, where \mathbf{T} is the node set, and each node $t_i \in \mathbf{T}$ denotes a term; \mathbf{M} is the edge set, and each edge $(t_i, t_j) \in \mathbf{M}$ exists iff there is a mapping between t_i and t_j with similarity greater than a specified threshold $\eta \in [0, 1)$.

We construct the term link graph for Bio2RDF using Falcon-AO [21], which is a fully automatic ontology matching tool. The strength of Falcon-AO is that it combines various powerful matchers including two linguistic matchers and a structural matcher. We also enhance Falcon-AO with background knowledge to support synonym identification in the life sciences, e.g. “disease” vs. “disorder”. It is worth noting that there are many approaches and tools can be used as alternatives for this analysis [14]. Among others, Ghazvinian *et al.* [17] used a simple lexical matching of preferred names and synonyms to generate mappings across all classes in 207 biomedical ontologies, while Nikolov and Motta [24] captured the mappings between classes by analyzing existing entity links. However, both of them did not consider the property matching problem.

For the 35 datasets in Bio2RDF, we create 82,689 mappings between classes, 1,540 mappings between object properties and 858 mapping between data properties, with similarity greater than 0.9. We set this threshold based on our empirical experience to achieve a high precision. Due to the simple structure of the Bio2RDF vocabularies, most mappings are found by linguistic matching (similar to [17]). We also note that the mappings between classes are largely in consistent with those discovered in [8] between SIO (Semanticscience Integrated Ontology) and 19 vocabularies in Bio2RDF Release 2. However, SIO only defines very general level properties (e.g. “has attribute”), and matches the properties in other vocabularies using the super/sub-property relation.

Table 3. Top-5 popular labels for classes and properties

| (a) Classes | | (b) Object properties | | (c) Data properties | |
|-------------|---------------|-----------------------|---------------|---------------------|---------------|
| Labels | Distinct URIs | Labels | Distinct URIs | Labels | Distinct URIs |
| Resource | 35 | x-uniprot | 11 | synonym | 25 |
| Gene | 10 | x-ncbigene | 10 | definition | 22 |
| Drug | 6 | article | 8 | comment | 9 |
| Enzyme | 5 | gene | 8 | chromosome | 8 |
| Pathway | 5 | source | 8 | name | 8 |

We extract the label of each term in these mappings and count the times of each label appearing in different terms (by ignoring their string cases). The five most frequently-occurred labels for classes, object and data properties are list in Table 3, where “Resource” is used in all the Bio2RDF datasets to define entities. However, unlike the findings in [17, 20], the degree distribution of term links does not obey the power law, because there is a significant overlap between terms in different vocabularies, indicating that most biomedical data providers have very similar topic interests like genes and drugs. Besides, the created mappings can be used to support query rewriting in applications.

6 Correlation of Different Link Graphs

Earlier in this paper, we have showed our link analysis of datasets, entities and terms respectively. It is also natural for us to ask whether the three types of link graphs are correlated or independent. The *Spearman’s rank correlation coefficient* (denoted by $\rho \in [-1, 1]$) measures the agreement degree between two rankings [23], which is suitable for answering our question. The sign of ρ indicates positive or negative correlation, while its absolute value assesses relative degree, with a larger absolute value being stronger correlation.

We abstract the entity and term link graphs to the dataset level and order the Bio2RDF dataset pairs based on their correlation values. For the entity link graph, the correlation value between two datasets D_i, D_j is defined as the number of direct entity links between D_i, D_j divided by the total number of entities in D_i, D_j . Note that both directions are involved, i.e. $D_i \rightarrow D_j$ and $D_j \rightarrow D_i$.

Inspired by [20], the correlation value of two datasets derived from the term link graph is defined as the ratio of the number of term mappings between the two datasets to the total number of their terms. Note that term mappings are undirected according to our definition.

For the dataset link graph in Fig. 1(a), the correlation value of two Bio2RDF datasets is obtained by finding the shortest path between them, with a shorter length being more strongly correlated. This measure has also been used in [10]. Therefore, we generate three rankings of all pairs of Bio2RDF datasets from the entity, term and dataset link graphs.

Table 4. Spearman’s rank correlation coefficients among link graphs

| | Dataset link graph | Entity link graph |
|-------------------|--------------------|-------------------|
| Entity link graph | 0.51 | |
| Term link graph | 0.42 | 0.16 |

Table 4 lists the correlation coefficients among the entity, term and dataset link graphs. The signs reflect that all the three graphs are positively correlated, where the dataset link graph has strong correlation with the entity link graph ($\rho = 0.51$) as well as the term link graph ($\rho = 0.42$). It can be explained as closer datasets in distance predicting more linked entities along with more matched classes and properties.

On the other hand, the correlation coefficient between the entity link graph and the term link graph is not strong ($\rho = 0.16$), which demonstrates that the number of linked entities contributes little to the overlap of vocabularies, since linked entities may centralize in a few classes, while entities under other classes have not been interlinked yet.

7 Related Work

Network analysis has long been used to study link structures in biomedicine and the Web. The small world phenomenon and the scale-free characteristic are often observed [1, 3, 7, 11]. Recently, it has been conducted on the SW. Theoharis *et al.* [27] investigated the graph features of 250 ontologies and found that a majority of ontologies with a significant number of properties approximate powers for the total degree, while each ontology owns a few focal classes with considerable properties and subclasses. Ell *et al.* [13] introduced a set of label-related metrics including completeness, accessibility, unambiguity and multi-linguality to measure the current state of labeling the Web of Data. These works did not address the relations across different datasets.

To examine entity links, Ding *et al.* [12] carried out an empirical experiment of the `owl:sameAs` deployment status and used the statistics to focus discussion on the usage of `owl:sameAs` in the BTC2010 dataset. Our findings in Bio2RDF do not match their results in some aspects. Halpin *et al.* [18] found that `owl:sameAs` is widely misused to capture different degrees of equivalence, and its practical use is not limited to the case where two entities are truly identical but instead includes application scenarios where they can be treated as being operationally equivalent. Our investigation on the x-relations in Bio2RDF well confirms their observation. For a more general notion of links, Ge *et al.* [16] defined the object link graph according to the RDF data model and compared the graph features of two object link graphs crawled by the Falcons search engine in 2008 and 2009 respectively, containing some incomplete biomedical data.

Analysis of term links has also been performed. Ghazvinian *et al.* [17] analyzed the morphology of term mappings between 207 vocabularies in BioPortal and UMLS, while Hu *et al.* [20] extended this idea to a larger scale containing

four thousand Web ontologies. Nikolov and Motta [24] created term mappings from declared coreference association (e.g. `owl:sameAs`) and co-typing, where a term mapping can hold either the equivalence or subsumption relation. Tordai *et al.* [27] empirically studied the quality of chains of (almost) equivalent terms in the domains of biomedicine, cultural heritage and library subject headings with multiple languages (English, Dutch, German and French). More generally, Cheng *et al.* [10] presented the declarative, topical and distributional relatedness between three thousand vocabularies and the correlation of these relatedness. Unlike these works, we holistically analyzed the life science Linked Data on the levels of datasets, entities and terms.

8 Discussion of Findings

The analytical results that we have presented in the previous sections allow us to make the following observations:

- Bio2RDF offers the biggest network of the life science Linked Data and also is a significant portion of Linked Open Data, which ensures the significance of our empirical study. Although our hypothesis is that the life science data network should be in consistence with that of the SW, we are surprised that some results turn out to be different than previously reported, e.g. the degree distribution of entity links does not strictly follow the power law.
- A dominated part of entities in Bio2RDF have been linked using x-relations, but the intended semantics of these entity links differs. When the meanings of two classes are identical or equivalent and their belonging datasets also have close topics, the entity links are likely to represent logical equivalence. Additionally, the classes and properties in different Bio2RDF datasets have large overlap and can be identified mainly by linguistic matching.
- Symmetric and transitive entity links exist in Bio2RDF, which can reinforce the correctness of these links, but their effectiveness is currently weakened due to the relatively small number. Adding more symmetric and transitive links should be an important future work for the life science data providers and aggregators (e.g. OpenLifeData⁴). Besides, the meanings of entity links may be shifted during transitive. In consideration of the quality and coverage of the entities and terms in Bio2RDF, we suggest to use KEGG, DrugBank and OMIM as the most prominent knowledge bases for applications in the life sciences.
- Previous work has demonstrated the effectiveness of using string matching to find linked entities or terms [15]. However, according to our benchmark, only considering the labels of entities may fail in some cases, e.g. comparing short-form abbreviations of gene names, while combining different properties and using simple machine learning algorithms like logistic regression achieve a good accuracy. However, discovering many-to-one links between entities is still a difficult problem that needs to be carefully studied.

⁴ <http://www.openlifedata.org/>

9 Conclusion

In this paper, we described our analytical results of the life science Linked Data, obtained from the Bio2RDF project, so as to better inform the development of novel methods for exploring, querying and analyzing this wealth of knowledge. Our link analysis coupled with a benchmark give a first glimpse concerning the structure of the life science Linked Data, and offer new results by which we and others may utilize in future. A question raised from our study is how to make use of the findings to improve applications in the life sciences. Another future work is to repeat analysis on other linked biomedical data and compare the findings.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (Nos. 61370019 and 61223003), and in part by the National Social Science Foundation of China (No. 11AZD121). Wei Hu thanks the Stanford Center for Biomedical Informatics Research for hosting his visit.

References

1. Adamic, L.A., Huberman, B.A.: Power-Law Distribution of the World Wide Web. *Science* **287**(5461), 2115 (2000)
2. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets with the VoID Vocabulary. W3C Interest Group Note (2011)
3. Barabási, A.-L., Gulbahce, N., Loscalzo, J.: Network Medicine: A Network-Based Approach to Human Disease. *Nature Reviews Genetics* **12**, 56–68 (2011)
4. Batchelor, C., et al.: Scientific lenses to support multiple views over linked chemistry data. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 98–113. Springer, Heidelberg (2014)
5. Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: Towards a Mashup to Build Bioinformatics Knowledge Systems. *Journal of Biomedical Informatics* **41**(5), 706–716 (2008)
6. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems* **5**(3), 1–22 (2009)
7. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph Structure in the Web. *Computer Networks* **33**(1–6), 309–320 (2000)
8. Callahan, A., Cruz-Toledo, J., Ansell, P., Dumontier, M.: Bio2RDF release 2: improved coverage, interoperability and provenance of life science linked data. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 200–212. Springer, Heidelberg (2013)
9. Chen, B., Dong, X., Jiao, D., Wang, H., Zhu, Q., Ding, Y., Wild, D.J.: Chem2Bio2RDF: A Semantic Framework for Linking and Data Mining Chemogenomic and Systems Chemical Biology Data. *BMC Bioinformatics* **11**, 255 (2010)
10. Cheng, G., Qu, Y.: Relatedness between Vocabularies on the Web of Data: A Taxonomy and an Empirical Study. *Journal of Web Semantics* **20**, 1–17 (2013)

11. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-Law Distributions in Empirical Data. *SIAM Review* **51**(4), 661–703 (2009)
12. Ding, L., Shinauer, J., Shangguan, Z., McGuinness, D.L.: SameAs networks and beyond: analyzing deployment status and implications of owl:sameAs in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 145–160. Springer, Heidelberg (2010)
13. Ell, B., Vrandečić, D., Simperl, E.: Labels in the web of 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. 162–176. Springer, Heidelberg (2011)
14. Euzenat, J., Shvaiko, P.: *Ontology Matching*, 2nd edn. Springer (2013)
15. Ferrara, A., Nikolov, A., Noessner, J., Scharffe, F.: Evaluation of Instance Matching Tools: The Experience of OAEI. *Journal of Web Semantics* **21**, 49–60 (2013)
16. Ge, W., Chen, J., Hu, W., Qu, Y.: Object link structure in the semantic web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) *ESWC 2010, Part II. LNCS*, vol. 6089, pp. 257–271. Springer, Heidelberg (2010)
17. Ghazvinian, A., Noy, N.F., Jonquet, C., Shah, N., Musen, M.A.: What four million mappings can tell you about two hundred ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) *ISWC 2009. LNCS*, vol. 5823, pp. 229–242. Springer, Heidelberg (2009)
18. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs isn't the same: an analysis of identity in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) *ISWC 2010, Part I. LNCS*, vol. 6496, pp. 305–320. Springer, Heidelberg (2010)
19. Hu, W., Chen, J., Zhang, H., Qu, Y.: How matchable are four thousand ontologies on the semantic web. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) *ESWC 2011, Part I. LNCS*, vol. 6643, pp. 290–304. Springer, Heidelberg (2011)
20. Hu, W., Qu, Y.: Falcon-AO: A Practical Ontology Matching System. *Journal of Web Semantics* **6**(3), 237–239 (2008)
21. Jupp, S., Malone, J., Bolleman, J., Brandizi, M., Davies, M., Garcia, L., Gaulton, A., Gehant, S., Laibe, C., Redaschi, N., Wimalaratne, S.M., Martin, M., Le Novère, N., Parkinson, H., Birney, E., Jenkinson, A.M.: The EBI RDF Platform: Linked Open Data for the Life Sciences. *Bioinformatics* **30**(9), 1338–1339 (2014)
22. Myers, J.L., Well, A.D., Lorch Jr., R.F.: *Research Design and Statistical Analysis*, 3rd edn. Routledge (2010)
23. Nikolov, A., Motta, E.: Capturing emerging relations between schema ontologies on the web of data. In: *International Workshop on Consuming Linked Data* (2010)
24. Ruttenberg, A., Rees, J.A., Samwald, M., Marshall, M.S.: Life Sciences on the Semantic Web: The Neurocommons and Beyond. *Briefings in Bioinformatics* **10**(2), 193–204 (2009)
25. Theoharis, Y., Tzitzikas, Y., Kotzinos, D., Christophides, V.: On Graph Features of Semantic Web Schemas. *IEEE Transactions on Knowledge and Data Engineering* **20**(5), 692–702 (2008)

26. Tordai, A., Ghazvinian, A., van Ossenbruggen, J., Musen, M.A., Noy, N.F.: Lost in translation? empirical analysis of mapping compositions for large ontologies. In: International Workshop on Ontology Matching (2010)
27. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Discovering and maintaining links on the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 650–665. Springer, Heidelberg (2009)
28. Xu, M., Wang, Z., Bie, R., Li, J., Zheng, C., Ke, W., Zhou, M.: Discovering missing semantic relations between entities in Wikipedia. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 673–686. Springer, Heidelberg (2013)

Author Index

- Acosta, Maribel I-111
Ali, Muhammad Intizar II-241, II-261,
II-374
Amanatullah, Brian II-205
Arko, Robert II-301
Artiss, Rachel II-205
Auer, Sören I-147, I-513
- Baier, Stephan I-640
Balduini, Marco II-321
Banda, Juan M. II-293
Banerjee, Jay II-3
Beek, Wouter II-339
Bernstein, Abraham II-429
Bhagavatula, Chandra Sekhar I-425
Bischof, Stefan II-57
Blanco, Roi II-429
Bojanapalli, Rajagopal II-205
Bonatti, Piero A. I-356
Brandt, Sebastian II-225
Bretz, Hiroko II-21
Brew, Chris II-21
Brown, Gavin I-533
Bühmann, Lorenz II-76
- Calvanese, Diego I-199
Carbotte, Suzanne II-301
Casini, Giovanni II-409
Chandler, Cynthia II-301
Chang, Shih-Fu II-205
Chavira, Luis Garnica II-310
Cheatham, Michelle II-301
Chen, Tao II-205
Cheng, Gong I-163
Chichester, Christine I-656
Chrysakis, Ioannis I-495
Corby, Olivier II-150
Corsar, David II-329
Costa, Nuno I-569
Cudré-Mauroux, Philippe I-458
- Dale, Chris II-21
Daskalaki, Evangelia I-375
- De Giacomo, Giuseppe I-305
De Vocht, Laurens I-128
del Rio, Nicholas II-310
Di Noia, Tommaso I-605
Di Sciascio, Eugenio I-605
Dietze, Stefan I-474
Difallah, Djellel Eddine I-458
Dimou, Anastasia II-133
Dividino, Renata II-356
Downey, Doug I-425
Dragoni, Mauro II-169
Du, Jianfeng I-339
Dumontier, Michel I-656, II-293, II-446
Duprey, John II-21
- Edwards, Peter II-329
Endris, Kemele M. I-513
Euzenat, Jérôme I-253
- Faisal, Sidra I-513
Fang, Zhijia I-286
Faron-Zucker, Catherine II-150
Ferreira, Lidia II-205
Fetahu, Besnik I-474
Fils, Douglas II-301
Finin, Timothy II-301
Fischer, Lorenz II-429
Flouris, Giorgos I-375, I-495
Flynt, David II-205
Freudenberg, Markus II-133
Fundulaki, Irini I-375
- Gadiraju, Ujwal I-474
Galkin, Michael I-147
Gandon, Fabien II-150
Gangemi, Aldo I-180
Gao, Feng II-374
Giese, Martin I-199
Gotttron, Thomas II-356
Griffin, Keith II-241
- Harrison, Johanna II-21
Hartig, Olaf I-73

- Hassanzadeh, Oktie II-270
 Hayes, Conor I-442
 Hellmann, Sebastian II-133, II-281
 Herschel, Melanie I-375
 Hiebel, Gerald II-205
 Hitzler, Pascal I-237, II-301
 Hogan, Aidan II-261
 Horrocks, Ian II-3, II-93, II-113
 Hovland, Dag I-199, II-93
 Hu, Wei II-446
 Hu, Yingjie II-301
 Hubauer, Thomas II-225
 Hughes, Todd II-205
 Hulpuş, Ioana I-442
- Inants, Armen I-253
- Janowicz, Krzysztof II-301
 Ji, Peng II-301
 Jiang, Jidong I-163
 Jiménez-Ruiz, Ernesto II-93, II-113
 Jones, Matthew II-301
- Kang, Yong-Bin I-322
 Kapoor, Dipsy II-205
 Karima, Nazifa II-301
 Karunamoorthy, Subessware S. II-205
 Kaysar, Mahedi II-241
 Kejriwal, Mayank I-392
 Kharlamov, Evgeny II-93, II-113
 Knight, Kevin II-205
 Knoblock, Craig A. II-205
 Knorr, Matthias I-569
 Knuth, Magnus II-281
 Kontokostas, Dimitris II-133, II-281
 Kostylev, Egor V. I-3
 Kotoulas, Spyros II-186
 Kozák, Jakub II-41
 Krauthammer, Michael I-656
 Krishnaswamy, Shonali I-322
 Krisnadhi, Adila II-301
 Krompaß, Denis I-640
 Kuhn, Tobias I-656, II-293
- Lamparter, Steffen II-225
 Lanti, Davide II-93
 Lee, Michael II-393
 Lehmann, Jens II-133, II-281
 Lehnert, Kerstin II-301
- Leite, João I-569
 Lembo, Domenico I-217
 Lepratti, Raffaello II-225
 Levesque, Hector I-305
 Li, Yuan-Fang I-322
 Lie, Hallstein II-93
 Lopez, Vanessa II-186
 Luggen, Michael I-458
- Mannens, Erik I-92, I-128, II-133
 Marcu, Daniel II-205
 Markovic, Milan II-329
 Martin, Christoph II-57
 Martin, David I-269
 Martin, Robert II-21
 Matentzoglou, Nico II-393
 Mehmood, Qaiser I-52, II-261
 Meyer, Thomas II-409
 Mickle, Audrey II-301
 Mika, Peter II-429
 Mileo, Alessandra II-241, II-374
 Miller, Renée J. II-270
 Miller, Tim II-21
 Minton, Steven II-205
 Miranker, Daniel P. I-392
 Molli, Pascal I-36
 Montoya, Gabriela I-36
 Moodley, Kody II-409
 Mora, Jose I-217, II-113
 Motik, Boris II-3
 Motta, Enrico I-408
 Musen, Mark A. I-551
- Narock, Thomas II-301
 Natarajan, Prem II-205
 Nečaský, Martin II-41
 Nelson, John D. II-329
 Nenov, Yavor II-3
 Ngonga Ngomo, Axel-Cyrille I-52, I-375, II-76, II-261
 Nguyen, Phuong T. I-605
 Noboa, Lisette Espín I-551
 Noraset, Thanapon I-425
 Nöbner, Jan I-269
- O'Brien, Margaret II-301
 Ono, Naomi II-241
 Orlandi, Fabrizio I-147, I-513
 Osborne, Francesco I-408

- Pan, Jeff Z. I-286, I-339
 Parsia, Bijan II-393
 Patel-Schneider, Peter F. I-269
 Paulheim, Heiko I-180
 Pennington, Deana II-310
 Pérez, Jorge I-73
 Petrova, Iliana M. I-356
 Philpot, Andrew II-205
 Pinkel, Christoph II-93, II-113
 Piro, Robert II-3
 Pirrò, Giuseppe I-622
 Pokorný, Jaroslav II-41
 Polleres, Axel II-57
 Prangnawarat, Narumol I-442
 Prokofyev, Roman I-458

 Qiu, Honglei II-446
 Qu, Yuzhong I-163

 Raymond, Lisa II-301
 Reutter, Juan L. I-3, I-19
 Rezk, Martin I-199, II-93
 Rietveld, Laurens II-339
 Ringsquandl, Martin II-225
 Romero, Miguel I-3
 Rosati, Riccardo I-217
 Roussakis, Yannis I-495
 Ruan, Tong I-286

 Saleem, Muhammad I-52, II-261
 Sattler, Uli I-533, II-393, II-409
 Sauro, Luigi I-356
 Saveta, Tzanina I-375
 Savo, Domenico Fabio I-217
 Sazonau, Viachaslau I-533
 Scerri, Simon I-513
 Scherp, Ansgar II-356
 Schilder, Frank II-21
 Schildhauer, Mark II-301
 Schlobach, Stefan II-339
 Schmidt, Renate A. I-587
 Schneider, Patrik II-57
 Sengupta, Kunal I-237
 Shah, Nigam H. II-293
 Shepherd, Adam II-301
 Singer, Philipp I-551
 Singh, Amandeep II-205
 Skaf-Molli, Hala I-36
 Skjæveland, Martin G. II-93, II-113

 Slepicka, Jason II-205
 Smiley, Charese II-21
 Song, Dezhao II-21
 Soto, Adrián I-19
 Stallard, David II-205
 Stavrakas, Yannis I-495
 Stefanidis, Kostas I-495
 Stephenson, Martin II-186
 Strohmaier, Markus I-551
 Szekely, Pedro II-205

 Tamayo, Mike II-205
 Thellmann, Klaudia I-147
 Thorstensen, Evgenij I-217, II-93, II-113
 Tomeo, Paolo I-605
 Tommasi, Pierpaolo II-186
 Tonon, Alberto I-458
 Tresp, Volker I-640
 Tudorache, Tania I-551

 Usbeck, Ricardo II-76

 Vaidya, Gaurav II-281
 Valle, Emanuele Della II-321
 Van de Walle, Rik I-92, I-128, II-133
 Van Herwegen, Joachim I-92, I-128
 Vander Sande, Miel I-92
 Varzinczak, Ivan II-409
 Verborgh, Ruben I-92, I-128, II-133
 Vidal, Maria-Esther I-36, I-111
 Villanueva-Rosales, Natalia II-310
 Vouilloz, Loic I-458
 Vrgoč, Domagoj I-3, I-19

 Walk, Simon I-551
 Wang, Haofen I-286
 Wiebe, Peter II-301
 Wu, Zhe II-3

 Xiao, Guohui II-93

 Yeh, Peter Z. I-269
 Yin, Chengye II-205

 Zhang, Le I-286
 Zhao, Yizheng I-587
 Zheleznyakov, Dmitriy II-93, II-113
 Zheng, Liang I-163
 Zielund, Tom II-21