Annette ten Teije Johanna Völker
Siegfried Handschuh
Heiner Stuckenschmidt Mathieu d'Acquin
Andriy Nikolov Nathalie Aussenac-Gilles
Nathalie Hernandez (Eds.)

LNAI 7603

Knowledge Engineering and Knowledge Management

18th International Conference, EKAW 2012
Galway City, Ireland, October 2012
Proceedings

Springer

Lecture Notes in Artificial Intelligence 7603

Subseries of Lecture Notes in Computer Science

Annette ten Teije Johanna Völker
Siegfried Handschuh
Heiner Stuckenschmidt Mathieu d'Acquin
Andriy Nikolov Nathalie Aussenac-Gilles
Nathalie Hernandez (Eds.)

Knowledge Engineering and Knowledge Management

18th International Conference, EKAW 2012
Galway City, Ireland, October 8-12, 2012
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

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

Johanna Völker
Heiner Stuckenschmidt
University of Mannheim, Germany
E-mail: {johanna, heiner}@informatik.uni-mannheim.de

Siegfried Handschuh
National University of Ireland, Galway, Ireland
E-mail: siegfried.handschuh@deri.org

Mathieu d'Acquin
Andriy Nikolov
The Open University, Milton Keynes, UK
E-mail: {m.dacquin, a.nikolov}@open.ac.uk

Nathalie Aussenac-Gilles
Nathalie Hernandez
Université de Toulouse, France
E-mail: {aussenac, hernandez}@irit.fr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33875-5 e-ISBN 978-3-642-33876-2
DOI 10.1007/978-3-642-33876-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012947926

CR Subject Classification (1998): I.2, H.4, H.3, J.1, C.2, H.2.8, H.5, D.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

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

Printed on acid-free paper

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

Preface

This volume contains the proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2012), held October 8–12, 2012 in Galway City, Ireland, 25 years after the first 'European Knowledge Acquisition Workshop', which took place at Reading University in the UK in 1987. We took this anniversary as an opportunity to ask ourselves what impact 25 years of research in the area had had on practical applications and we announced this year's event under the Motto "Knowledge Engineering and Knowledge Management that Matters". On the one hand, this motto addresses real-world applications of knowledge engineering in areas such as medicine or e-business, on the other hand, it addresses new developments on the Web that make knowledge available to a wide range of users in terms of linked open data, a development that can really have an impact on the ways people use information. In both directions, knowledge engineering faces a variety of theoretical and practical problems. These were the subject of the research discussed at the conference.

Following the motto of the conference, the event featured three invited talks and five tutorials highlighting real-world applications of knowledge engineering such as medicine, e-commerce, e-science, and content management. The invited talks are summarized below:

Martin Hepp: From Ontologies to Web Ontologies: Lessons Learned from Conceptual Modeling for the World Wide Web In this keynote talk, Martin discusses whether there is a fundamental difference between traditional ontologies and Web ontologies, and analyzes the specific economic, social, and technical challenges of building, maintaining, and using socially agreed, global data structures that are suited for the WWW at large, also with respect to the skills, expectations, and particular needs of companies and Web developers.

Michael Uschold: Building Enterprise Ontologies: Report from the Trenches Michael explores the issues and challenges that arise when building real-world enterprise ontologies for large organizations. He addresses questions like: What is the purpose of the ontology? Where do you start? What do you include? Where does the knowledge come from? Do you use an upper ontology? Which one? He looks at how to represent concepts beyond the usual people, places, and time and explores how these things impact the business.

Lee Harland: Practical Semantics in the Pharmaceutical Industry – The Open PHACTS Project Lee outlines the traditional approaches major pharmaceutical companies have taken to knowledge management and describes the business reasons why pre-competitive, cross-industry, and public-private partnerships have gained much traction in recent years. He considers the scientific challenges concerning the integration of biomedical knowledge,

highlighting the complexities in representing everyday scientific objects in computerized form leading him to the third strand, technology, and how the semantic web might lead at least someway to a long-overdue solution.

The research paper track focused more on principles and methods than on specific applications. This clearly shows that after 25 years of research, knowledge engineering is still an active field that continuously develops novel results and solutions for practical problems. The topics covered include social and cognitive aspects of knowledge representation, natural language processing, knowledge extraction and enrichment, ontology engineering and evaluation, and linked data. For the main track, we received 107 submissions, 83 of which were research papers, 10 position papers, and only 14 where explicitly submitted as in-use papers and therefore mainly focusing on a particular application. Out of these submissions, 13 papers where accepted as long and 12 as short research papers. This amounts to an acceptance rate of 15% for long papers and about 30% in total, including the short papers. In the in-use category five papers where accepted, two as long and three as short papers, equaling an acceptance rate of about 15% for long and about 40% including the short papers in this category. In addition three of the ten position papers where accepted for discussion at the conference. These numbers are in line with the acceptance policies at previous conferences continuing the tradition of balancing rigid quality control with the idea of having a community event that leaves space for discussion and exchange of ideas.

Supporting the idea of EKAW as a community event, this year's event again featured a rich side program complementing the main paper track in terms of a PhD symposium, workshops, and a poster and demo session. More specifically, the conference featured six official workshops covering topics such as knowledge extraction, ontology engineering, knowledge exploration, and applications to medical and personal data. Overall, we think that we managed to set up a very good program for EKAW 2012 and hope that all participants enjoyed the conference. We want to thank everybody who contributed to the conference including the local organization team, all members of the organizing and program committees, as well as our sponsors and the invited speakers.

August 2012

Heiner Stuckenschmidt
Siegfried Handschuh

Organization

Executive Committee

Conference Chair

Heiner Stuckenschmidt	University of Mannheim, Germany

Local Chair

Siegfried Handschuh	Digital Enterprise Research Institute, Ireland

Program Chairs

Annette ten Teije	VU University Amsterdam, The Netherlands
Johanna Völker	University of Mannheim, Germany

Demo and Poster Chairs

Andriy Nikolov	KMi The Open University, UK
Mathieu d'Aquin	KMi The Open University, UK

Workshop and Tutorial Chairs

Krzysztof Janowicz	University of California, Santa Barbara, USA
Claudia d'Amato	University of Bari, Italy

Doctoral Consortium Chairs

Nathalie Hernandez	IRIT-CNRS, Toulouse, France
Nathalie Aussenac-Gilles	IRIT-CNRS, Toulouse, France

Sponsorship Chair

Chris Bizer	Free University Berlin, Germany

Local Organization

Keith Cortis	Digital Enterprise Research Institute, Ireland
Maciej Dabrowski	Digital Enterprise Research Institute, Ireland
Maria Smyth	Digital Enterprise Research Institute, Ireland

Program Committee

Stuart Aitken	University of Edinburgh, UK
Harith Alani	The Open University, UK
Lora Aroyo	VU University Amsterdam, The Netherlands
Sören Auer	Universität Leipzig, Germany
Nathalie Aussenac-Gilles	IRIT-CNRS, Toulouse, France
Eva Blomqvist	Linköping University, Sweden
Johan Bos	University of Groningen, The Netherlands
Paolo Bouquet	University of Trento, Italy
Christopher Brewster	Aston University, UK
Liliana Cabral	CSIRO, Australia
Iván Cantador	Universidad Autónoma de Madrid, Spain
Caterina Caracciolo	United Nations Food and Agriculture Organization, Italy
Jean Charlet	INSERM/AP-HP Paris, France
Vinay Chaudhri	SRI International, USA
Paolo Ciancarini	University of Bologna, Italy
Philipp Cimiano	Universität Bielefeld, Germany
Paul Compton	University of New South Wales, Sydney, Australia
Olivier Corby	INRIA Sophia Antipolis, France
Claudia d'Amato	University of Bari, Italy
Mathieu d'Aquin	The Open University, UK
Stefan Decker	DERI, National University of Ireland, Ireland
Klaas Dellschaft	Universität Koblenz, Germany
Jérôme Euzenat	INRIA and LIG, France
Dieter Fensel	STI Innsbruck, University of Innsbruck, Austria
Jesualdo Tomás Fernández-Breis	Universidad de Murcia, Spain
Aldo Gangemi	ISTC-CNR, Italy and Université Paris 13, LIPN, CNRS, France
Serge Garlatti	Télécom Bretagne, France
Dragan Gasevic	Athabasca University, Canada
Chiara Ghidini	Fondazione Bruno Kessler, Italy
Luca Gilardoni	Quinary, Milan, Italy
Paul Groth	VU University Amsterdam, The Netherlands
Michael Grüninger	University of Toronto, Canada
Jon Atle Gulla	Norwegian University of Science and Technology, Norway
Asunción Gómez-Pérez	Universidad Politécnica de Madrid, Spain
Peter Haase	fluid Operations AG, Germany
Harry Halpin	W3C/MIT, USA

Cory Henson	Wright State University, USA
Martin Hepp	Universität der Bundeswehr, Germany
Aidan Hogan	DERI, National University of Ireland, Ireland
Andreas Hotho	Universität Würzburg, Germany
Eero Hyvönen	Aalto University, Finland
Gilles Kassel	Université de Picardie Jules Verne, France
Khaled Khelif	Université Nice Sophia Antipolis, France
Patrick Lambrix	Linköping University, Sweden
Jens Lehmann	Universität Leipzig, Germany
Wolfgang Maass	Universität des Saarlandes, Germany
Diana Maynard	University of Sheffield, UK
Robert Meersman	Vrije Universiteit Brussel, Belgium
Michele Missikoff	IASI-CNR, Rome, Italy
Riichiro Mizoguchi	Osaka University, Japan
Dunja Mladenic	Jozef Stefan Institute, Slovenia
Paola Monachesi	University of Utrecht, The Netherlands
Enrico Motta	The Open University, UK
Mark Musen	Stanford University, USA
Roberto Navigli	Sapienza University of Rome, Italy
Axel-Cyrille Ngonga Ngomo	Universität Leipzig, Germany
Vit Novacek	DERI, National University of Ireland, Ireland
Natasha Noy	Stanford University, USA
Viktoria Pammer	Know-Center, Austria
Adrian Paschke	Freie Universität Berlin, Germany
Carlos Pedrinaci	The Open University, UK
Wim Peters	University of Sheffield, UK
H. Sofia Pinto	IST, Technical University of Lisbon, Portugal
Enric Plaza	IIIA-CSIC (Spanish National Research Council), Spain
Alun Preece	Cardiff University, UK
Valentina Presutti	CNR, Semantic Technology Laboratory, Rome, Italy
Yannick Prié	Université Claude Bernard Lyon 1, France
Alan Rector	University of Manchester, UK
Ulrich Reimer	University of Applied Sciences St. Gallen, Switzerland
Chantal Reynaud	Université Paris-Sud, France
Marco Rospocher	Fondazione Bruno Kessler, Italy
Sebastian Rudolph	Karlsruhe Institute of Technology (KIT), Germany
Marta Sabou	MODUL University Vienna, Austria
Harald Sack	Hasso-Plattner Institute for IT Systems Engineering, Germany
Ansgar Scherp	University of Koblenz-Landau, Germany
Stefan Schlobach	VU University Amsterdam, The Netherlands

Guus Schreiber	VU University Amsterdam, The Netherlands
Luciano Serafini	Fondazione Bruno Kessler, Italy
Amit Sheth	Wright State University, USA
Elena Simperl	Karlsruhe Institute of Technology (KIT), Germany
Derek Sleeman	University of Aberdeen, UK
Pavel Smrz	Brno University of Technology, Czech Republic
Steffen Staab	Universität Koblenz, Germany
Nenad Stojanovic	Forschungszentrum Informatik, Karlsruhe, Germany
Rudi Studer	Karlsruhe Institute of Technology (KIT), Germany
Mari Carmen Suárez-Figueroa	Universidad Politécnica de Madrid, Spain
Ondrej Svab-Zamazal	University of Economics, Prague, Czech Republic
Vojtech Svatek	University of Economics, Prague, Czech Republic
Valentina Tamma	University of Liverpool, UK
Christoph Tempich	inovex GmbH, Pforzheim, Germany
Robert Tolksdorf	Freie Universität Berlin, Germany
Francky Trichet	Université de Nantes, France
Giovanni Tummarello	DERI Galway, Ireland
Iraklis Varlamis	Harokopio University of Athens, Greece
Fabio Vitali	University of Bologna, Italy
Michael Witbrock	Cycorp Europe, Slovenia
Hannes Werthner	Vienna University of Technology, Austria
Fouad Zablith	American University of Beirut, Lebanon

Additional Reviewers

Loris Bozzato	Fondazione Bruno Kessler, Italy
Smitashree Choudhury	The Open University, UK
Ioana Ciuciu	Vrije Universiteit Brussel, Belgium
Miriam Fernandez	The Open university, UK
Mouzhi Ge	Universität der Bundeswehr, Germany
Olaf Görlitz	Universität Koblenz, Germany
Christian Hentschel	Hasso-Plattner Institute for IT Systems Engineering, Germany
Daniel Herzig	Karlsruhe Institute of Technology (KIT), Germany
Naouel Karam	Freie Universitaet Berlin, Germany
Magnus Knuth	Hasso-Plattner Institute for IT Systems Engineering, Germany
Günter Ladwig	Karlsruhe Institute of Technology (KIT), Germany

Nadeschda Nikitina	Karlsruhe Institute of Technology (KIT), Germany
Bene Rodriguez	Universität der Bundeswehr, Germany
Willem Robert Van Hage	VU University Amsterdam, The Netherlands
Brigitte Safar	Université Paris-Sud, France
Fatiha Saïs	Université Paris-Sud, France
Nadine Steinmetz	Hasso-Plattner Institute for IT Systems Engineering, Germany
Yan Tang	Vrije Universiteit Brussel, Belgium
Duc Thanh Tran	Karlsruhe Institute of Technology (KIT), Germany
Kia Teymourian	Freie Universität Berlin, Germany
Alexandru Todor	Freie Universität Berlin, Germany
Sara Tonelli	Fondazione Bruno Kessler, Italy
Harry Yu	The Open University, UK

Demo/Poster Program Committee

Alessandro Adamou	Semantic Technology Lab, ISTC-CNR, Italy
Pierre-Antoine Champin	LIRIS, France
Tudor Groza	School of ITEE, The University of Queensland, Australia
Vanessa Lopez	IBM Research Lab, Dublin, Ireland
Raul Palma	Poznan Supercomputing and Networking Center, Poland
Ignazio Palmisano	University of Manchester, UK
Matthew Rowe	The Open University, UK
Bernhard Schandl	Gnowsis.com, Austria
Hala Skaf-Molli	Nantes University, France
Christopher Thomas	Wright State University, USA
Raphael Troncy	EURECOM, France
Tania Tudorache	Stanford University, USA
Boris Villazon-Terrazas	Universidad Politecnica de Madrid, Spain

Sponsoring Organizations

Silver Sponsors

Elsevier, The Netherlands
fluid Operations, Germany
Interaction-Design.org, Denmark
STI International, Austria

Gold Sponsors

DERI - Digital Enterprise Research Institute, Ireland
di.me - Integrated digital.me Userware, FP7 Research Project
InEs - Institute for Enterprise Systems, Germany
LOD2 - Creating Knowledge out of Interlinked Data, FP7 Research Project
National University of Ireland, Ireland
Open PHACTS - Open Pharmacological Space, IMI Research Project
Semantic Web Company, Austria

Table of Contents

Invited Paper

Open PHACTS: A Semantic Knowledge Infrastructure for Public and Commercial Drug Discovery Research 1
Lee Harland for The Open PHACTS Consortium

PhD Symposium (Best Paper)

Allowing End Users to Query Graph-Based Knowledge Bases 8
Camille Pradel

Position Papers

Nichesourcing: Harnessing the Power of Crowds of Experts 16
Victor de Boer, Michiel Hildebrand, Lora Aroyo, Pieter De Leenheer, Chris Dijkshoorn, Binyam Tesfa, and Guus Schreiber

Dimensions of Argumentation in Social Media 21
Jodi Schneider, Brian Davis, and Adam Wyner

Semantic Knowledge Discovery from Heterogeneous Data Sources 26
Claudia d'Amato, Volha Bryl, and Luciano Serafini

Research Papers

Knowledge Extraction and Enrichment

Automatic Subject Metadata Generation for Scientific Documents Using Wikipedia and Genetic Algorithms 32
Arash Joorabchi and Abdulhussain E. Mahdi

Advocatus Diaboli – Exploratory Enrichment of Ontologies with Negative Constraints 42
Sébastien Ferré and Sebastian Rudolph

Universal OWL Axiom Enrichment for Large Knowledge Bases 57
Lorenz Bühmann and Jens Lehmann

Proxemic Conceptual Network Based on Ontology Enrichment for Representing Documents in IR 72
Chiraz Latiri, Lamia Ben Ghezaiel, and Mohamed Ben Ahmed

Natural Language Processing

Extracting Multilingual Natural-Language Patterns for RDF Predicates 87
Daniel Gerber and Axel-Cyrille Ngonga Ngomo

Improving the Performance of a Named Entity Recognition System with Knowledge Acquisition 97
Myung Hee Kim and Paul Compton

Knowledge Extraction Based on Discourse Representation Theory and Linguistic Frames 114
Valentina Presutti, Francesco Draicchio, and Aldo Gangemi

Investigating the Semantics of Frame Elements 130
Sara Tonelli, Volha Bryl, Claudio Giuliano, and Luciano Serafini

Linked Data

Keys and Pseudo-Keys Detection for Web Datasets Cleansing and Interlinking 144
Manuel Atencia, Jérôme David, and François Scharffe

Ranking RDF with Provenance via Preference Aggregation 154
Renata Dividino, Gerd Gröner, Stefan Scheglmann, and Matthias Thimm

Freshening up While Staying Fast: Towards Hybrid SPARQL Queries ... 164
Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira

Linked-Data Aware URI Schemes for Referencing Text Fragments 175
Sebastian Hellmann, Jens Lehmann, and Sören Auer

An Interactive Guidance Process Supporting Consistent Updates of RDFS Graphs 185
Alice Hermann, Sébastien Ferré, and Mireille Ducassé

Effective Retrieval Model for Entity with Multi-valued Attributes: BM25MF and Beyond 200
Stéphane Campinas, Renaud Delbru, and Giovanni Tummarello

Ontology Engineering and Evaluation

Ontology Testing - Methodology and Tool 216
Eva Blomqvist, Azam Seil Sepour, and Valentina Presutti

A Model of Derived Roles 227
Kouji Kozaki, Yoshinobu Kitamura, and Riichiro Mizoguchi

ONSET: Automated Foundational Ontology Selection and Explanation 237
Zubeida Khan and C. Maria Keet

Detecting and Revising Flaws in OWL Object Property Expressions 252
C. Maria Keet

Validating Ontologies with OOPS! 267
María Poveda-Villalón, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez

Social and Cognitive Aspects of Knowledge Representation

Towards the Semantic Web – Incentivizing Semantic Annotation Creation Process 282
Szymon Łazaruk, Monika Kaczmarek, Jakub Dzikowski, Oksana Tokarchuk, and Witold Abramowicz

Evaluating Wiki-Enhanced Ontology Authoring 292
Chiara Di Francescomarino, Chiara Ghidini, and Marco Rospocher

SlideWiki: Elicitation and Sharing of Corporate Knowledge Using Presentations 302
Ali Khalili, Sören Auer, Darya Tarasowa, and Ivan Ermilov

Applications of Knowledge Engineering

A Knowledge-Based Approach to Augment Applications with Interaction Traces 317
Olivier Curé, Yannick Prié, and Pierre-Antoine Champin

Building a Library of Eligibility Criteria to Support Design of Clinical Trials 327
Krystyna Milian, Anca Bucur, and Frank van Harmelen

Realizing Networks of Proactive Smart Products 337
Mathieu d'Aquin, Enrico Motta, Andriy Nikolov, and Keerthi Thomas

In-Use Papers

LODStats – An Extensible Framework for High-Performance Dataset Analytics 353
Sören Auer, Jan Demter, Michael Martin, and Jens Lehmann

Implementing an Automated Ventilation Guideline Using the Semantic Wiki KnowWE 363
Reinhard Hatko, Dirk Schädler, Stefan Mersmann, Joachim Baumeister, Norbert Weiler, and Frank Puppe

Knowledge Management on the Desktop 373
Laura Drăgan and Stefan Decker

Improving the Quality of SKOS Vocabularies with Skosify 383
Osma Suominen and Eero Hyvönen

The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation 398
Silvio Peroni, David Shotton, and Fabio Vitali

Demonstrations

Key-Concept Extraction for Ontology Engineering 413
Marco Rospocher, Sara Tonelli, Luciano Serafini, and Emanuele Pianta

Latest Developments to LODE 417
Silvio Peroni, David Shotton, and Fabio Vitali

YAM++ : A Multi-strategy Based Approach for Ontology Matching Task 421
DuyHoa Ngo and Zohra Bellahsene

User-Friendly Pattern-Based Transformation of OWL Ontologies 426
Ondřej Šváb-Zamazal, Marek Dudáš, and Vojtěch Svátek

Guided Semantic Annotation of Comic Panels with Sewelis 430
Alice Hermann, Sébastien Ferré, and Mireille Ducassé

I-CAW: Intelligent Data Browser for Informal Learning Using Semantic Nudges 434
Dhavalkumar Thakker, Vania Dimitrova, and Lydia Lau

RightField: Scientific Knowledge Acquisition by Stealth through Ontology-Enabled Spreadsheets 438
Katy Wolstencroft, Stuart Owen, Matthew Horridge, Wolfgang Mueller, Finn Bacall, Jacky Snoep, Franco du Preez, Quyen Nguyen, Olga Krebs, and Carole Goble

TrustMe, I Got What You Mean!: A Trust-Based Semantic P2P Bookmarking System 442
Mustafa Al-Bakri, Manuel Atencia, and Marie-Christine Rousset

NIF Combinator: Combining NLP Tool Output 446
Sebastian Hellmann, Jens Lehmann, Sören Auer, and Marcus Nitzschke

Author Index 451

Open PHACTS: A Semantic Knowledge Infrastructure for Public and Commercial Drug Discovery Research

Lee Harland*

The Open PHACTS Consortium
ConnectedDiscovery Ltd, London, UK
pmu@openphacts.org

1 Introduction

Technology advances in the last decade have led to a "digital revolution" in biomedical research. Much greater volumes of data can be generated in much less time, transforming the way researchers work [1]. Yet, for those seeking to develop new drugs to treat human disease, the task of assembling a coherent picture of existing knowledge from molecular biology to clinical investigation, can be daunting and frustrating. Individual electronic resources remain mostly disconnected, making it difficult to follow information between them. Those that contain similar types of data can describe them very differently, compounding the confusion. It can also be difficult to understand exactly where specific facts or data points originated or how to judge their quality or reliability. Finally, scientists routinely wish to ask questions that the system does not allow, or ask questions that span multiple different resources. Often the result of this is to simply abandon the enquiry, significantly diminishing the value to be gained from existing knowledge. Within pharmaceutical companies, such concerns have led to major programmes in data integration; downloading, parsing, mapping, transforming and presenting public, commercial and private data. Much of this work is redundant between companies and significant resources could be saved by collaboration [2]. In an industry facing major economic pressures [3], the idea of combining forces to "get more for less" is very attractive and is arguably the only feasible route to dealing with the exponentially growing information landscape.

The development of a scalable, semantic system holding critical, interoperable decision-making data could provide many benefits regarding the issues outlined above. Approaches based in semantic web technologies are an attractive option in part due to their technical capabilities, but critically, they also provide an open standard at the core of such a shared infrastructure. Increasingly, pharmaceutical companies are outsourcing elements of drug discovery and development to contract research organisations and academic collaboration. This makes efficient data exchange between partners a critical need for the future [4]. A vendor-neutral infrastructure built upon widely adopted, open standards, provides the widest possible potential for adoption across commercial and non-profit sectors alike. It is with this goal in mind

* Corresponding author.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 1–7, 2012.

that the Open PHACTS (Open **Pha**rmacological **C**oncept **T**riple **S**tore) project [5] was conceived. Open PHACTS is a major public-private partnership involving organisations from major pharmaceutical companies, academia and small-medium enterprises (SMEs). The project is funded by the European Federation of Pharmaceutical Industries and Associations (EFPIA) and the European Union through the Innovative Medicines Initiative [6] and scheduled to complete in early 2014.

2 Strategic Aims

Semantic technologies have already gained much traction within biomedicine with initiatives such as the World Wide Web Consortium Semantic Web Health Care and Life Sciences Interest Group [7] Bio2RDF [8], Chem2Bio2RDF [9] and many others. Open PHACTS is complementary to such efforts, focusing on the creation of a task focused, production-grade and sustainable public-private infrastructure. This latter point is critical, as the output from the project should form the foundation for future pre-competitive efforts. As the scientific domain is large, the initial phase of the project focused on defining scope by generation and ranking of the key scientific questions that scientists within drug discovery would like to be able to ask of such a system [10]. An excerpt from these questions is shown in Box 1 and provides clear direction for both the data required and software functionality, as well as strong criteria for measuring success.

- For a given compound, summarize all 'similar compounds' and their activities
- A lead molecule is characterized by a substructure S. Retrieve all bioactivity data in serine protease assays for molecules that contain substructure S.
- For a specific target, which active compounds have been reported in the literature? What is also known about upstream and downstream targets?
- Find homologous targets to my target of interest and identify active compounds against these homologous targets.
- For a given disease/indication provide all targets in the pathway and all active compounds hitting them

Box 1. Example scientific questions the Open PHACTS system will address

3 Data Standards and Quality

High quality data is at the heart of this endeavour. Wherever possible, Open PHACTS will re-use and augment existing efforts in the life-sciences domain by promoting the

publishing of pharmacological data in RDF using W3C standards and publically available ontologies. By actively engaging with data providers, we hope to further develop the standards and tooling require for full semantic representation of required content. For instance, we have collaborated with the ChEMBL [11] group, to implement the Quantities, Units, Dimensions and Types ontology (QUDT, [12]) to enable quantitative data queries and interconversion between different classes of measurement. Quality issues are also being addressed, particularly the representation of chemicals on the semantic web. This is still far from optimal and often due to inaccuracies in the electronic representation of these molecules in their source databases through error or subtly different algorithms for creating them. When these are published as linked data, incorrect or missing links have a significantly detrimental effect on data analysis. To address this, the project has developed very detailed guidelines for structure processing and normalisation (based on guidelines from the US Food and Drug Administration [13]) that will deliver more consistency between different databases. In addition, RDF describing novel chemical structure quality metrics and multiple physiochemical properties is being generated for each chemical using software from ChemSpider [14] and ACD/Labs [15] respectively, contributing novel and important data to the community.

Prominence is also given to the provenance of the data within the system, to benefit both consumers (who know where a particular "fact" came from) and producers (crediting the original source). Each Open PHACTS data set is accompanied by a VoID [16] based specification, enhanced with provenance information encoded using the Provenance Authoring and Versioning ontology [17]. We are also actively contributing to the W3C Provenance task [18] to help define, and ensure alignment with this emerging standard. Finally, the project is using nanopublications [19] to record information for individual assertions, both from databases and those generated by individuals through annotation tools. By enabling users to understand where the answer to their query actually came from, we hope to promote data citation [20] and provide credit to those producing important scientific assertions.

4 Technical Architecture

The overall architecture of the Open PHACTS core platform is shown schematically in Fig. 1. The approach is to create a modular system, based on the reuse of existing software and protocols rather than developing these from scratch. While this still requires development to make these components robust enough for a "production-grade" system, it leverages and supports existing efforts by the life science informatics community.

As outlined above, data are sourced as RDF and VoID descriptors are created if not already present. These data are loaded into a triple store and are expected to be in the range of 50-80 billion triples by the end of the project. As humans think in terms of natural language and not RDF, the Identity Resolution Service (provided by ConceptWiki [21]) recognises the objects within user-entered text strings and output corresponding URIs for downstream queries. Unfortunately, concepts in life science

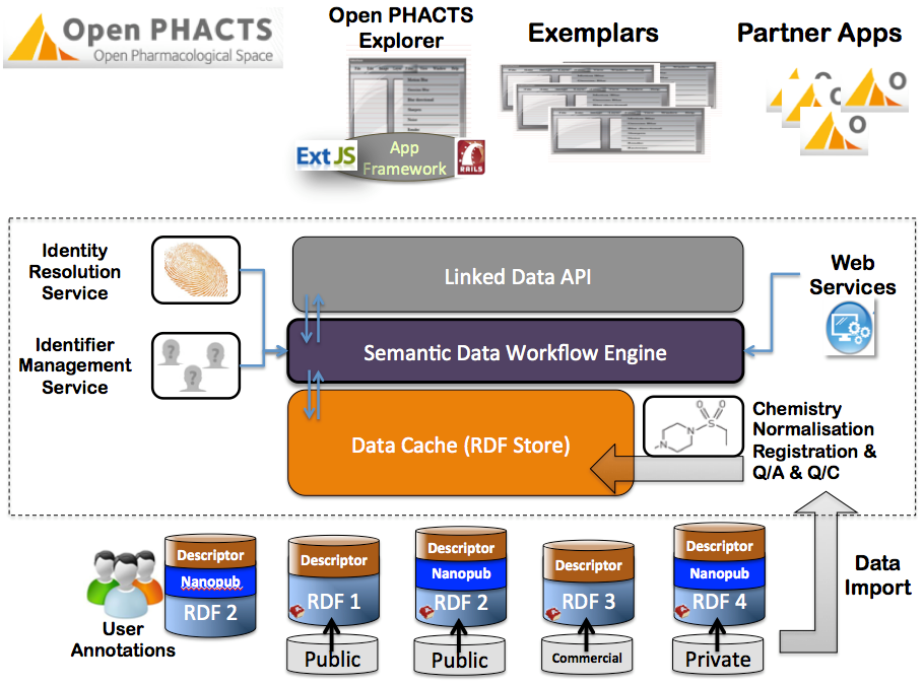


Fig. 1. The Open PHACTs Core Architecture

suffer from a proliferation of identifiers and URIs [22] often making cross-linking of data difficult. However, prescribing rigid rules for URIs that data must use (e.g. "Open PHACTS only accepts "purl.org"-based Uniprot identifiers for proteins) would significantly limit the content that can be consumed. Even worse, conversion and over-writing of identifiers in the data source would mean that the system no longer faithfully represents the original data and make it more difficult to track down the cause of unexpected results. Therefore, Open PHACTS takes a liberal approach and accepts any identifier namespace and syntax for a given entity class, as long as mappings to URIs that are already known to the system are provided. These mappings are resolved at query time using the Identifier Mapping Service (IMS), which incorporates a semantic version of the BridgeDB system [23]. An additional benefit of this is that the definition of what is "the same" can be adjusted to suit the query. These definitions will be governed by the use of profiles that instruct the IMS which URI classes can be combined. For instance, under certain circumstances it may be reasonable to join data attached to URIs for a protein and its equivalent gene or between a chemical compound and a drug of which it is a part; but not in others.

The IRS and IMS are distinct modules, exposing functionality via web-services. The Large Knowledge Collider (LarKC, [24]) provides the necessary middleware required to combine invocation of these services with data in the triple store within a single SPARQL query. This allows for further federation, for example, incorporating

the results of live chemical structure similarity searches (provided by the ChemSpider API) into the current query. Ultimately, it is at this point that the SPARQL queries integrate data by joining across multiple different data sets and address the specific questions that cannot be otherwise easily answered by scientists. These queries are packaged into the Open PHACTS API, implementing the Linked Data API specification [25] and providing a highly accessible mechanism for non-semantic web developers to access the system via the RESTful/JSON paradigm. The API is used by the "Open PHACTS Explorer" which is a web-based user interface to the system, built on the Lundbeck Life Science Platform [26]. The Explorer will allow end-users to browse data within the platform, perform simple queries and aggregations and export results for use in downstream tools such as chemistry analysis software, pathway analyses and Excel.

5 Sustainability

While the Open PHACTS Explorer will address the immediate need to deliver a tool to end-users, it is not intended to be the only portal to the system. Rather, the major goal of the project is to create an "application ecosystem" in which non-profit and commercial organisations consume data via the Open PHACTS API for specific scientific applications. We hope that the availability of a professionally hosted, high-quality, integrated pharmacology data platform, developed using standards endorsed by many major pharmaceutical companies should present an attractive resource that commercial organisations can licence and use to enhance their own offerings. Combined with other revenue streams, this will be crucial in tackling perhaps the biggest challenge within the project, namely sustaining the development of this unique infrastructure after the current funding expires. To that end, the project includes a number of exemplar applications, designed to demonstrate this principle and "kick-start" the application offerings. In addition, Open PHACTS has a flourishing partnership programme, by which interested parties can learn more and identify opportunities for collaboration or API deployment.

6 Conclusion

Open PHACTS is a unique initiative, bringing major industry and non-profit groups together to develop a shared platform for integration and knowledge discovery. The project aims to deliver on multiple fronts, enhancing the quality of relevant RDF data, addressing key scientific bottlenecks, developing and promoting open standards and creating a lasting infrastructure for cross-sector collaboration. A first version of the software is due for release in late 2012 and will be announced via the project website [5]. We hope that the project will demonstrate that semantic technologies are ready for "prime-time" use as a dependable, core infrastructure for future initiatives in drug discovery.

Acknowledgements. The Open PHACTS project consortium consists of leading experts in semantics, pharmacology and informatics including academic institutions, pharmaceutical companies and scientific software companies. The work described here is the result of contribution by all collaborators in the project. The financial support provided by the IMI-JU project Open PHACTS, grant agreement n◦ 115191 is gratefully acknowledged. Finally, I would like to thank Niklas Blomberg, Carole Goble, Alasdair Gray, Paul Groth, Antonis Loizou and Antony Williams for valuable input and feedback on the article.

References

1. Kell, D.B., Oliver, S.G.: Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era. Bioessays 26, 99–105 (2004)
2. Barnes, M.R., Harland, L., Ford, S.M., Hall, M.D., Dix, I., Thomas, S., Williams-Jones, B.I., et al.: Lowering industry firewalls: pre-competitive informatics initiatives in drug discovery. Nat. Rev. Drug Discov. 8, 701–708 (2009)
3. Scannell, J.W., Blanckley, A., Boldon, H., Warrington, B.: Diagnosing the decline in pharmaceutical R&D efficiency. Nat. Rev. Drug Discov. 11, 191–200 (2012)
4. Harland, L., Larminie, C., Sansone, S.A., Popa, S., Marshall, M.S., Braxenthaler, M., Cantor, M., et al.: Empowering industrial research with shared biomedical vocabularies. Drug Discov. Today 21, 940–947 (2011)
5. http://openphacts.org
6. http://imi.europa.eu
7. http://www.w3.org/blog/hcls
8. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems 41, 706–716 (2008)
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 12, 256 (2010)
10. Azzaoui, K., Jacoby, E., Senger, S., Rodríguez, E.C., Loza, M., Zdrazil, B.: Scientific questions to a next generation semantically enriched biomolecular internet resource (in Preparation)
11. Gaulton, A., Bellis, L.J., Bento, A.P., Chambers, J., Davies, M., Hersey, A., Light, Y., et al.: ChEMBL: a large-scale bioactivity database for drug discovery 40, D1100–D1107 (2011)
12. http://www.qudt.org
13. http://1.usa.gov/snNNdn
14. http://www.chemspider.com
15. http://www.acdlabs.com
16. http://vocab.deri.ie/void
17. http://swan.mindinformatics.org/spec/1.2/pav.html
18. http://www.w3.org/TR/prov-primer
19. Groth, P., Gibson, A., Velterop, J.: The anatomy of a nanopublication. Information Services and Use 30, 51-56 (2010)
20. Mons, B., van Haagen, H., Chichester, C., Hoen, P.B., den Dunnen, J.T., van Ommen, G., van Mulligen, E., et al.: The value of data. Nat. Genet. 43, 281–283 (2011)
21. http://conceptwiki.org

22. http://www.slideshare.net/dullhunk/the-seven-deadly-sins-of-bioinformatics
23. van Iersel, M.P., Pico, A.R., Kelder, T., Gao, J., Ho, I., Hanspers, K., Conklin, B.R., et al.: The BridgeDB framework: standardized access to gene, protein and metabolite identifier mapping services. BMC Bioinformatics 11, 5 (2010)
24. Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Emanuele, D.V., Fischer, F., et al.: Towards LarKC: a Platform for Web-scale Reasoning. In: Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2008), vol. 8 (2008)
25. http://code.google.com/p/linked-data-api/
26. Kallesøe, C.S.: Building scalable solutions with open source tools. In: Harland, L., Forster, F. (eds.) Open Source Software in Life Science Research. Woodhead Publishing Series in Biomedicine, London (2012)

Allowing End Users to Query Graph-Based Knowledge Bases

Camille Pradel

IRIT, Université de Toulouse le Mirail,
Département de Mathématiques-Informatique, 5 allées Antonio Machado,
F-31058 Toulouse Cedex
camille.pradel@univ-tlse2.fr

Abstract. Our purpose is to provide end users a means to query knowledge bases using natural language queries and thus hide the complexity of formulating a query expressed in a graph query language such as SPARQL. The main originality of our approach lies in the use of query patterns. Our contribution is materialized in a system named *SWIP*, standing for *Semantic Web Interface Using Patterns*, which is situated in the Semantic Web framework. This paper presents the main issues addressed by our work and establishes the list of the important steps (to be) carried out in order to make SWIP a fully functional system.

1 Introduction

In this article, we present an approach aiming at simplifying the formulation of queries expressed in graph languages such as SPARQL. Two kinds of works have already been proposed to achieve this goal. The first one aims at helping the user to formulate his query in a language dedicated to the interrogation of graph bases. Such an approach is not well suited for final users since it requires the user to know the syntax of the query language and the general schema of the data managed by the system. The help provided to the user can rely on graphical interfaces such as ICS-FORTH [1] for RQL queries, NITELIGHT [14] and Smeagol [2] for SPARQL queries or COGUI [3] for conceptual graph queries which can be requested on querying systems such as [7]. Even if these graphical interfaces are useful for final users, the latter need to be familiar with such an interface and, moreover, they have to understand the semantics of the expression of queries in terms of graphs, which is not that natural. The work presented in [5] aims at extending the SPARQL language and its querying mechanism in order to take into account keywords and wildcards when the user does not know exactly the schema he/she wants to query on. Here again, such an approach requires that the user knows the SPARQL language.

Other works aim at generating automatically – or semi-automatically – formal queries from user queries expressed in terms of keywords or natural language. Our work is situated in this family of approaches. The user expresses his/her information need in an intuitive way, without having to know the query language or the knowledge representation formalism used by the system. Some works have

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 8–15, 2012.

already been proposed to express formal queries in different languages such as SeREQL [9], SPARQL [18,15] or conceptual graphs [4].

In these systems, the generation of the query requires the following steps: (i) matching the keywords to semantic entities defined in the knowledge base, (ii) building query graphs linking the entities previously detected by exploring the knowledge base, (iii) ranking the built queries, (iv) making the user select the right one. The existing approaches focus on three main issues: optimizing the first step by using external resources (such as WordNet or Wikipedia)[9,16], optimizing the knowledge exploration mechanism for building the query graphs [18,15], and enhancing the query ranking score [16]. Autosparql [8] extends this category: after a basic interpretation of the user NL query, the system interacts with the user, asking for positive and negative examples (i.e. elements which are or are not in the list of expected answers), in order to refine the initial interpretation relying on supervised active learning algorithm.

The main issue we address in our work is the same as the second category of approaches presented above, i.e. interpreting a natural (or at least high level) language query and translating it in a formal graph language. The main postulate leading our work states that, in real applications, the submitted queries are variations of a few typical query families. Our approach differs from existing ones in the way that we propose to enhance the effectiveness and the efficiency of the query building step by using predefined query patterns which represent these query families. The use of patterns avoids exploring the ontology to link the semantic entities identified from the keywords since potential relations are already expressed in the patterns. The process thus benefits from the pre-established families of frequently expressed queries for which we know that real information needs exist. This idea makes the interpretation easier but raises a new important issue: generating these query patterns.

In Section 2, we describe our approach. Then, in Section 3, we present the implementation of our work in the SWIP system and the first evaluation results, before concluding and stating remaining tasks in Section 4.

2 Proposed Approach

In the SWIP system, the query interpretation process is made of two main steps: the translation of the NL user query into a pivot query, and the formalization of this pivot query, respectively described in subsections 2.1 and 2.2. In Subsection 2.3, we propose some ideas we consider to carry out in order to automatise the process of building patterns which are for the time being determined manually.

2.1 From Natural Language to Pivot Query

As explained earlier, the whole process of interpreting a natural language query is divided into two main steps, with an intermediate result, which is the user query translated into a new structure called the *pivot query*. This structure is half way between the NL query and the targeted formal query, and can be expressed

through a language, called pivot language, which is formally defined in [13]. Briefly, this structure represents a query made of keywords and also expresses relations between those keywords. This translation step consists of four stages.

The first of these stages is the query type identification; it aims at recognizing the type of query that is being processed. This identification is performed by comparing the beginning of the query sentence to a set of predefined strings. For the time being, the SWIP system distinguishes three types of queries which are defined by the form of the expected result:

- *list queries* expect a list of resources fulfilling certain conditions as a result and their SPARQL translation is a "classical" `SELECT` query; such queries start with "List", "List all", "I am looking for", "What are"...

  Example 1. List all members of The Notwist.

- *count queries* ask for the number of resources fulfilling certain conditions and correspond in SPARQL to a `SELECT` query using a `COUNT` aggregate as a projection attribute; such queries start with "How many", "What is the number"...

  Example 2. How many albums did Michael Jackson record?

- *dichotomous (or boolean) queries* allow only two answers, True or False (alternatively Yes or No), and are expressed in SPARQL with an `ASK` query; such queries start with "Is", "Does", "Is it true that"...

  Example 3. Was Keith Richards a member of The Rolling Stones?

After the identification of the query type, the substring that allowed this identification is removed from the sentence and the remaining part is considered for further treatment. In the case when no query type can be inferred, the query is considered to be a list query. Thus we allow the end user to express that kind of query without formulating a complete sentence, as in Example 4.

Example 4. members of The Notwist.

The second stage aims at identifying in the remaining part of the sentence named entities corresponding to knowledge base resources. This allows these entities to be considered as a whole and prevents the parser from separating them in the next stage. This stage is particularly crucial when querying knowledge bases containing long labels, like group names or track titles in a music knowledge base, made of several words or even sometimes of a full sentence.

In the third stage, dependency analysis is performed on this same sentence part, taking into account the previously identified named entities. The objective of this parsing stage is to identify the head of the sentence and obtain a dependency graph expressing syntactical relations between substrings of the sentence. The nature and semantics of these relations depend on the parser we are using.

Finally, a set of predefined rules are applied to the obtained dependency graph in order to obtain the pivot query. These rules obviously depend on the previously used parser and are manually defined. A rule consists of two parts: the first part describing a condition that must be spotted in the dependency graph, and the second part describing the pivot query chunk that must be produced when the condition appears. Then, in the obtained structure, we need to identify the query object, i.e. the entity in which the user is particularly interested, which appears as a projection attribute in the SPARQL translation. We consider the substring identified as the head of the sentence to be the query object.

2.2 From Pivot to Formal Query

Formalizing pivot queries using query patterns was the first task we tackled and is thus extensively described in [12] and [13]. We briefly describe the structure of a query pattern and the process of this formalization.

A pattern is composed of an RDF graph which is the prototype of a relevant family of queries. Such a pattern is characterized by a subset of its elements – either class, property or literal type –, called the qualifying elements, which can be modified during the construction of the final query graph. It is also described by a sentence in natural language in which a distinct substring must be associated with each qualifying element. For now, the patterns are designed by experts who know the application domain. The designer of a pattern builds its RDF graph manually, selects its qualifying elements and also gives the describing sentence.

The process of this step is as follows. Each element of the user query expressed in the pivot language is matched to an element of the knowledge base. Elements of the knowledge base can either be a class, a property, an instance or a literal type (which can be any type supported by SPARQL, i.e. any type defined in RDF Schema). Then we map query patterns to these elements. The different mappings are presented to the user by means of natural language sentences. The selected sentence allows the final SPARQL query to be built.

A recent evolution of the pattern structure makes the patterns more modular and the query generation more dynamic. We can now assign values of minimal and maximal cardinalities to subgraphs of the patterns, making these subgraphs optional or repeatable when generating the formal query. The descriptive sentence presented to the user also benefits from this novelty and no longer contains non relevant parts (parts of the pattern which were not addressed by the user query), making thus our system more ergonomic.

2.3 Pattern Generation

The preliminary step of building query patterns is done manually and thus requires a large amount of work. Moreover, this step must be repeated each time we want to address a new knowledge base. This is why the automatic or assisted pattern generation is the last important task we need to carry out to obtain a fully functional system.

Two leads appear to be followable for this purpose: building patterns covering a set of graph queries, or learning these patterns from a set of natural language queries. At a first glance, the first considered method seems to be the easiest to implement, since the input of this method are formal structures which are easy to handle. However, end user queries expressed in a graph formalism could be costly to obtain in practice, when natural language queries on a domain should be easy to find, looking on forums, FAQs, or simply asking users.

We plan to begin by developing the idea of building patterns from graph queries. Initially, we aim at automatically determine only the graph of each pattern, which is the most time consuming task for experts defining the patterns; we also consider making a first guess of qualifying elements and optional or repeatable parts; refining this guess and writing the explicative sentence will remain under the responsibility of the experts. While designing this method, several objectives must be kept in mind: obtaining a set of patterns as small as possible, covering all or the majority of the sample queries, and generating patterns which are not too general and of reasonable size. The formulations "not too general" and "of reasonable size" still need to be formally characterized, but it is intuitively easy to understand that a big graph exclusively made of `owl:Thing` vertices and `owl:ObjectProperty` edges would certainly cover a very large proportion of queries (because generated queries are specifications of patterns) but would not be of great use to interpret user queries.

As regards NL queries, we would like to apply methods of query classification, inspired by those presented in [10], [17] or [11], and compare the obtained classes to the patterns learned from graph queries. Resulting observations could lead to discussions aiming at designing a process for pattern learning from NL query examples.

3 Evaluation

A prototype of our approach has been implemented in order to evaluate its effectiveness. It has been implemented in Java and uses the Supple parser [6] for the dependency analysis of English user queries. The system performs the second main process step (translating from pivot to formal query) by exploiting a SPARQL server based on the ARQ[1] query processor, here configured to exploit LARQ[2], allowing the use of Apache Lucene[3] features, such as indexation and Lucene score (used to obtain the similarity score between strings).

Experiments have been carried out on the evaluation framework proposed by the QALD-1 challenge[4], in which we participated. It is based on MusicBrainz, i.e. a music ontology and 50 million associated triples. We built patterns with a set of 50 training queries, provided by the challenge organizers in the form of natural language queries and their SPARQL translation, and then we proceeded with

[1] `http://openjena.org/ARQ/`
[2] LARQ = Lucene + ARQ, see `http://jena.sourceforge.net/ARQ/lucene-arq.html`
[3] `http://lucene.apache.org/`
[4] `http://www.sc.cit-ec.uni-bielefeld.de/qald-1`

the evaluation by translating into the pivot language 50 different test queries (given only in natural language). Although these experiments concerned only the second part of the process, results were encouraging.

The evaluation method was imposed by the challenge organizers. It consists in calculating, for each test query, the precision, the recall and the F-mesure of the SPARQL translation returned by the system, compared with handmade queries of a gold standard document. The results are summarized in the following table, which presents, for SWIP and for FREyA, another system which took part in the challenge, the total number of queries the systems had to process, the number of actually processed queries, the number of correctly interpreted queries (i.e. their SPARQL translations gave exactly the same results as gold standard SPARQL queries), the number of wrongly interpreted queries (i.e. the other processed queries), and the overall precision, recall and F-measure.

Table 1. FREyA and SWIP results at the QALD-1 challenge

	total	processed	right	wrong	recall	precision	f-measure
FREyA	50	41	30	11	0.6	0.73	0.66
SWIP	50	35	28	7	0.56	0.8	0.66

These results are satisfactory, but could be improved: our implementation does not handle some kinds of queries, which moreover cannot be expressed in the pivot language (for instance, queries like "What is the longest song by John Cage?" or "Which person recorded the most singles?"). We ignored these kinds of queries, which explains the relatively low recall despite quite a good precision.

4 Conclusion and Future Work

In this paper, we presented the approach we are designing to allow end users to query graph-based knowledge bases. This approach is implemented in the SWIP system and is mainly characterized by the use of query patterns leading the interpretation of the user NL query and its translation into a formal graph query. Although some arrangements still need to be carried out, the setting up of the two main parts of the system process is almost done and first results are very encouraging. We planed to extend our work in two directions:

- performing new experiments taking into account the translation from NL query into pivot query; we will probably use the QALD challenge data again for experiments on English queries, and we are developing a partnership with the IRSTA (a French institute on ecology and agriculture) in order to build a real application framework concerning French queries on organic farming.
- experimenting methods to automate or assist the conception of query patterns; we first want to automatically determine the pattern structures by analysing graph query examples, and then compare the developed method(s) to an approach based on NL queries learning.

References

1. Athanasis, N., Christophides, V., Kotzinos, D.: Generating On the Fly Queries for the Semantic Web: The ICS-FORTH Graphical RQL Interface (GRQL). In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 486–501. Springer, Heidelberg (2004)
2. Clemmer, A., Davies, S.: Smeagol: A "Specific-to-General" Semantic Web Query Interface Paradigm for Novices. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011, Part I. LNCS, vol. 6860, pp. 288–302. Springer, Heidelberg (2011)
3. CoGui: A conceptual graph editor. Web site (2009), http://www.lirmm.fr/cogui/
4. Comparot, C., Haemmerlé, O., Hernandez, N.: An Easy Way of Expressing Conceptual Graph Queries from Keywords and Query Patterns. In: Croitoru, M., Ferré, S., Lukose, D. (eds.) ICCS 2010. LNCS, vol. 6208, pp. 84–96. Springer, Heidelberg (2010)
5. Elbassuoni, S., Ramanath, M., Schenkel, R., Weikum, G.: Searching rdf graphs with sparql and keywords. IEEE Data Eng. Bull. 33(1), 16–24 (2010)
6. Gaizauskas, R., Hepple, M., Saggion, H., Greenwood, M.A., Humphreys, K.: Supple: A practical parser for natural language engineering applications. In: International Workshop on Parsing Technologies, p. 200 (2005)
7. Genest, D., Chein, M.: A content-search information retrieval process based on conceptual graphs. Knowl. Inf. Syst. 8(3), 292–309 (2005)
8. Lehmann, J., Bühmann, L.: AutoSPARQL: Let Users Query Your Knowledge Base. 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. 63–79. Springer, Heidelberg (2011)
9. Lei, Y., Uren, V.S., Motta, E.: SemSearch: A Search Engine for the Semantic Web. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 238–245. Springer, Heidelberg (2006)
10. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics, vol. 1, pp. 1–7. Association for Computational Linguistics (2002)
11. Merkel, A.P., Klakow, D.: Language Model Based Query Classification. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 720–723. Springer, Heidelberg (2007)
12. Pradel, C., Haemmerlé, O., Hernandez, N.: Expressing Conceptual Graph Queries from Patterns: How to Take into Account the Relations. In: Andrews, S., Polovina, S., Hill, R., Akhgar, B. (eds.) ICCS-ConceptStruct 2011. LNCS, vol. 6828, pp. 229–242. Springer, Heidelberg (2011)
13. Pradel, C., Haemmerlé, O., Hernandez, N.: A Semantic Web Interface Using Patterns: The SWIP System. In: Croitoru, M., Rudolph, S., Wilson, N., Howse, J., Corby, O. (eds.) GKR 2011. LNCS, vol. 7205, pp. 172–187. Springer, Heidelberg (2012)
14. Russell, A., Smart, P.R.: Nitelight: A graphical editor for sparql queries. In: International Semantic Web Conference. Posters & Demos (2008)
15. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In: ICDE, pp. 405–416 (2009)

16. Wang, H., Zhang, K., Liu, Q., Tran, T., Yu, Y.: Q2Semantic: A Lightweight Keyword Interface to Semantic Search. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 584–598. Springer, Heidelberg (2008)
17. Zhang, D., Lee, W.: Question classification using support vector machines. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pp. 26–32. ACM (2003)
18. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: SPARK: Adapting Keyword Query to Semantic Search. 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. 694–707. Springer, Heidelberg (2007)

Nichesourcing: Harnessing the Power of Crowds of Experts

Victor de Boer[1], Michiel Hildebrand[1], Lora Aroyo[1], Pieter De Leenheer[1,2], Chris Dijkshoorn[1], Binyam Tesfa[1], and Guus Schreiber[1]

[1] The Network Institute, Dep. of Computer Science, VU University, Amsterdam, The Netherlands
{v.de.boer,m.hildebrand,l.m.aroyo, p.g.m.de.leenheer,c.r.dijkshoorn,guus.schreiber}@vu.nl, b.m.tesfa@student.vu.nl
[2] Collibra nv/sa, Brussels 12, Belgium

Abstract. In this position paper we identify nichesourcing, a specific form of human-based computation that harnesses the computational efforts from niche groups rather than the "faceless crowd". We claim that nichesourcing combine the strengths of the crowd with those of professionals, optimizing the result of human-based computation for certain tasks. We illustrate our claim using scenarios in two domains: cultural heritage and regreening in Africa. The contribution of this paper is to provide a definition of the main characteristics of nichesourcing as a natural extension of crowdsourcing and to outline research challenges for realizing nichesourcing applications.

1 Introduction

In 2005 Luis Von Ahn coined *"a paradigm for utilizing human processing power to solve problems that computers cannot yet solve"* [1]. Since then it has grown into a research field called *human-based computation* applied successfully in various domains for a great number of tasks that rely on the advantage that humans have over machines in skills such as visual recognition and human communication [2]. A popular and well-described form of human-based computation is *crowdsourcing* where the computational task is delegated to a *crowd*, i.e. a very large and redundant workforce of people. Digitization, classification, translation and annotation are usually split into many simple tasks, that can be performed by anonymous people without any specific skills. The crowd is gathered via a more or less open call [3] in a (social) network [4]. In some cases, such as the Amazon's Mechanical Turk, participants are paid small fees for their effort [5]. However, as most initiatives do not provide financial reward, other motivational incentives are deployed in order to attract a sufficient amount of people [6].

Many institutions and companies are currently utilizing the knowledge of the crowd as an alternative to professional efforts. Initially, these crowdsourcing initiatives were centered around users performing *simple tasks* and their overall target was geared towards *achieving quantity* rather than quality. Currently, we observe [7] that there is (i) a growing demand for solving also *complex knowledge-intensive tasks* [8], and (ii) a natural expectation to focus on the quality of the final result [9]. In this paper, we argue that *nichesourcing* is the next natural step in the evolution of crowdsourcing to address

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 16–20, 2012.

those two demands. Nichesourcing is a specific type of crowdsourcing where complex tasks are distributed amongst a small crowd of amateur experts (for example art enthusiast or African ex-pats) rather than the "faceless" crowd. A niche is gathered from either distributed experts on a specific topic or from an existing network centered around the same culture, location or topic. In both cases the members have domain knowledge and an intrinsic motivation to contribute and provide high quality results [10].

The contribution of this paper is to provide a definition of the main characteristics of nichesourcing as a natural extension of crowdsourcing (Sec. 2), and to identify the research challenges for its realization (Sec. 4). We describe two use cases for which we are currently implementing nichesourcing solutions (Sec. 3).

2 Harnessing the Crowd and the Niche: Comparison

In this section, we provide a basic framework to compare three main aspects of crowdsourcing and nichesourcing. We identify (from a process-centric perspective) three main dimensions in this framework, namely *the task* and its complexity, *the product* that is targeted as a result of the task, and *the resource pool* of users needed in order to realize this task. The purpose of this comparison is to explain the fundamental differences between crowdsourcing and nichesourcing, and to identify for what types of tasks, for what desired result and for what type of "crowd" each of them would be most suitable. This framework supports making an informed decision whether to apply crowdsourcing or nichesourcing for a specific human computation problem.

The Atomic Task: Simple vs. Knowledge-intensive. Crowdsourcing deals with large complex tasks by dividing it up into smaller, *atomic tasks*, the latter which do not require specific knowledge or skills from the crowd members. Consider, e.g., different peers tagging the same image for triangulation purposes. Crowdsourcing typically focuses on repeatable atomic tasks and can be pitched easily through broad syndication channels to random pools of resources. For some complex tasks, however, the atomic tasks are too hard to execute for any random crowd member for two reasons: either (i) the atomic task itself requires specific background knowledge that cannot be assumed to be present in the crowd; or (ii) it will take so much effort that crowdsourcing motivation techniques are not sufficient to engage a significantly sized crowd. Atomic tasks that require a specific level of knowledge or effort can be outsourced to niches in which the members possess particular knowledge or motivation is present.

Product: Quantity versus Quality. The success of the crowdsourced product is determined by quantity. As there is no specific qualification for the targeted resource pool, the quality of the product can only be expressed (i) either globally (e.g., accuracy of the results) or locally (e.g., specificity of individual tags). Hence, typical crowdsourcing solutions require careful application design as well as post-processing of the produced results. For example, by awarding users producing good results. Post-processing typically involves aggregating the results and exploiting redundancy to determine the most likely correct and/or most specific results. Complex nichesourcing tasks strategically target socially-trusted *communities of practice* [11] (see further). The rationale for such strategy is that an acceptable number of appropriately skilled resources are assumed to

have the intrinsic courtesy to provide higher quality individual results. This removes much of the need for statistical processing. Secondly, experts typically provide more specific input than non-experts. If the task requires a more controllable level of quality, nichesourcing is a good alternative. This gain in quality control comes at a cost of quantity, since we assume a community of practice to be smaller than any subset of Web users that can be considered a crowd.

Resource Pool: Crowd vs. Community of Practice. Quinn et al. [2] identify five methods for motivation in human-based computation applications: *payment, altruism, enjoyment, reputation* and *implicit work*. Building and maintaining a dedicated crowd is essential to crowdsourcing applications that use altruism, reputation and (to a lesser extent) enjoyment as motivation. A key measure of success is the size of this crowd, and to a certain extent, the level of redundancy (necessary for statistical processing). Such a crowd is usually anonymous and heterogenous, i.e., there is no shared goal or any social affinity required whatsoever. In nichesourcing, composite and complex tasks are distributed within existing communities. Communities, in contrast to crowds, have a common purpose, peers have an identity and affinity with this purpose, and their regular interactions engenders social trust and reputation. These niches can correspond to the notion of a community of practice or interest[11]. Although communities, in contrast to crowds, provide smaller pools to draw resources from, their specific richness in skill is suited for the complex tasks with high-quality product expectations found in nichesourcing. Moreover, the peer resources receptive to complex tasks may exploit their own social trust relations to transitively trigger other communities that may offer reinforcing resource pools.

3 Two Use Cases for Nichesourcing

Rijksmuseum Prints Annotation. The Print Room of the Rijksmuseum in Amsterdam has a collection of about 700 000 prints, drawings and photographs. Within the project *Print Room Online* they register the basic properties of each print, such as the object ID, storage location, title, creator and measurements. In addition, they describe the subject matter of the prints. The annotation is performed by eleven professional cataloguers which are expected to describe 20% of the collection during the 3 years of the project.

Clearly, there is a need for more human effort in order to describe the entire collection of the Print Room. Crowdsourcing seems like the natural solution to the problem. Indeed, many museums and cultural heritage archives have already embraced the notion of human-computing in order to solve the same problem. Typically, they apply crowdsourcing techniques [7], [12]. As a result they receive large quantities of metadata, but not necessarily of sufficient quality [13]. For the Rijksmuseum, the quality of the produced annotations is a prime concern. They are interested in highly domain specific annotations, such as the names of the people, places and events depicted on the prints, and they prefer specific annotations of the objects and concepts, e.g. “the symbolic meaning of a frog in a Japanese print”, over generic annotations, e.g. “frog”. Therefore, the “faceless crowd” is not the right target and instead the Rijksmuseum is in need of hobbyists, self-thought experts and retired professionals that can perform this knowledge-intensive task.

Digitizing Pluvial Data from the Sahel. Governments in the African Sahel region have recorded 1000s of pages of data about rainfall and crop harvest in their region. This data is very useful when aggregated over multiple regions for analysis supporting decisions in re-greening initiatives. For this goal, low resolution digital scans have been made of handwritten documents containing tabular as well as textual data[1]. The data in these documents is to be converted into digitized structured data. Automated techniques digitization are error-prone and require a lot of configuration. Although crowdsourcing has been used for digitizing handwritten documents (e.g. [14]), this specific task is fairly complex in the sense that (i) the semantics of the tables is often not easily understood; and (ii) decoding the handwriting does require specific language and domain knowledge (e.g., familiarity with the regional geography and villages). We expect that the level of quality that the faceless crowd can provide is not sufficient for the goals of the re-greening initiative and that therefore this task is very well suited for nichesourcing. The niche being targeted is the so-called *African Diaspora*: African expatriates who now reside in better connected parts of the world. Members of the diaspora are very much affiliated with local issues in their region of origin. This intrinsic motivation can be exploited through nichesourcing. Existing Web communities set up by members of the Diaspora (e.g., on Facebook) can be addressed. The network connections can be used to distribute human computation tasks as well as reinforce motivation through reputation. Furthermore, the domain knowledge of the niche members (including the local language and names of villages) may guarantee to produce a higher level of quality than which could be obtained by a general crowd.

4 Nichesourcing Challenges

To achieve a systematic, sustainable and efficient nichesourcing process, an institution needs to (1) employ mechanisms to actively engage and support the experts in the task; and (2) define quality measures for both individual results and the overall production.

Task Distribution. Finding the appropriate niche containing people that are most suited to perform complex tasks is not straightforward. This niche identification and matching challenge requires concise descriptions of the task itself as well as descriptions of the type and level of expertise of the niche required for the task. Additionally, the individual tasks that need be performed might require specific types of domain knowledge. For example, in the Rijksmuseum use case the annotation of one print might require knowledge about specific types of castles while others might require knowledge about historic political issues. The research challenge here is to match tasks with the most appropriate experts within a niche. We can benefit from the extensive research done in the field of *expert finding* to automate this process [15]. In existing communitiesocial connections can be exploited to (re-)distribute tasks among niche members.

Quality Assurance. Although in crowdsourcing reputation plays a role in quality control, for complex tasks in nichesourcing the reputation within the social network of contributing peers is key. *Trust* emerges from an history of collaboration on other tasks, and may be based on similarly evolving interests of the peers. These parameters are

[1] Samples can be viewed at `http://www.few.vu.nl/~vbr240/pluvialdata/`

difficult to measure because this data is distributed across different platforms peers use to work and collaborate.

Even though expert contributions can be expected to be of a higher quality than those produced by a crowd, it is most likely necessary to be able to identify the quality of the individual contributions of members. Where complex tasks are performed within peer networks, quality measurements should explore the social roles, responsibilities and interactions that led to the overall result. When a task is published, there are certain product quality expectations attached to its description. As discussed in Sect. 2, for complex tasks this is not straightforward. However, a poor description may impede the receptivity by the communities. To this end, we could learn from service level agreements which provide frameworks for. e.g., outsourcing, to define and agree on complicated considerations of expected quality of a results.

References

1. von Ahn, L.: Human Computation. PhD thesis, School of Computer Science. Carnegie Mellon University (2005)
2. Quinn, A.J., Bederson, B.B.: Human computation: a survey and taxonomy of a growing field. In: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, CHI 2011, pp. 1403–1412. ACM, New York (2011)
3. Howe, J.: The rise of crowdsourcing. Wired Magazine 14(6) (June 2006)
4. Hand, E.: Citizen science: People power. Nature (2010)
5. Huber, A.J., Lenz, G.A., Michael, G.S., Alvarez, R.E.: Evaluating online labor markets for experimental research: Amazon.com's mechanical turk. Political Analysis (2012)
6. Raddick, J., Szalay, A.S., Vandenberg, J., Bracey, G., Gay, P.L., Lintott, C.J., Murray, P., Schawinski, K.: Galaxy zoo: Exploring the motivations of citizen science volunteers. Astronomy Education Review 9(1) (2010)
7. Oomen, J., Aroyo, L.: Crowdsourcing in the cultural heritage domain: opportunities and challenges. In: Proceedings of the 5th International Conference on Communities and Technologies, C&T 2011, pp. 138–149. ACM, New York (2011)
8. Oded Nov, O.A., Anderson, D.: Dusting for science: motivation and participation of digital citizen science volunteers. In: Proceedings of the 2011 iConference, iConference 2011 (2011)
9. Ipeirotis, P.G., Provost, F., Wang, J.: Quality management on amazon mechanical turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP 2010, pp. 64–67. ACM, New York (2010)
10. Schroer, J., Hertel, G.: Voluntary engagement in an open web based encyclopedia: Wikipedians, and why they do it. Media Psychology 12, 1–25 (2009)
11. Wenger, E., Richard, McDermott, W.S.: Cultivating communities of practice: a guide to managing knowledge. Harvard Business School Press, Cambridge (2002)
12. Kalfatovic, M., Kapsalis, E., Spiess, K., Van Camp, A., Edson, M.: Smithsonian team flickr: a library, archives, and museums collaboration in web 2.0 space. Archival Science 8, 267–277 (2008), doi: 10.1007/s10502-009-9089-y
13. Gligorov, R., Hildebrand, M., van Ossenbruggen, J., Schreiber, G., Aroyo, L.: On the role of user-generated metadata in audio visual collections. In: K-CAP, pp. 145–152 (2011)
14. Holley, R.: Many Hands Make Light Work: Public Collaborative OCR Text Correction in Australian Historic Newspapers. National Library of Australia (2009)
15. Balog, K., Azzopardi, L., de Rijke, M.: Formal models for expert finding in enterprise corpora. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2006, pp. 43–50. ACM, New York (2006)

Dimensions of Argumentation in Social Media

Jodi Schneider[1], Brian Davis[1], and Adam Wyner[2]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
firstname.lastname@deri.org
[2] Department of Computer Science, University of Liverpool
A.Z.Wyner@liverpool.ac.uk

Abstract. Mining social media for opinions is important to governments and businesses. Current approaches focus on sentiment and opinion detection. Yet, people also justify their views, giving arguments. Understanding arguments in social media would yield richer knowledge about the views of individuals and collectives. Extracting arguments from social media is difficult. Messages appear to lack indicators for argument, document structure, or inter-document relationships. In social media, lexical variety, alternative spellings, multiple languages, and alternative punctuation are common. Social media also encompasses numerous genres. These aspects can confound the extraction of well-formed knowledge bases of argument. We chart out the various aspects in order to isolate them for further analysis and processing.

1 Introduction

In social media, people continually express their opinions. These opinions are used to help businesses understand their customers and for governments to understand citizen needs and desires. 80% of data on the Web and on internal corporate intranets is unstructured, hence analysing and structuring the data is a large and growing endeavour[1]. In our view, an important way in which the data can be analysed and further structured is in terms of argumentation. However, we first have to understand the dimensions of expression of argument, which can then be isolated for further analysis and processing. Besides driving the input to knowledge bases, argumentation can also be used for the output of knowledge bases, providing justification and explanation.

Consistency in knowledge bases is essential since we cannot draw informative inferences with inconsistent knowledge bases. In social media, it is obvious that there is lots of disputed information concerning matters of fact (what is or is not true) and of opinion (what individuals believe or prefer). To make use of the knowledge in social media and reason with it, we must treat inconsistency. While a knowledge base may be filtered or truncated based on heuristics, some inconsistencies may remain, whether explicitly or implicitly. Alternatively, users may resolve inconsistencies based on limited weighting information such as provenance or a preference ranking. But to decide which fact is correct or which opinion is most relevant to them, consumers need to go beyond such rankings and to understand how statements are justified and the sources of disagreement. For this, we believe argumentation theory is crucial.

[1] http://www.gartner.com/it/page.jsp?id=1454221

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 21–25, 2012.

Current approaches to extracting and retrieving information from social media use opinion summarisation (e.g. summing votes for or against), topic-based [8] and feature-based text summarisation [7], and visualisation [4]. Such approaches discover trends, relationships, and correlations in data. While they may record inconsistency, they do not provide the means to articulate an elaborate structure of justification and disagreement.

While social media records arguments, current information extraction and knowledge acquisition systems do not represent these arguments, hence people must assimilate and use them unaided. One approach in the direction of representing argument is stance detection [9], which concerns identifying which side a party is taking in a debate, and which responses are rebuttals. While this adds further substance, it does not enable identifying the structure and layers of rationales for and against a position.

Even though current approaches are highly useful in decision making, the whole chain of rationale may be crucial. The overall popularity of an opinion is not as important as the reasons supporting it: overwhelming numbers of people buying a product may not matter as much as a particular reason for *not* buying it. The issue is whether it is the right product for the buyer, which is a matter not only of the pros and cons, but also of the explanations and counterarguments given. In our view, current approaches detect problems, but obscure the chains of reasoning about them.

The challenge is to extract the arguments from the text, turning textual sources into a representation that we can reason with even in the face of inconsistency. We explore these issues as follows. In Section 2, we first introduce the goals of argumentation extraction and provide a sample problem. In Section 3, we outline formalisations of argumentation that enable reasoning with inconsistent data. However, we note the gap between the formalisation and the argument analysis and extraction from source material. This highlights the need for greater understanding of the dimensions of argumentation in the social media landscape, which we discuss in Section 4. In closing, we outline the next steps to bridge between textual sources and the target formal analysis.

2 Goals and Example

Our goal is to extract and reconstruct argumentation into formal representations which can be entered into a knowledge base. Drawing from existing approaches to subjectivity, topic identification, and knowledge extraction, we need to indicate disagreements and other relationships between opinions, along with justifications for opinions. This is currently done by hand. The goal really is to figure out how to automate the analysis. Issues include the informality of language in social media, the amount of implicit information, and various ‘meta’ information that contributes to the argument reconstruction, as we later discuss.

Consider the situation where a consumer wants to buy a camera. In reviews, there may be a high degree of negative sentiment related to the battery, which a consumer can use to decide whether or not she wants to buy the camera. Yet, in the comments to a discussion, we may find statements about whether or not this is in fact true, whether it outbalances other features of the camera, whether the problem can be overcome, and so on. It is not enough to say “you shouldn’t buy this camera” – one needs to give the reasons why. Then the debate becomes an argument about the justifications: “it’s lightweight, you should buy it”, “the lens sucks, you shouldn’t buy it”, “the lens

doesn't matter, it has a bad battery" and so on. The argument is not just point and counterpoint; it is also about how each premise is itself supported and attacked. Each of these justifications may be further discussed, until the discussion 'grounds out' with no further messages. This has the structure of an argument, where points and counterpoints are presented, each implied by premises, which themselves can be argued about further.

Thus we envision deepening the knowledge bases constructed from social media based on the justifications given for statements. To do so, we need to better understand how disagreements and justifications–which we refer to collectively as argumentation–are expressed in social media. However, we first consider our target formalisation.

3 Formalising Argumentation and Argumentation Schemes

Abstract argumentation frameworks have been well-developed to support reasoning with inconsistent information starting with [6] and much subsequent research ([1], [2], [3]). An abstract argument framework, as introduced by Dung, [6] is a pair $AF = \langle \mathcal{A}, attack \rangle$, where $\mathcal{A}$ is a set of arguments and *attack* a binary relation on $\mathcal{A}$. A variety of semantics are available to evaluate the arguments. For example, where AF $= \langle \{A1, A2, A3, A6, A7\}, \{att(A6, A1), att(A1, A6), att(A7, A2)\} \rangle$, then the preferred extensions are: {A3, A6, A7} and {A2, A3, A7}.

However, Dung's arguments are entirely abstract and the attack relation is stipulated. In other words, it is unclear why one argument attacks another argument, as there is no content to the arguments. In order to instantiate arguments we need argumentation schemes, which are presumptive patterns of reasoning [10].

An instantiated argumentation scheme, such as Position To Know, has a textual form such as: 1. Ms. Peters is in a position to know whether Mr. Jones was at the party. 2. Ms. Peters asserts that Mr. Jones was at the party. 3. Therefore, presumptively, Mr. Jones was at the party. This has a formal representation in a typed logical language with functions from argument objects to predicates. The language formally represents the propositions required of the scheme as well as aspects of defeasible reasoning [12].

While this is an attractive approach to tying textual arguments to abstract argumentation, it relies on abstracting away the context and auxiliary aspects. It is far from clear how an argument such as represented in Section 2 can be transformed into a formal argumentation scheme so that it can be reasoned in an argumentation framework. To make use of the formal analyses and related implemented tools for social media discussions, a range of additional issues must be considered, as we next discuss.

4 Dimensions of Expression

To extract well-formed knowledge bases of argument, we must first chart out the various dimensions of social media, to point the way towards the aspects that argumentation reconstruction will need to consider, so that we later can isolate these aspects.

Social media encompasses numerous **genres**, each with their own conversational styles, which affect what sort of rhetoric and arguments may be made. One key feature is the extent to which a medium is used for broadcasts (e.g. monologues) versus conversations (e.g. dialogues), and in each genre, a prototypical message or messages could be described, but these vary across genres due to social conventions and technical constraints. De Moor and Efimova compared rhetorical and argumentative aspects

of listservs and blogs, identifying features such as the likelihood that messages receive responses, and whether spaces are owned communities or by a single individual, and the timeline for replies [5]. Important message characteristics include the typical and allowable message length (e.g. space limitations on microblogs) and whether messages may be continually refined by a group (such as in StackOverflow).

Metadata associated with a post (such as poster, timestamp, and subject line for listservs) and additional structure (such as pingbacks and links for blogs) can also be used for argumentation. For example, a user's most recent post is generally taken to identify their current view, while relationships between messages can indicate a shared topic, and may be associated with agreement or disagreement.

Users are different, and **properties of users** are factors that contribute not only to substance of the user's comment, but as well to how they react to the comments of others. These include demographic information such as the user's age, gender, location, education, and so on. In a specific domain, additional user expectations or constraints could also be added. Different users are persuaded by different kinds of information. Therefore, to solve peoples' problems, based on knowledge bases, when dealing with inconsistency, understanding the purposes and goals that people have would be useful.

Therefore, the **goals of a particular dialogue** also matter. These have been considered in argumentation theory: Walton & Krabbe have categorized dialogue types based on the initial situation, participant's goal, and the goal of the dialogue [11]. The types they distinguish are inquiry, discovery, information seeking, deliberation, persuasion, negotiation and eristic. These are abstractions–any single conversation moves through various dialogue types. For example, a deliberation may be paused in order to delve into information seeking, then resumed once the needed information has been obtained.

Higher level **context** would also be useful: different amounts of **certainty** are needed for different purposes. Some of that is inherent in a task: Reasoning about what kind of medical treatment to seek for a long-term illness, based on PatientsLikeMe, requires more certainty than deciding what to buy based on product reviews.

Informal language is very typically found in social media. Generic language processing issues, with misspellings and abbreviations, slang, language mixing emoticons, and unusual use of punctuation, must be resolved in order to enable text mining (and subsequently argumentation mining) on informal language. Indirect forms of speech, such as sarcasm, irony, and innuendo, are also common. A step-by-step approach, focusing first on what *can* be handled, is necessary.

Another aspect of the informality is that **much information is left implicit**. Therefore, inferring from context is essential. **Elliptical statements** require us to infer common world knowledge, and connecting to existing knowledge bases will be needed.

We apply **sentiment techniques** to provide candidates for argumentation mining and especially to identify **textual markers of subjectivity and objectivity**. The arguments that are made about or against purported facts have a different form from the arguments that are made about opinions. Arguments about objective statements provide the reasons for believing a purported fact or how certain it is. Subjective arguments might indicate, for instance, which users would benefit from a service or product (those similar to the poster). Another area where subjective arguments may appear is discussions of the trust and credibility about the people making the arguments.

5 Conclusions

There is intense interest in extracting information from social media, and particularly in the views people express, and how they express agreement and disagreement, and justify their views. This motivates us to translate existing approaches for text analysis and argumentation mining into techniques for identifying and structuring arguments from social media [13]. But these tools and resources must first be adapted for differences in social media. Understanding these differences is a critical first step, therefore, we have discussed the dimensions of argumentation in social media. Our purpose has been to make explicit the various challenges, so that we can move towards creating knowledge bases of argumentation. Next, the challenges identified should be transformed into requirements.

Acknowledgements. The first and second authors' work was supported by Science Foundation Ireland under Grant No. SFI/09/CE/I1380 (Líon2). The third author was supported by the FP7-ICT-2009-4 Programme, IMPACT Project, Grant Agreement Number 247228. The views expressed are those of the authors.

References

1. Bench-Capon, T.J.M.: Persuasion in practical argument using value-based argumentation frameworks. Journal of Logic and Computation 13(3), 429–448 (2003)
2. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93, 63–101 (1997)
3. Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence 171(5-6), 286–310 (2007)
4. Chen, C., Ibekwe-Sanjuan, F., Juan, E.S., Weaver, C.: Visual analysis of conflicting opinions. In: Proceedings of IEEE Symposium on Visual Analytics Science and Technology, VAST (2006)
5. de Moor, A., Efimova, L.: An argumentation analysis of weblog conversations. In: The 9th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2004), Rutgers University, pp. 2–3 (2004)
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–357 (1995)
7. Lu, Y., Zhai, C., Sundaresan, N.: Rated aspect summarization of short comments. In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009 (2009)
8. Titov, I., McDonald, R.: Modeling online reviews with multi-grain topic models. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008 (2008)
9. Walker, M.A., Anand, P., Abbott, R., Tree, J.E.F., Martell, C., King, J.: That's your evidence?: Classifying stance in online political debate. Decision Support Sciences (2011)
10. Walton, D.: Argumentation Schemes for Presumptive Reasoning. Erlbaum, N.J. (1996)
11. Walton, D.N.: Commitment in dialogue. State University of New York Press, Albany (1995)
12. Wyner, A., Atkinson, K., Bench-Capon, T.: A functional perspective on argumentation schemes. In: Proceedings of Argumentation in Multi-Agent Systems, ArgMAS 2012 (2012)
13. Wyner, A., Schneider, J., Atkinson, K., Bench-Capon, T.: Semi-automated argumentative analysis of online product reviews. In: Proceedings of the Fourth International Conference on Computational Models of Argument, COMMA 2012 (2012)

Semantic Knowledge Discovery from Heterogeneous Data Sources

Claudia d'Amato[1], Volha Bryl[2], and Luciano Serafini[2]

[1] Department of Computer Science - University of Bari, Italy
claudia.damato@di.uniba.it
[2] Data & Knowledge Management Unit - Fondazione Bruno Kessler, Italy
{bryl,serafini}@fbk.eu

Abstract. Available domain ontologies are increasing over the time. However there is a huge amount of data stored and managed with RDBMS. We propose a method for learning association rules from both sources of knowledge in an integrated way. The extracted patterns can be used for performing: data analysis, knowledge completion, ontology refinement.

1 Introduction

From the introduction of the Semantic Web view [4], many domain ontologies have been developed and stored in open access repositories. However, still huge amounts of data are managed privately with RBMS by industries and organizations. Existing domain ontologies may describe domain aspects that complement data in RDMS. This complementarity could be fruitfully exploited for setting up methods aiming at (semi-)automatizing the ontology refinement and completion tasks as well as for performing data analysis. Specifically, hidden knowledge patterns could be extracted across ontologies and RDBMS. To this aim, an approach for learning *association rules* [1] from hybrid sources of information is proposed. Association rule mining methods are well know in Data Mining [12]. They are generally applied to propositional data representations with the goal of discovering patterns and rules in the data. To the best of our knowledge, there are very few works concerning the extraction of association rules from hybrid sources of information. For better explaining the intuition underlying our proposal, let us consider the following example. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an ontology expressed in Description Logics (DLs) [3], composed of a Tbox $\mathcal{T}$ describing general knowledge on kinships and an Abox $\mathcal{A}$ on the kinships of a group of people.

$$\mathcal{T} = \left\{ \begin{array}{lll} \mathsf{Person} \equiv \mathsf{Man} \sqcup \mathsf{Woman} & \mathsf{Man} \sqsubseteq \neg \mathsf{Woman} & \top \sqsubseteq \forall \mathsf{hasChild}.\mathsf{Person} \\ \exists \mathsf{hasChild}.\top \sqsubseteq \mathsf{Person} & \mathsf{Parent} \equiv \exists \mathsf{hasChild}.\mathsf{Person} & \mathsf{Mother} \equiv \mathsf{Woman} \sqcap \mathsf{Parent} \\ \mathsf{Father} \equiv \mathsf{Man} \sqcap \mathsf{Parent} & \mathsf{Grandparent} \equiv \exists \mathsf{HasChild}.\mathsf{Parent} & \mathsf{Child} \equiv \exists \mathsf{HasChild}^{-}.\top \end{array} \right\}$$

$$\mathcal{A} = \left\{ \begin{array}{llll} \mathsf{Woman}(\mathsf{alice}) & \mathsf{Man}(\mathsf{xavier}) & \mathsf{hasChild}(\mathsf{alice}, \mathsf{claude}) & \mathsf{hasChild}(\mathsf{alice}, \mathsf{daniel}) \\ \mathsf{Man}(\mathsf{bob}) & \mathsf{Woman}(\mathsf{yoana}) & \mathsf{hasChild}(\mathsf{bob}, \mathsf{claude}) & \mathsf{hasChild}(\mathsf{bob}, \mathsf{daniel}) \\ \mathsf{Woman}(\mathsf{claude}) & \mathsf{Woman}(\mathsf{zurina}) & \mathsf{hasChild}(\mathsf{xavier}, \mathsf{zurina}) & \mathsf{hasChild}(\mathsf{yoana}, \mathsf{zurina}) \\ \mathsf{Man}(\mathsf{daniel}) & \mathsf{Woman}(\mathsf{maria}) & \mathsf{hasChild}(\mathsf{daniel}, \mathsf{maria}) & \mathsf{hasChild}(\mathsf{zurina}, \mathsf{maria}) \end{array} \right\}$$

Given an ontology and a DL reasoner, it is possible to derive new knowledge that is not explicitly asserted in $\mathcal{K}$. For instance, in the example above it is possible to derive that alice is a Mother and xavier is a Father. Let $\mathbf{D} \subseteq \text{NAME} \times \text{SURNAME} \times \text{QUALIFICATION} \times \text{SALARY} \times \text{AGE} \times \text{CITY} \times \text{ADDRESS}$ be a job information database (see Tab. 1, for simplicity a single table is used). The link between $\mathcal{K}$ and $\mathbf{D}$ is given by

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 26–31, 2012.

Table 1. The job information database

ID	NAME	SURNAME	QUALIFICATION	SALARY	AGE	CITY	ADDRESS
p001	Alice	Lopez	Housewife	0	60	Bari	Apulia Avenue 10
p002	Robert	Lorusso	Bank-employee	30.000	55	Bari	Apulia Avenue 10
p003	Xavier	Garcia	Policeman	35.000	58	Barcelona	Carrer de Manso 20
p004	Claude	Lorusso	Researcher	30.000	35	Bari	Apulia Avenue 13
p005	Daniel	Lorusso	Post Doc	25.000	28	Madrid	calle de Andalucia 12
p006	Yoana	Lopez	Teacher	34.000	49	Barcelona	Carrer de Manso 20
p007	Zurina	Garcia-Lopez	Ph.D student	20.000	25	Madrid	calle de Andalucia
p008	Maria	Lorusso	Pupil	0	8	Madrid	calle de Andalucia

$\{(\mathsf{alice}, P001), (\mathsf{xavier}, p003), (\mathsf{claude}, p004), (\mathsf{daniel}, p005), (\mathsf{yoana}, p006), (\mathsf{zurina}, p007), (\mathsf{maria}, p008)\}$ where the first element is an individual of $\mathcal{K}$ and the second element is an attribute value of $\mathbf{D}$.

Given a method for analyzing jointly the available knowledge sources, it could be possible to induce more general information such as *Women that earn more money are not mothers*. The knowledge of being Woman and Mother comes from the ontology and the knowledge on the salary comes from $\mathbf{D}$. In the following, the approach for accomplishing such a goal based on learning association rules is illustrated.

2 The Framework

Association rules [1] provide a form of rule patterns for data mining. Let $\mathbf{D}$ be a dataset made by a set of attributes $\{A_1, \ldots, \mathcal{A}_n\}$ with domains $D_i : \ i \in \{1, \ldots, n\}$. Learning association rules from $\mathbf{D}$ consists in finding rules of the form $((A_{i_1} = a) \wedge \cdots \wedge (A_{i_k} = t)) \Rightarrow (A_{i_k+1} = w)$ where $a, \ldots, t, w$ are values in $D_{i_1}, \ldots, D_{i_k}, D_{i_k+1}$. The pattern $(A_{i_1} = a) \wedge (A_{i_2} = b) \wedge \cdots \wedge (A_{i_k} = t)$ is called *itemset*. An association rule has the general form $\theta \Rightarrow \varphi$ where θ and φ are itemset patterns. Given the itemset θ, the *frequency* of θ (*fr*(θ)) is the number of cases in $\mathbf{D}$ that match θ. The frequency of $\theta \wedge \varphi$ (*fr*($\theta \wedge \varphi$)) is called *support*. The *confidence* of a rule $\theta \Rightarrow \varphi$ is the fraction of rows in $\mathbf{D}$ that match φ among those rows that match θ, namely *conf*($\theta \Rightarrow \varphi$) = $fr(\theta \wedge \varphi)/fr(\theta)$. A frequent itemset expresses the variables and the corresponding values that occur reasonably often together in $\mathbf{D}$.

The algorithms for learning association rules typically divide the learning problem into two parts: 1) finding the frequent itemsets w.r.t. a given support threshold; 2) extracting the rules from the frequent itemsets satisfying a given confidence thresholds. The solution to the first subproblem is the most expensive one, hence most of the algorithms concentrate on finding optimized solutions to this problem. The most well known algorithm is APRIORI [1]. It is grounded on the key assumption that a set X of variables can be frequent only if all the subsets of X are frequent. The frequent itemsets are discovered as follows:

APRIORI($\mathbf{D}$:dataset, *sp-tr*: support threshold): L frequent itemsets
$L = \emptyset$; L_1 = {frequent itemsets of length 1}
for (k = 1; $L_k \neq \emptyset$; k++) do
  C_{k+1} = candidates generated by joining L_k with itself
  L_{k+1} = candidates in C_{k+1} with frequency equal or greater than *sp-tr*
  $L = L \cup L_{k+1}$
return L;

As a first step, all frequent sets L_{1_i} (w.r.t. to a support threshold) consisting of one variable are discovered. The candidate sets of two variables are built by joining L_1 with itself. By depurating them of those sets having frequency lower than the fixed threshold, the sets L_{2_i} of frequent itemsets of length 2 are obtained. The process is iterated, incrementing the length of the itemsets at each step, until the set of candidate itemsets is empty. Once the set L of all frequent itemsets is determined, the association rules are extracted as follows: 1) for each $I \in L$, all nonempty subsets S of I are generated; 2) for each S, the rule $S \Rightarrow (I - S)$ is returned **iff** $(fr(I)/fr(S)) \geq$ *min-confidence*, where *min-confidence* is the minimum confidence threshold.

The basic form of APRIORI focuses on propositional representation. There exist several upgrades focusing on different aspects: reduction of computational costs for finding the set of frequent items [9], definition of heuristics for pruning patterns and/or assessing their *interestingness* [9], discovery of association rules from multi-relational settings, i.e. relational and/or distributed databases [6,8,7], DATALOG programs [11,5]. Algorithms focusing on this third aspect usually adopt the following approach: 1) the entity, i.e. the attribute/set of attributes, of primary interest for extracting association rules is determined; 2) a view containing the attributes of interest w.r.t. the primary entity is built. Moving from this approach, a method for building an integrated data source, containing both data of a database $\mathbf{D}$ and of an ontology $\mathcal{K}$, is proposed. Consequently association rules are learnt. The approach is grounded on the assumption that $\mathbf{D}$ and $\mathcal{K}$ share (a subset of) common individuals. This assumption is reasonable in practice. An example is given by the biological domain where research organizations have their own databases that could be complemented with existing domain ontologies. The method for building an integrated source of information involves the following steps:

1. choose the primary entity of interest in $\mathbf{D}$ or $\mathcal{K}$ and set it as the first attribute A_1 in the table $\mathbf{T}$ to be built; A_1 will be the primary key of the table
2. choose (a subset of) the attributes in $\mathbf{D}$ that are of interest for A_1 and set them as additional attributes in $\mathbf{T}$; the corresponding values can be obtained as a result of a SQL query involving the selected attributes and A_1
3. choose (a subset of) concept names $\{C_1, \ldots, C_m\}$ in $\mathcal{K}$ that are of interest for A_1 and set their names as additional attribute names in $\mathbf{T}$
4. for each $C_k \in \{C_1, \ldots, C_m\}$ and for each value a_i of A_1, if $\mathcal{K} \models C_k(a_i)$ then set to 1 the corresponding value of C_k in $\mathbf{T}$, set the value to 0 otherwise
5. choose (a subset of) role names $\{R_1, \ldots, R_t\}$ in $\mathcal{K}$ that are of interest for A_1 and set their names as additional attribute names in $\mathbf{T}$
6. for each $R_l \in \{R_1, \ldots, R_t\}$ and for each value a_i of A_1, if $\exists y \in \mathcal{K}$ *s.t.* $\mathcal{K} \models R_l(a_i, y)$ then set to 1 the value of R_l in $\mathbf{T}$, set the value to 0 otherwise
7. choose (a subset of) the datatype property names $\{T_1, \ldots, T_v\}$ in $\mathcal{K}$ that are of interest for A_1 and set their names as additional attribute names in $\mathbf{T}$
8. for each $T_j \in \{T_1, \ldots, T_v\}$ and for each value a_i of A_1, if $\mathcal{K} \models T_j(a_i, \mathit{dataValue}_j)$ then set to $\mathit{dataValue}_j$ the corresponding value of T_j in $\mathbf{T}$, set 0 otherwise.

The choice of representing the integrated source of information within tables allows us to avoid the migration of large amount of data stored in RDMS in alternative representation models in order to extract association rules and also allows for directly applying state of the art algorithms for learning association associations.

Table 2. The integrated data source

NAME	QUALIFICATION	SALARY	AGE	CITY	HasChild	Woman	Man	Mother	Father	Child
Alice	Housewife	[0,14999]	[55,65]	Bari	1	1	0	1	0	0
Robert	Bank-employee	[25000,34999]	[55,65]	Bari	0	0	0	0	0	0
Xavier	Policeman	[35000,44999]	[55,65]	Barcelona	1	0	1	0	1	0
Claude	Researcher	[25000,34999]	[35,45]	Bari	0	1	0	0	0	1
Daniel	Post Doc	[15000,24999]	[25,34]	Madrid	1	0	1	0	1	1
Yoana	Teacher	[25000,34999]	[46,54]	Barcelona	1	1	0	1	0	0
Zurina	Ph.D student	[15000,24999]	[25,34]	Madrid	1	1	0	1	0	1
Maria	Pupil	[0,14999]	[0,16]	Madrid	0	1	0	0	0	1

In the following, the proposed method is applied to the example presented in Sect. 1. Let NAME be the primary entity and let QUALIFICATION, SALARY, AGE, CITY be the selected attributes from $\mathbf{D}$. Let Woman, Man, Mother, Father, Child be the selected concept names from $\mathcal{K}$ and let HasChild be the selected role name. The attribute values in the table are obtained as described above. Numeric attributes are pre-processed (as usual in data mining) for performing data discretization [12] namely for transforming numerical values in corresponding range of values. The final resulting table is shown in Tab. 2. Once the integrated data source has been obtained, the APRIORI algorithm is applied to discover the set of frequent items, hence the association rules are lernt. By applying[1] APRIORI to Tab. 2, given a support threshold *sp-tr* $= 0.2$ (namely 20% of the tuples in the table) and a confidence threshold 0.7, some association rules learnt are:

1. SALARY=$[15000, 24999] \Rightarrow (\mathsf{HasChild} = 1) \wedge (\mathsf{Child} = 1)$ (100%)
2. $(\mathsf{Woman} = 1) \Rightarrow (\mathsf{Man} = 0)$ (100%)
3. (AGE=$[25, 34]$) $\wedge$ (CITY =Madrid) $\Rightarrow (\mathsf{HasChild} = 1)$ (100%)
4. $(\mathsf{HasChild} = 1) \wedge (\mathsf{Man} = 1) \Rightarrow (\mathsf{Father} = 1)$ (100%)

The first rule means that if someone earns between 15000 and 24999 euro, he/she has a 100% confidence of having a child and being a Child. The third rule means that if someone is between 25 and 34 years old and lives in Madrid, he/she has a 100% confidence of having a child. The other two rules can be interpreted similarly. Because of the very few tuples in Tab. 2 and quite high confidence threshold, only rules with the maximum confidence value are returned. By decreasing the confidence threshold, i.e. to 0.6, additional rules can be learnt such as (CITY =Madrid) $\Rightarrow$ $(\mathsf{Parent} = 1) \wedge (\mathsf{HasChild} = 1) \wedge (\mathsf{Child} = 1)$ (66%). The method for learning association rules exploits the evidence of the data. Hence it is not suitable for small datasets.

Association rules extracted from hybrid data sources can be used for performing data analysis. For example rule (3) suggests the average age of being a parent in Madrid that could be different in other geographical areas, e.g. Bari. These rules can be also exploited for data completion both in $\mathcal{K}$ and $\mathbf{D}$. For instance, some individuals can be asserted to be an instance of the concept Child in $\mathcal{K}$. Also rules (2), (4) that could seem trivial since they encode knowledge already modeled in $\mathcal{K}$, can be useful for the ontology refinement taks. Indeed, rules come up from the assertional data. Hence, it is possible to discover intentional axioms that have not been modeled in the ontology. If

[1] The Weka toolkit could be easily used for the purpose http://www.cs.waikato.ac.nz/ml/weka/

in the TBox in the example there was no disjointness axiom for Man and Woman but the data in the ABox extensively contained such information (that is our case), rule (2) mainly suggests a disjointness axiom. Similarly for (4).

3 Discussion

The main goal of this work is to show the potential of the proposed approach. Several improvements can be done. In building the integrated data source, concepts and roles are treated as boolean attributes thus adopting an implicit *Closed World Assumption*. To cope with the *Open Wolrd* semantics of DLs, three valued attributes could be considered for treating explicitly unknown information. Concepts and roles are managed without considering inclusion relationships among them. The treatment of this information could save computational costs and avoid the extraction of redundant association rules. Explicitly treating individuals that are fillers of the considered roles could be also of interest. It could be also useful to consider the case when an individual of interest is a filler in the role assertion. Currently these aspects are not managed. An additional improvement is applying the algorithm for learning association rules directly on a relational representation, without building an intermediate propositional representation.

To the best of our knowledge there are very few works concerning the extraction of association rules from hybrid sources of information. The one most close to ours is [10], where a hybrid source of information is considered: an ontology and a constrained DATALOG program. Association rules at different granularity levels (w.r.t. the ontology) are extracted, given a query involving both the ontology and the DATALOG program. In our framework, no query is specified. A collection of data is built and all possible patterns are learnt. Some restrictions are required in [10], i.e. the set of DATALOG predicate symbols has to be disjoint from the set of concept and role symbols in the ontology. In our case no restrictions are put. Additionally, [10] assumes that the alphabet of constants in the DATALOG program coincides with the alphabet of the individuals in the ontology. In our case a partial overlap of the constants would be sufficient.

4 Conclusions

A framework for learning association rules from hybrid sources of information has been presented. Besides discussing the potential of the proposed method, its current limits have been analyzed and the wide spectrum of lines of research have been illustrated. For the future we want to investigate on: 1) the integration of the learnt association rules in the deductive reasoning procedure; 2) alternative models for representing the integrated source of information.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: SIGMOD Conference, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. of the Int. Conf. on Very Large Data Bases, VLDB 1994 (1994)

3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press (2003)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scient. Amer. (2001)
5. Dehaspe, L., Toivonen, H.: Discovery of frequent DATALOG patterns. Journal of Data Minining and Knowledge Discovery 3(1), 7–36 (1999)
6. Džeroski, S.: Multi-Relational Data Mining: an Introduction. SIGKDD Explor. Newsl. 5(1), 1–16 (2003)
7. Goethals, B., Le Page, W., Mampaey, M.: Mining Interesting Sets and Rules in Relational Databases. In: Proc. of the ACM Symp. on Applied Computing (2010)
8. Gu, et al.: MrCAR: A Multi-relational Classification Algorithm based on Association Rules. In: Proc. of WISM 2009 Int. Conf. (2009)
9. Hand, D., Mannila, H., Smyth, P.: Principles of data mining. Adaptive Computation and Machine Learning Series, ch. 13. MIT Press (2001)
10. Lisi, F.A.: AL-QuIn: An Onto-Relational Learning System for Semantic Web Mining. In: Int. J. of Sem. Web and Inf. Systems. IGI Global (2011)
11. Wang, S.-L., Hong, T.-P., Tsai, Y.-C., Kao, H.-Y.: Multi-table association rules hiding. In: Proc. of the IEEE Int. Conf. on Intelligent Syst. Design and Applications (2010)
12. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann (2011)

Automatic Subject Metadata Generation for Scientific Documents Using Wikipedia and Genetic Algorithms

Arash Joorabchi and Abdulhussain E. Mahdi

Department of Electronic and Computer Engineering, University of Limerick, Ireland
{Arash.Joorabchi,Hussain.Mahdi}@ul.ie

Abstract. Topical annotation of documents with keyphrases is a proven method for revealing the subject of scientific and research documents. However, scientific documents that are manually annotated with keyphrases are in the minority. This paper describes a machine learning-based automatic keyphrase annotation method for scientific documents, which utilizes Wikipedia as a thesaurus for candidate selection from documents' content and deploys genetic algorithms to learn a model for ranking and filtering the most probable keyphrases. Reported experimental results show that the performance of our method, evaluated in terms of inter-consistency with human annotators, is on a par with that achieved by humans and outperforms rival supervised methods.

Keywords: text mining, scientific digital libraries, subject metadata, keyphrase annotation, keyphrase indexing, Wikipedia, genetic algorithms.

1 Introduction

Automatic keyphrase annotation methods for scientific documents can be divided into two main categories:

1. Keyphrase Extraction: keyphrases are picked from a set of candidate phrases extracted from the content of the document itself and are ranked and filtered based on their various statistical and/or semantical features, such as frequency, position, length, and coherence. The ranking function could be either (a) unsupervised, where it is heuristically defined by manual analysis of sample documents and encoding general properties of typical keyphrases, e.g., see [1, 2]; or (b) supervised, where it is automatically derived by a general-purpose ML algorithm from a training dataset, e.g., see [3-6]. Keyphrase extraction approach has two main weaknesses: 1) it is prone to generating phrases composed of a set or sequence of words that occur contiguously within the document and have statistically significant properties, such as high frequency (a.k.a statistically motivated phrases), but are ill-formed, grammatically wrong, or meaningless; 2) it limits the scope of potential candidates to the phrases explicitly appearing in the document.

2. Keyphrase Assignment: keyphrases are picked from controlled vocabularies, such as taxonomies, thesauri, and subject heading systems (e.g., LCSH, MeSH, AGROVOC, Eurovoc) and are not confined to the phrases appearing in the document.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 32–41, 2012.

In this approach, keyphrase annotation is treated as a multi-label text classification problem and general-purpose ML algorithms (e.g., SVM, NB) are utilized to learn a model for each term in the controlled vocabulary from a set of manually annotated training documents. The learnt models are then applied to test documents for classification resulting in a set of high-probability classes (i.e., keyphrases) per document, e.g., see [7, 8]. Using this approach, assigned keyphrases are well formed, grammatically correct, and not limited to those appearing in the document. Therefore, it can cope with cases where a concept is discussed but not explicitly mentioned in the document. However, depending on the characteristics of the target domain, this approach may suffer one or more drawbacks common among supervised ML-based approaches to information retrieval in general, including lack of high quality and/or quantity training data, data sparsity and/or skewed distribution, and concept drift.

Medelyan and Witten [9, 10] proposed a hybrid approach as an intermediate between keyphrase extraction and keyphrase assignment which they have called keyphrase indexing. In this approach, candidate phrases are limited to a set of descriptors, i.e., preferred and commonly used terms for the represented concepts in a domain-specific thesaurus, which either themselves or their synonyms/alternative lexical forms (a.k.a non-descriptors, encoded in form of semantic relations in the thesaurus) occur in the document. This method of candidate generation eliminates the two above-mentioned weaknesses of keyphrase extraction approach as the generated candidate phrases are well-formed, semantically rich, and not restricted to those occurring in the document explicitly. Similar to keyphrase extraction, in this approach an unsupervised or supervised ranking function is deployed to model the general properties of keyphrases in order to rank and filter the most probable ones. This method of rank and filtering requires either no or limited training data, depending on the type of ranking function deployed. This is in contrast to keyphrase assignment approach which requires a set of annotated documents per descriptor. The main weakness of the keyphrase indexing approach is that it assumes there exists a comprehensive domain-specific thesaurus for the target domain, which is not always a feasible assumption. This weak point has been addressed by automatic construction of a universal thesaurus from Wikipedia [11, 12] and replacing the domain-specific thesauri with the thesaurus derived from Wikipedia [13, 14].

In this work, we aim to extend the keyphrase indexing approach, described above, by: (a) introducing a new set of features for the candidate phrases derived from Wikipedia, which enhances the performance of rank and filtering process, and (b) introducing a new supervised ranking function based on Genetic Algorithms (GA) which eliminates the need for manual feature selection and outperforms general-purpose ML algorithms used for keyphrase annotation.

2 Candidate Generation

Following the work of Medelyan and Witten [13, 14], we utilize an open-source toolkit called Wikipedia-Miner [15] for candidate generation. We use the topic detection functionality of the Wikipedia-Miner to extract all the Wikipedia topics (i.e., Wikipedia articles) whose descriptor or non-descriptor lexical representations occur in the document, and use the descriptors of the extracted topics as candidate phrases for the document. We have devised a set of twenty statistical, positional, and semantical

features for candidate topics/phrases to capture and reflect various properties of those candidates which have the highest keyphraseness probability:

1. Term Frequency (TF): the occurrence frequency of the candidate phrase (i.e., descriptor of the extracted Wikipedia topic) and its synonyms and alternative lexical forms/near-synonyms (i.e., non-descriptors of the extracted Wikipedia topic) in the document. The TF values are normalized by dividing them by the highest TF value in the document.
2. First Occurrence: the distance between the start of the document and the first occurrence of the candidate topic, measured in terms of the number of characters and normalized by the length of the document.
3. Last Occurrence: the distance between the end of the document and the last occurrence of the candidate topic, measured in terms of the number of characters and normalized by the length of the document.
4. Occurrence Spread: the distance between the first and last occurrences of the candidate topic, measured in terms of the number of characters and normalized by the length of the document. This feature reflects the observation that candidates which are more evenly spread within the document have a higher keyphraseness probability.
5. Length: the number of words in the candidate phrase, i.e., the descriptor of the candidate topic. This feature reflects the general observation that multi-word phrases have a higher keyphraseness probability as they tend to be more specific and less ambiguous. The keyphrase annotation studies which adopt this feature (e.g., see [10, 13, 14, 16, 17]) compute the length of a candidate phrase by simply counting its number of words or characters. However, our approach is to: (a) split the hyphenated words, (b) count the stopwords as 0.5 and non-stopwords as 1.0, (c) normalize the count value by dividing it by 10.0, (d) eliminate candidates which either have a normalized value greater than 1.0 or those which do not contain any letters (e.g., numbers, numerical dates).
6. Lexical Diversity: the descriptor and non-descriptors of a given topic could appear in a document in various lexical forms. We calculate the lexical diversity by (a) case-folding and stemming all the lexical forms of the candidate topic which appear in the document, using an improved version of Porter stemmer called the English (Porter2) stemming algorithm [18]; (b) counting the number of unique stems minus one, so that the lexical diversity value would be zero if there is only one unique stem. Lexical diversity values are normalized by dividing them by the highest possible lexical diversity value between all topics in Wikipedia. As explained in Section 3, this feature is only used in the supervised ranking function to balance and complement the lexical unity feature.
7. Lexical Unity: inverse of lexical diversity calculated as: $1.0 - \textit{lexical diversity}$. Our assumption is that the candidates with higher lexical unity values would have a higher keyphraseness probability.
8. Average Link Probability: the average value of the link probabilities of all the candidate topic's lexical forms which appear in the document. The link probability of a lexical form is the ratio of the number of times it occurs in Wikipedia articles as a hyperlink to the number of times it occurs as plain text.
9. Max Link Probability: the maximum value of all link probabilities of the lexical forms for a candidate topic which appear in the document. Both the average and max

link probability features are based on the assumption that candidate topics whose descriptor and/or non-descriptor lexical forms appearing in the document have a high probability of being used as a hyperlink in Wikipedia articles, also would have a high keyphraseness probability.

10. Average Disambiguation Confidence: in many cases a term from the document corresponds to multiple topics in Wikipedia and hence needs to be disambiguated. For example, the term "Java" could refer to various topics, such as "Java programming language", "Java Island", etc. As described in [19], the Wikipedia-Miner uses a novel ML-based approach for word-sense disambiguation which yields an F-measure of 97%. We have set the disambiguator to perform a strict disambiguation, i.e., each term in the document can only correspond to a single topic which has the highest probabilistic confidence. The value of the *average disambiguation confidence* feature for a candidate topic is calculated by averaging the disambiguation confidence values of its descriptor and non-descriptor lexical forms that appear in the document.

11. Max Disambiguation Confidence: the maximum disambiguation confidence value among the lexical forms of a candidate topic which appear in the document. Both the average and max disambiguation confidence features are incorporated into the ranking function to reduce the likelihood of candidate topics with low disambiguation confidence values being ranked as top keyphrases.

12. Link-Based Relatedness to Other Topics: the Wikipedia-Miner measures the semantic relatedness between topics using a new approach called Wikipedia Link-based Measure (WLM). In this approach the relatedness between two Wikipedia articles/topics is measured according to the number of Wikipedia topics which discuss/mention and have hyperlinks to both the two topics being compared (see [20] for details). The *link-based relatedness to other topics* feature value of a candidate is calculated by measuring and averaging its relatedness to all the other candidates in the document.

13. Link-Based Relatedness to Context: the only difference between this feature and the *link-based relatedness to other topics* is that the relatedness of the candidate topic is only measured against those of other candidate topics in the document which are unambiguous, i.e., their descriptor and non-descriptor lexical forms occurring in the document have only one valid sense. Both the *link-based relatedness to context* and *link-based relatedness to other topics* features are designed to increase the likelihood of those candidate topics with high semantic relevance to other topics in the document being picked as top keyphrases. However, the former only takes into account the unambiguous topics in the document and therefore has high accuracy but low coverage, whereas the latter also includes the ambiguous topics which have been disambiguated based on their surrounding unambiguous context (i.e., unambiguous topics in the document) and therefore has lower accuracy but conclusive coverage.

14. Category-Based Relatedness to Other Topics: Our study shows that as of July 2011, 95% of Wikipedia articles are classified and on average each classified article belongs to 3.82 categories. When a candidate topic is classified, we can utilize its categorization data to measure its semantic relatedness to other candidates in the document. We measure the category-based relatedness of two Wikipedia topics as:

$$\text{Relatedness}(topic_1, topic_2) = 1 - \frac{\text{Distance}(topic_1, topic_2) - 1}{2D - 3}, \quad (1)$$

where D is the maximum depth of the taxonomy, i.e., 16 in case of the Wikipedia dump used in this work. The distance function returns the length of the shortest path between $topic_1$ and $topic_2$ in terms of the number of nodes along the path. The term $2D-3$ gives the longest possible path distance between two topics in the taxonomy, which is used as the normalization factor, i.e., $2 \times 16 - 3 = 29$. The shortest possible distance between two nodes/topics is 1 (in case of siblings) and the longest is $2D-3$. Therefore subtracting one from the outcome of the distance function results in a highest possible relatedness value of 1.0, e.g., $1-(1-1)/(2 \times 16-3)=1.0$, and a lowest possible relatedness value of 0.03, e.g., $1-(29-1)/(2 \times 16-3)=0.03$. Changing the divisor from $2D-3$ to $2D-4$ reduces the lowest possible relatedness value to zero, however we have adopted the former and instead assign a zero value to relatedness when either $topic_1$ or $topic_2$ are amongst the 5% of Wikipedia topics which are not classified. The value for *category-based relatedness to other topics* for each candidate is calculated by measuring and averaging its category-based relatedness to all the other candidates in the document.

15. Generality: the depth of the topic in the taxonomy measured as its distance from the root category in Wikipedia, normalized by dividing it by the maximum possible depth, and inversed by deducting the normalized value from 1.0. It ranges between 0.0 for the topics farthest from the root and unclassified ones, and 1.0 for the root.

16. Speciality: inverse of generality calculated as: $1.0 - generality$. This feature is only used in the supervised ranking function (see Section 3) to balance and complement the generality feature.

17. Distinct Links Count: total number of distinct Wikipedia topics which are linked in/out to/from the candidate topic, normalized by dividing it by the maximum possible distinct links count value in Wikipedia.

18. Links Out Ratio: total number of distinct Wikipedia topics which are linked out from the candidate topic, divided by the *distinct links count* value of the candidate. Our preliminary experiments show that the candidates with a higher ratio of links out to links in, have a higher keyphraseness probability.

19. Links In Ratio: total number of distinct Wikipedia topics which are linked in to the candidate topic divided by the *distinct links count* value of the candidate. This feature is only used in the supervised ranking function (see Section 3) to balance and complement the *links out ratio*.

20. Translations Count: number of languages that the candidate topic is translated to in the Wikipedia, normalized by dividing it by the maximum possible translations count value in Wikipedia.

3 Rank and Filtering

As discussed in Section 1, the function applied to rank all the candidates and filter out those with highest keyphraseness probabilities could be either supervised or unsupervised. Since all the features defined in Section 2 are normalized to range from 0.0 to 1.0, a simple unsupervised function could be defined as:

$$\text{Score}(topic_j) = \sum_{i=1}^{|F|} f_{ij} \tag{2}$$

which computes the sum of all feature values of a given candidate topic, $topic_j$, as its keyphraseness score. The feature set, F, does not contain the inverse features f_6, f_{16}, and f_{19} as they are only designed to be used in the supervised function. The main advantage of this unsupervised approach is that it does not involve a training process and, therefore, does not require any manually annotated documents for learning a rank and filtering function from. Hence, it may be readily applied to document collections across all domains with minimum effort. However, this approach forces a number of naive assumptions on the general properties of keyphrases in research documents, which negatively impact the accuracy performance of the rank and filtering function:

- All the summed features carry the same weight in respect to their capacity for measuring the keyphraseness probability of a candidate. This is a virtually impossible assumption as shown previously (e.g., see [14]).
- All the features correspond and contribute to the keyphraseness probability of candidates linearly. This assumption does not hold neither intuitively nor empirically [14]. For example, in case of positional features such as first occurrence (f_2) and last occurrence (f_3), only extreme values (>~0.9) should have a significant effect on the overall keyphraseness scores of candidates. Therefore, an exponential correspondence between the values of these features and the scores of candidates better captures the behavior of these features.

The above issues may be addressed to a large degree by adding a weight, w_i, and a degree parameter, d_i, to each feature in equation 3:

$$\text{Score}(topic_j) = \sum_{i=1}^{|F|} w_i f_{ij}^{\,d_i} \tag{3}$$

In an unsupervised setting, knowledge engineers would require to heuristically find and assign the (near) optimum values to these two parameters via examination of a collection of manually annotated documents. However, the optimum values for these parameters could change from one domain or dataset to another. Hence, in order to achieve the best performance, the learning process needs to be repeated whenever the underlying nature of the dataset changes. In our supervised approach we automate this learning process by utilizing genetic algorithms to learn the optimum values for the weight and degree parameters from a set of manually annotated documents. In this approach, the learning process consists of the following steps:

1. Generating an initial population consisting of a set of ranking functions with random weight and degree parameter values from within a predefined range.
2. The fitness of each individual ranking function in the population is evaluated via applying it to a set of training documents to rank and filter their most probable keyphrases, and comparing the resulted top n keyphrases per document with those assigned by human annotators.
3. Based on their fitness, a number of individuals in the current population are stochastically selected, crossed, and mutated to form a new population for the next generation.
4. Steps 2 and 3 are repeated successively until one of the following termination conditions is reached:

(a) A ranking function with the highest possible fitness is found, i.e., for all the documents in the training set, all the top *n* keyphrases resulted from applying the ranking function match those assigned by human annotators. In practice we use the inter-indexer consistency measure to evaluate the fitness of individual ranking functions (see Section 4 for details of the evaluation measure used).

(b) The threshold on the number of generations is invoked. We have defined a greedy thresholding scheme which initially allows a predefined number of generations specified by the *threshold* variable to pass, and from that point on it counts the number of generations that pass without any improvement in the fitness of their fittest individual compared to that of the previous generation. Each time there is an improvement, the counter is reset to zero. The iteration process terminates when the number of generations passed without improvement equals half the total number of generations passed. It should be noted that since we use elitism the fitness of the best individual in each generation is guaranteed to be equal or higher than the fitness of the best individual in the previous generation, and the proposed thresholding mechanism would not work effectively without elitism.

5. The best individuals of all the generations built since the last generation with an improvement up to the last one before termination, are stored to be used for rank and filtering high probability keyphrases in unseen documents.

The degree parameters are float numbers with values between 0.0 and 2.0, allowing each feature, f_i, to be scaled logarithmically ($0.0<d_i<1.0$), linearly ($d_i=1.0$), exponentially ($1.0<d_i\leq 2.0$), or to become neutralized ($d_i=0.0$). The weight parameters have the same type and range as the degree parameters, allowing the weight of each feature, i.e., the magnitude of its impact on the total keyphraseness score of a given candidate, to become nil ($w_i=0.0$), a fraction of neutral ($0.0<w_i<1.0$), neutral ($w_i=1.0$), or up to twice bigger than neutral ($1.0<w_i\leq 2.0$). The defined ranges allow the genetic algorithm to render the features that are too noisy or counterproductive redundant via setting their degree to zero, or setting their weight to zero (or close to it). This in effect automates the feature selection process.

In the unsupervised setting, the universal ranking function defined in Equation 2 is applied to unseen documents and the top *n* keyphrases with the highest keyphraseness probability are filtered out. In the supervised setting however, the data stored at the fifth step of the learning process is used to apply an individual or a set of ranking functions defined in Equation 3, whose weight and degree parameters are adjusted according to the general properties of keyphrases in the target dataset. We have developed and evaluated two different methods to this:

- **Last best:** in this method, simply the individual function with the highest fitness from the last generation before termination is applied to the documents to rank and filter their top *n* keyphrases.
- **Unique bests:** in this method, the final score of each candidate keyphrase in a given document is calculated as the sum of its scores from all the unique individual

ranking functions created during the learning process, which have yielded the best fitness value (achieved before termination) with different weight and degree value sets. This ensembled scoring method takes into account all the variations of the weight and degree value sets which have yielded the final best fitness value.

4 Experimental Results and Evaluation

For evaluating the performance of our keyphrase annotation method, we have used a dataset called wiki-20 [21] created by Medelyan and Witten [13, 14]. The wiki-20 collection consists of 20 Computer Science (CS) related technical research reports, each manually annotated by fifteen different human teams independently. Each team consisted of two senior undergraduate and/or graduate CS students. The teams were instructed to assign about five keyphrases to each document from a controlled vocabulary of over two million terms which served as article titles (i.e. topic descriptors) in Wikipedia at the time the dataset was compiled. We follow the evaluation approach from [13, 14] and use the inter-indexer consistency formula proposed by Rolling [22] to measure the quality of keyphrases assigned to the test documents by our method via comparing them with those assigned by each team of human annotators:

$$\text{Inter - indexer consistency}(A,B) = \frac{2c}{a+b} \tag{4}$$

where A and B represent the two annotators whose inter-consistency is being measured, a and b are the number of terms assigned by each annotator, and c is the number of terms they have in common. The overall inter-indexer consistency score of an annotator is calculated by first measuring and averaging its inter-consistency with all the other annotators per document, and then averaging the results over all the documents. This measure is also used in the second step of the learning process described in Section 3, to evaluate the fitness of individual ranking functions in a given population.

In order to achieve a thorough evaluation of the performance of our method, we have conducted three rounds of evaluation, each consisting of three sets of experiments: (ExpA) 2-fold cross-validation, (ExpB) 4-fold cross-validation, and (ExpC) 20-fold cross-validation, which corresponds to Leave-One-Out Cross-Validation (LOOCV). The data and results of all the experiments are available for download (http://www.skynet.ie/~arash/zip/KA_Wiki20_WM1.2-R233_ECJ20.zip).

Table 1 presents the results of the three rounds of evaluation. In the first round, we set the threshold to 400 (population-size multiplied by 10) which forces the GA to go through a minimum of 800 generations before terminating the learning process. In the second round, we reduced the threshold value from 400 to 200 to speed up the learning process and measure the effect it has on the quality of results. In the third round however, we doubled the threshold from 400 to 800 to examine if having a larger threshold and lengthier learning process would result in an improved overall performance. The results indicate underfitting in case of the second round and overfitting in case of the third round, both resulting in an underperforming model. Table 2 compares the performance of our machine annotator on the wiki-20 dataset with human annotators, a baseline machine annotator based on TFIDF, two un-supervised machine annotators: the work of Grineva et al. [1], and CKE [2], and two supervised machine annotators: KEA++ (KEA-5.0) [13, 17] and Maui [14].

Table 1. Results of the three rounds of evaluation

Round	Dataset	Train				Test					Settings	
						Supervised			Avg. Un-supervised	Avg. Highest Possible	*nk*	5
		Avg. # of Gens	Avg. Time Mins.	Avg. Fitness	Avg. Highest Possible	Avg. Unique Bests	Avg. # of Unique Bests	Avg. Last Best			Selection method	Tournament
First	ExpA (2-fold CV)	800	20.93	37.1%	48.5%	33.4%	202	32.7%	30.7%	48.5%	*Elites*	1
	ExpB (4-fold CV)	3137	131.03	37.4%	48.5%	32.7%	100	32.8%	30.7%	48.5%	*Population*	40
	ExpC (LOOCV)	1510	75.19	36.7%	48.5%	33.5%	73	33.5%	30.7%	48.5%	Genomes	40
Second	ExpA (2-fold CV)	672	17.07	37.6%	48.5%	32.8%	84	33.5%	30.7%	48.5%	Chunks	2
	ExpB (4-fold CV)	584	24.34	36.8%	48.5%	31.4%	54	31.5%	30.7%	48.5%	Crossover	2 ps
	ExpC (LOOCV)	1000	49.83	36.5%	48.5%	32.9%	53	32.6%	30.7%	48.5%	*w,d*	[0,2]
Third	ExpA (2-fold CV)	2420	62.22	38.3%	48.5%	33.3%	194	33.5%	30.7%	48.5%	Mutation	Reset
	ExpB (4-fold CV)	2487	102.55	37.5%	48.5%	32.9%	218	33.1%	30.7%	48.5%	*Mutation probability*	0.05
	ExpC (LOOCV)	3104	154.38	37.1%	48.5%	32.8%	149	31.8%	30.7%	48.5%		

Table 2. Performance comparison with human annotators and rival machine annotators

Method	Learning Approach	Number of Keyprases Assgined per document, *nk*	Avg. inter consistency with human annotators (%)		
			Min.	Avg.	Max.
TFIDF (baseline)	n/a - unsupervised	5	5.7	8.3	14.7
KEA++ (KEA-5.0)	Naïve Bayes	5	15.5	22.6	27.3
Grineva et al.	n/a - unsupervised	5	18.2	27.3	33.0
Maui	Naïve Bayes (all 14 features)	5	22.6	29.1	33.8
Maui	Bagging decision trees (all 14 features)	5	25.4	30.1	38.0
Human annotators (gold standard)	n/a - senior CS students	Varied, with an average of 5.7 per document	21.4	30.5	37.1
CKE	n/a - unsupervised	5	22.7	30.6	38.3
Current work	**n/a - unsupervised**	**5**	**19.1**	**30.7**	**37.9**
Maui	Bagging decision trees (13 best features)	5	23.6	31.6	37.9
Current work (LOOCV)	GA, threshold=800, unique bests method	5	12.3	32.8	58.1
Current work (LOOCV)	GA, threshold=200, unique bests method	5	13.9	32.9	56.7
Current work (LOOCV)	**GA, threshold=400, unique bests method**	**5**	**14.0**	**33.5**	**58.1**

5 Conclusion

In this paper, we introduced an unsupervised and a supervised GA-based keyphrase annotation method for scientific literature utilizing a large number of features derived from Wikipedia. We evaluated the performance of both methods in terms of the consistency of their resulted keyphrases for a collection of test documents with those assigned by human annotators. The results of the three rounds of evaluation show that both methods outperform their rivals and yield a performance above the average performance of human annotators in terms of overall inter-indexer consistency.

References

1. Grineva, M., Grinev, M., Lizorkin, D.: Extracting key terms from noisy and multi-theme documents. In: 18th International Conference on World Wide Web, Madrid, Spain (2009)
2. Mahdi, A.E., Joorabchi, A.: A Citation-based approach to automatic topical indexing of scientific literature. Journal of Information Science 36, 798–811 (2010)

3. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: practical automatic keyphrase extraction. In: Fourth ACM Conference on Digital Libraries. ACM, Berkeley (1999)
4. Turney, P.D.: Learning Algorithms for Keyphrase Extraction. Inf. Retr. 2, 303–336 (2000)
5. Turney, P.D.: Coherent keyphrase extraction via web mining. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, Mexico, pp. 434–439 (2003)
6. Nguyen, T.D., Kan, M.-Y.: Keyphrase extraction in scientific publications. In: Proceedings of the 10th International Conference on Asian Digital Libraries, Vietnam, pp. 317–326 (2007)
7. Markó, K.G., Hahn, U., Schulz, S., Daumke, P., Nohama, P.: Interlingual Indexing across Different Languages. In: Computer-Assisted Information Retrieval, RIAO, pp. 82–99 (2004)
8. Pouliquen, B., Steinberger, R., Ignat, C.: Automatic annotation of multilingual text collections with a conceptual thesaurus. Ontologies and Information Extraction. In: Workshop at EUROLAN 2003 (2003)
9. Medelyan, O., Witten, I.H.: Thesaurus based automatic keyphrase indexing. In: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, USA, pp. 296–297 (2006)
10. Medelyan, O., Witten, I.H.: Domain-independent automatic keyphrase indexing with small training sets. Journal of the American Society for Information Science and Technology 59, 1026–1040 (2008)
11. Milne, D., Medelyan, O., Witten, I.H.: Mining Domain-Specific Thesauri from Wikipedia: A Case Study. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 442–448. IEEE Computer Society (2006)
12. Medelyan, O., Milne, D., Legg, C., Witten, I.H.: Mining meaning from Wikipedia. Int. J. Hum.-Comput. Stud. 67, 716–754 (2009)
13. Medelyan, O., Witten, I.H., Milne, D.: Topic Indexing with Wikipedia. In: First AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI 2008). AAAI Press, US (2008)
14. Medelyan, O.: Human-competitive automatic topic indexing. Department of Computer Science. PhD thesis. University of Waikato, New Zealand (2009)
15. Milne, D.: An open-source toolkit for mining Wikipedia. In: New Zealand Computer Science Research Student Conference (2009)
16. Turney, P.D.: Learning to Extract Keyphrases from Text. National Research Council. Institute for Information Technology (1999)
17. Barker, K., Cornacchia, N.: Using Noun Phrase Heads to Extract Document Keyphrases. In: Hamilton, H.J. (ed.) Canadian AI 2000. LNCS (LNAI), vol. 1822, pp. 40–52. Springer, Heidelberg (2000)
18. Snowball, `http://snowball.tartarus.org/algorithms/english/stemmer.html`
19. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, USA, pp. 509–518 (2008)
20. Milne, D., Witten, I.H.: An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: First AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI 2008), Chicago, I.L. (2008)
21. Wiki20, `http://maui-indexer.googlecode.com/files/wiki20.tar.gz`
22. Rolling, L.: Indexing consistency, quality and efficiency. Information Processing & Management 17, 69–76 (1981)

Advocatus Diaboli – Exploratory Enrichment of Ontologies with Negative Constraints

Sébastien Ferré[1] and Sebastian Rudolph[2]

[1] IRISA, Université Rennes 1, France
Sebastien.Ferre@irisa.fr
[2] KIT, Karlsruhe, Germany
Sebastian.Rudolph@kit.edu

Abstract. With the persistent deployment of ontological specifications in practice and the increasing size of the deployed ontologies, methodologies for ontology engineering are becoming more and more important. In particular, the specification of negative constraints is often neglected by the human expert, whereas they are crucial for increasing an ontology's deductive potential. We propose a novel, arguably cognitively advantageous methodology for identifying and adding missing negative constraints to an existing ontology. To this end, a domain expert navigates through the space of satisfiable class expressions with the aim of finding absurd ones, which then can be forbidden by adding a respective constraint to the ontology. We give the formal foundations of our approach, provide an implementation, called Possible World Explorer (PEW) and illustrate its usability by describing prototypical navigation paths using the example of the well-known pizza ontology.

1 Introduction

Ontologies – logical descriptions of a domain of interest – are at the core of Semantic Technologies. Expressive ontology languages like OWL allow for very precise specifications of semantic interdependencies between the notions describing a domain of interest. While it has been argued that "a little semantics goes a long way" and lightweight formalisms provide for better scalability properties, it is also widely accepted that expressive formalisms are superior in terms of modeling power and the capability of deriving implicit knowledge, thereby allowing for a more intelligent way of handling information.

From the viewpoint of formal semantics, the axioms of an OWL ontology can be seen as conditions or constraints which a possible world has to satisfy for being in line with what the ontology modeler has specified to be "true" in the considered domain. Thereby, one can distinguish between *positive constraints*, which specify what must be necessarily true, and *negative constraints* declaring what is impossible.[1] It has often been noted that positive constraints – such as

[1] Note that, on this general level, the distinction is conceptual rather than technical: for instance, the positive constraint that all catholic priests must be unmarried can likewise be read as the negative constraint that the existence of a married catholic priest is impossible.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 42–56, 2012.

class memberships, class and role hierarchies, or domain and range restrictions – are more salient and graspable to human beings and are preferably specified by modelers, whereas typical negative constraints – like class or role disjointness – are often neglected. However, negative constraints are crucial for exploiting the full deductive potential of expressive ontological modeling. In particular, they are essential for causing inconsistencies, being a helpful feature for many ontology management tasks, e.g. detecting modeling errors in the course of ontology creation and refinement [6], repairing mappings between two ontologies [9] or revising ontologies interactively [12].

In order to overcome this problem, many automated techniques have been designed to extract negative constraints from ontologies themselves or other sources, such as texts [17,18,10,4]. The majority of these techniques rely on heuristics and machine learning methods whence their results are not entirely reliable and usually need to be inspected manually. Moreover, the mentioned approaches are restricted to disjointness, the simplest form of negative constraints. On another note, in the course of interactive ontology completion strategies based on Formal Concept Analysis [14,1,15], negative constraints are naturally acquired next to positive ones. As a downside, these techniques are rather expensive in terms of user interaction and tend to patronize the expert by forcing her to just answer a prescribed row of questions.

We propose to approach the problem from a different angle by providing more freedom to the domain expert and representing the task of specifying negative constraints in a cognitively apt (and, hopefully, interesting or even playful) way. This is achieved by letting the expert navigate the possibilities left open by the currently specified ontology, using a faceted browsing approach, and discover absurd configurations. In a sense, the modeler explores the "Platonic universe" where everything ontologically possible also exists. This way, configurations which are practically impossible can be identified, turned into negative constraints, and added to the ontology.

The paper is structured as follows. In Section 2, we lay out the basic idea of our methodology. Section 3 provides syntax and semantics of the description logic underlying OWL in a condensed way, while Section 4 introduces further formal notions needed for our approach. Section 5 describes our navigation approach on a technical level and establishes properties which ensure its adequacy. In Section 6, we introduce the Possible World Explorer (PEW), a tool which implements the proposed methodology and in Section 7 we illustrate its usefulness and usability by describing exemplary runs of it. Section 8 concludes and describes avenues for future work. An extended version of this paper including full formal proofs is available as technical report [3].

2 The Advocatus Diaboli Methodology

Here we give a non-technical overview of our envisioned methodology for the specification of negative constraints by exploring possible worlds.

As a starting point, we assume that an OWL ontology has been created by stipulating the used vocabulary and possibly arranging classes and properties in

taxonomies. It is not essential that the ontology contains individuals, our method works equally well for non-populated ontologies. Also, the ontology may or may not already contain axioms beyond taxonomic relationships.

According to the model-theoretic semantics, an ontology can be seen as a set of constraints characterizing possible worlds (the models of the ontology). Adding axioms to an ontology results in strengthening these constraints and thereby "ruling out" models.

Our methodology can be seen as an exploration of the possible worlds admitted by an ontology. Thereby, a domain expert starts to describe an individual of one of these possible worlds by specifying its class memberships and relationships to other individuals. This specification process is supported by an interface in the spirit of faceted browsing, which naturally constrains the specification process in a way that no descriptions can be constructed which would contradict the ontology. In other words, the navigation-like stepwise refinement of the description of a possible individual ensures that an individual matching this description indeed exists in at least one of the models of the ontology. In this sense, the proposed methodology can indeed be seen as an "exploration of possible worlds".

The actual task of the domain expert is now to construct descriptions which are possible according to the ontology but absurd given the experts domain knowledge. That is, the domain expert is supposed to assume the role of the "devils advocate" by actively trying to construct situations which are impossible according to his/her knowledge of the domain, thereby showing that the given ontology is underconstrained. Once such a problematic description has been constructed, it can be converted into an axiom which exactly prohibits this situation. By adding this axiom to the ontology, the just constructed absurd description is made impossible and every model featuring such a situation is excluded from the possible worlds.

From a cognitive viewpoint, the particular twist of this methodology is that it facilitates the specification of negative constraints by staying on a positive, scenario-like level, by exploring what is (logically) possible, pushing the boundaries of what is conceivable, and trying to cross them by constructing "nonsensical" descriptions. Arguably, this is much easier and more intuitive than the task of directly coming up with negative constraints.

3 Preliminaries

Although the proposed methodology is suitable for any sufficiently expressive logical formalism, we focus our consideration on the OWL Web Ontology Language [13] as the currently most prominent expressive ontology language.[2]

The OWL DL version of the current OWL standard is based on the very expressive description logic $\mathcal{SROIQ}$ [8]. For a description of the relationship

[2] Note that RDF and RDFS, though certainly more widespread, do not allow for the specification of negative constraints. In fact, the only way to cause an inconsistency in RDFS – namely via XML-clashes – should be seen as a feature which was introduced accidentally rather than intentionally, cf. [7], Section 3.3.3.

Table 1. Syntax and semantics of role and class constructors in $\mathcal{SROIQ}$. Thereby a denotes an individual name, R an arbitrary role name and S a simple role name. C and D denote class expressions.

Name	Syntax	Semantics
inverse role	R^-	$\{\langle x,y\rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid \langle y,x\rangle \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominals	$\{a\}$	$\{a^\mathcal{I}\}$
univ. restriction	$\forall R.C$	$\{x \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in R^\mathcal{I}$ implies $y \in C^\mathcal{I}\}$
exist. restriction	$\exists R.C$	$\{x \in \Delta^\mathcal{I} \mid$ for some $y \in \Delta^\mathcal{I}$, $\langle x,y\rangle \in R^\mathcal{I}$ and $y \in C^\mathcal{I}\}$
Self class	$\exists S.\mathsf{Self}$	$\{x \in \Delta^\mathcal{I} \mid \langle x,x\rangle \in S^\mathcal{I}\}$
qualified number restriction	$\leqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I}$ and $y \in C^\mathcal{I}\} \leq n\}$
	$\geqslant n\, S.C$	$\{x \in \Delta^\mathcal{I} \mid \#\{y \in \Delta^\mathcal{I} \mid \langle x,y\rangle \in S^\mathcal{I}$ and $y \in C^\mathcal{I}\} \geq n\}$

between OWL and the underlying description logics, the reader is referred to [7] or [16]. In this paper we will use description logic notation for its brevity. Thus, we briefly recap syntax and semantics of the description logic $\mathcal{SROIQ}$, although we will only actively work with a restricted sublanguage of it thereafter.

Let N_I, N_C, and N_R be finite, disjoint sets called *individual names*, *class names* and *role names* respectively,[3] with $N_R = \mathbf{R_s} \uplus \mathbf{R_n}$ called *simple* and *non-simple* roles, respectively. These atomic entities can be used to form complex classes and roles in the usual way (see Table 1). A *$\mathcal{SROIQ}$-knowledge base*[4] is a tuple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $\mathcal{T}$ is a $\mathcal{SROIQ}$-TBox, $\mathcal{R}$ is a regular $\mathcal{SROIQ}$-role hierarchy,[5] and $\mathcal{A}$ is a $\mathcal{SROIQ}$-ABox containing axioms as presented in Table 2. The semantics of $\mathcal{SROIQ}$ is defined via interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ composed of a non-empty set $\Delta^\mathcal{I}$ called the *domain of* $\mathcal{I}$ and a function $\cdot^\mathcal{I}$ mapping individuals to elements of $\Delta^\mathcal{I}$, classes to subsets of $\Delta^\mathcal{I}$ and roles to subsets of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. This mapping is extended to complex roles and classes as displayed in Table 1 and finally used to evaluate axioms (see Table 2). We say $\mathcal{I}$ satisfies a knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ (or $\mathcal{I}$ is a model of KB, written: $\mathcal{I} \models KB$) if it satisfies all axioms of $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$. We say that a knowledge base KB *entails* an axiom α (written $KB \models \alpha$) if all models of KB are models of α. Finally, a knowledge base KB is satisfiable if it has a model and a class C is called satisfiable w.r.t. a knowledge base KB if there is a model $\mathcal{I}$ of KB with $C^\mathcal{I} \neq \emptyset$. We also recap that C is

[3] Finiteness of the vocabulary is required for the further considerations. This does not impose a restriction since the vocabulary is not bounded and can be extended whenever this should be necessary.

[4] We use the terms knowledge base and ontology interchangeably.

[5] We assume the usual regularity assumption for $\mathcal{SROIQ}$, but omit it for space reasons.

Table 2. Syntax and semantics of $\mathcal{SROIQ}$ axioms

Axiom α	$\mathcal{I} \models \alpha$, if	
$R_1 \circ \cdots \circ R_n \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$	RBox $\mathcal{R}$
$\mathsf{Dis}(S,T)$	$S^{\mathcal{I}} \cap T^{\mathcal{I}} = \emptyset$	
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox $\mathcal{T}$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox $\mathcal{A}$
$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$	
$a \doteq b$	$a^{\mathcal{I}} = a^{\mathcal{I}}$	
$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

satisfiable if and only if $KB \cup \{C(a)\}$ is satisfiable where a is a "fresh" individual not occurring in KB. Also, C is unsatisfiable if and only if $KB \models C \sqsubseteq \bot$.

4 Formal Foundations

We now define a subclass of OWL class expressions which we deem particularly intuitive to deal with from a cognitive perspective as they essentially represent (alternatives of) existing structures, while negations are only used at an elementary level.[6] To see that this choice is reasonable, note that humans would normally have no problems with handling the class of non-smokers or childless persons, while classes such as non-(persons having a big dog and a small cat) occur unnatural, contrived and are harder to cognitively deal with.

Definition 1. *Given sets N_C, N_R, N_I of atomic class names, atomic role names and individual names, respectively,* simple class expressions *are class expressions of one of the forms A, $\neg A$ (with $A \in N_C$), $\exists r.\top$, $\neg\exists r.\top$, $\exists r^-.\top$, $\neg\exists r^-.\top$ (for $r \in N_R$), $\{o\}$, $\neg\{o\}$ (for $o \in N_I$).*

Next, the set $\mathcal{CI}$ of cognitively intuitive class expressions *is inductively defined as follows:*

1. *every simple class expression is in $\mathcal{CI}$,*
2. *for $C_1, C_2 \in \mathcal{CI}$, $C_1 \sqcap C_2$ and $C_1 \sqcup C_2$ are in $\mathcal{CI}$,*
3. *for $r \in N_R$ and $C \in \mathcal{CI}$, $\exists r.C$ and $\exists r^-.C$ are in $\mathcal{CI}$.*

The set $\mathcal{CI}[X]$ of pointed $\mathcal{CI}$ class expressions *denotes $\mathcal{CI}$ class expressions with the symbol X occurring exactly once in the place of an unnegated class name.*

In words, $\mathcal{CI}$ class expressions allow for the description of situations: existing objects, their interrelations and their properties (in terms of being (non)-members of atomic classes, (not) participating in a relationship, or (not) being identical to a named individual). Thereby, the structure of the axioms enforces that only

[6] In fact, the navigation paradigm for building such class expression will be such that it even discourages the use of this simple form of negation.

tree-like relationships can be described. Moreover, the use of disjunction allows for specifying alternatives for parts of the situation descriptions.

Pointed class expressions are used to put a *focus* on a subexpression of a class expression. This focus will serve as a marker to indicate a point in the expression where new subexpressions can be attached. Consequently, given a pointed $\mathcal{CI}$ class expression $C(X)$ and a $\mathcal{CI}$ class expression D, we write $C(D)$ for the class expression $C(X)[D/X]$ obtained by replacing the occurrence of X in $C(X)$ by D. The following proposition is an easy consequence of the observation that by construction, X occurs in a position with positive polarity.

Proposition 1. *Let KB be a knowledge base, let D and D' be arbitrary class expressions and let $C(X)$ be a pointed $\mathcal{CI}$ class expression. Then $KB \models D \sqsubseteq D'$ implies $KB \models C(D) \sqsubseteq C(D')$.*

Definition 2. *Given a knowledge base KB and a pointed $\mathcal{CI}$ class expression $C(X)$, we call $C(X)$ satisfiable w.r.t. KB, if $C(\top)$ is satisfiable w.r.t. KB. We further define the* possible adjuncts *of $C(X)$ (denoted by $\mathsf{poss}_{KB}(C(X))$) as all simple class expressions D for which $C(D)$ is satisfiable w.r.t. KB. Moreover, we define the* necessary adjuncts *of $C(X)$ (denoted by $\mathsf{nec}_{KB}(C(X))$) as the set of all simple class expressions D for which $C(\neg D)$ is unsatisfiable w.r.t. KB.*

Example 1. Let KB be a knowledge base containing just the following two axioms: (A1) $\exists \mathtt{colonyOf}^{-}.\top \sqcap \mathtt{EUCountry} \sqsubseteq \bot$ stating that EU countries must not have colonies and the axiom (A2) $\exists \mathtt{colonyOf}^{-}.\top \sqsubseteq \mathtt{Country}$ expressing that only countries may have colonies. Then, considering the pointed class expression $\mathtt{Country} \sqcap \exists \mathtt{colonyOf}.X$ has $\mathtt{Country}$ as a necessary adjunct since (A2) would render $\mathtt{Country} \sqcap \exists \mathtt{colonyOf}.\neg \mathtt{Country}$ unsatisfiable. On the other hand, $\mathtt{EUCountry}$ is *not* a possible adjunct, since $\mathtt{Country} \sqcap \exists \mathtt{colonyOf}.\mathtt{EUCountry}$ is not satisfiable.

Clearly, the sets of possible and necessary adjuncts of a pointed class expression provide useful information on how the expression can be reasonably extended and what extending adjuncts would be implied anyway, both taking the provided knowledge base into account. Still, in specific cases, with disjunctive information being involved, $\mathsf{poss}_{KB}(C(X))$ might not quite capture the needed information, as illustrated by the following example.

Example 2. Considering the knowledge base KB introduced above, the pointed class expression $\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.(\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.X)$ would allow for the class $\mathtt{EUCountry}$ as possible adjunct, as the class $\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.(\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.\mathtt{EUCountry})$ is still satisfiable thanks to either of the disjuncts $\mathtt{EUCitizen}$ and $\mathtt{EUCountry}$.

To exclude such unwanted cases, we introduce the notion of balancedness (Definition 4) as a desired property of class expressions. Intuitively, a class expression is balanced, if all alternatives described by unions can possibly occur. Toward the formal definition, we first have to introduce the notion of prunings (Definition 3). By pruning a pointed class expression, we specialize it by removing

disjunctive side branches, thus enforcing that the disjunctive branch in which X is located must be "realized".

Definition 3. *The pruning of a pointed $\mathcal{CI}$ class expression is obtained by applying the recursive function* prune *(we tacitly exploit commutativity of $\sqcap$ and $\sqcup$ to reduce cases):*

$$\begin{aligned} \mathsf{prune}(X) &:= X \\ \mathsf{prune}(C(X) \sqcap D) &:= \mathsf{prune}(C(X)) \sqcap D \\ \mathsf{prune}(C(X) \sqcup D) &:= \mathsf{prune}(C(X)) \\ \mathsf{prune}(\exists r.C(X)) &:= \exists r.\mathsf{prune}(C(X)) \\ \mathsf{prune}(\exists r^-.C(X)) &:= \exists r^-.\mathsf{prune}(C(X)) \end{aligned}$$

Example 3. Continuing the above example, we obtain

$$\begin{aligned} &\mathsf{prune}(\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.(\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.X)) \\ &\qquad = \exists \mathtt{livesIn}.\exists \mathtt{colonyOf}.X. \end{aligned}$$

Definition 4. *Let KB be a knowledge base and let C be a $\mathcal{CI}$ class expression in which a union $D_1 \sqcup D_2$ occurs as a subexpression. Let $C'(X)$ be obtained from C by replacing this occurrence with X, such that $C = C'(D_1 \sqcup D_2)$. Then, we call the occurrence of $D_1 \sqcup D_2$ in C* balanced *if both $\mathsf{prune}(C'(X))[D_1/X]$ and $\mathsf{prune}(C'(X))[D_2/X]$ are satisfiable w.r.t. KB. Otherwise, we say the occurrence is* imbalanced *and call every D_i with unsatisfiable $\mathsf{prune}(C'(X))[D_i/X]$ a* fake disjunct.

A $\mathcal{CI}$ class expression C is called fully balanced *if it is satisfiable and all occurrences of union subexpressions are balanced. A pointed class expression $C(X)$ is called fully balanced if $C(\top)$ is fully balanced.*

Example 4. $\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.(\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.\mathtt{EUCountry})$ can be found to be not fully balanced since it contains the imbalanced occurrence of $\mathtt{EUCountry} \sqcup \exists \mathtt{colonyOf}.\mathtt{EUCountry}$ with $\exists \mathtt{colonyOf}.\mathtt{EUCountry}$ being the fake disjunct since $\mathsf{prune}(\mathtt{EUCitizen} \sqcup \exists \mathtt{livesIn}.X)[\exists \mathtt{colonyOf}.\mathtt{EUCountry}/X] = (\exists \mathtt{livesIn}.X)[\exists \mathtt{colonyOf}.\mathtt{EUCountry}/X] = \exists \mathtt{livesIn}.\exists \mathtt{colonyOf}.\mathtt{EUCountry}$) is unsatisfiable w.r.t. KB (see above).

By definition, full balancedness of a class can be checked by a twofold class satisfiability test for each disjunctive subexpression, thus the number of necessary class satisfiability checks is linearly bounded by the size of the class expression.

It is rather easy to see that by restricting to fully balanced class expressions, we do not lose anything in terms of expressivity, since for any satisfiable class we find an equivalent one which is fully balanced by pruning away the fake disjuncts.

Proposition 2. *For any not fully balanced $\mathcal{CI}$ class expression C there is a $\mathcal{CI}$ class expression C' such that*

- *C' is fully balanced,*
- *$KB \models C \equiv C'$*
- *C' is obtained by the repeated replacement of imbalanced occurrences of unions by the respective non-fake disjunct.*

Example 5. Given KB from above, we find that the (not fully balanced) class expression $\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.\texttt{EUCountry})$ and the fully balanced class expression $\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry})$ are equivalent w.r.t. KB.

These findings justify our suggestion to restrict the possible adjuncts for a pointed class expression to those which would not just maintain its satisfiability, but also its balancedness.

Definition 5. *Given a knowledge base KB and a pointed $\mathcal{CI}$ class expression $C(X)$, we define the* nice possible adjuncts *of $C(X)$ (denoted by $\mathsf{poss}^{\textcircled{\scriptsize\smile}}_{KB}(C(X))$) as all simple class expressions D for which $C(D)$ is fully balanced w.r.t. KB.*

Example 6. Considering the knowledge base from above, we obtain $\texttt{EUCountry} \notin \mathsf{poss}^{\textcircled{\scriptsize\smile}}_{KB}(\texttt{EUCitizen} \sqcup \exists \texttt{livesIn}.(\texttt{EUCountry} \sqcup \exists \texttt{colonyOf}.X))$ since inserting $\texttt{EUCountry}$ for X would result in a not fully balanced class.

5 Navigation

We now describe the navigation operations of our class exploration methodology on an abstract level as modifications of a pointed class expression $C(X)$.

(M) **Moving the focus.** For moving the focus, one picks an occurrence of a subclass D which is not in the scope of a negation. Then we obtain $C'(X, Y)$ from $C(X)$ by replacing the chosen occurrence of D by Y if $D = \top$ and by $D \sqcap Y$ otherwise. Thereafter, we obtain $C''(Y)$ from $C'(X, Y)$ by replacing $E \sqcap X$ by E if X occurs in such a conjunction, or otherwise replacing X by $\top$. Finally, we obtain the result $C_{\texttt{new}}(X)$ of this operation from $C''(Y)$ by substituting Y with X.

(D) **Deleting subexpression at focus.** This operation is applicable if X occurs in $C(X)$ inside a conjunction $E \sqcap X$. In this case, the result $C_{\texttt{new}}(X)$ is obtained by replacing $E \sqcap X$ by X.

(I) **Inserting a disjunction.** This operation is applicable if X occurs in $C(X)$ inside a conjunction $E \sqcap X$. In this case, the result $C_{\texttt{new}}(X)$ is obtained by replacing $E \sqcap X$ by $E \sqcup X$.

(E) **Extending the expression.** Pick a class expression $D \in \mathsf{poss}^{\textcircled{\scriptsize\smile}}_{KB}(C(X))$ and obtain $C_{\texttt{new}}(X)$ by replacing X with $\exists r^{(-)}.X$ in case $D = \exists r^{(-)}.\top$ or otherwise with $D \sqcap X$.

We now provide two desirable properties, which justify the choice of the navigation steps introduced above. The not overly intricate but in places a bit verbose and tedious full proofs can be found in [3]. First, we show that in the course of navigation, only fully balanced classes can be obtained if one starts from a fully balanced pointed class expression.

Proposition 3. *Each of the described navigation steps* (M), (D), (I)*, and* (E) *results in a fully balanced pointed class expression, if it is applied to a fully balanced pointed class expression.*

The second proposition shows that the proposed navigation methodology is complete in the sense that we can construct all potentially "interesting" class expressions.

Proposition 4. *Each fully balanced pointed $\mathcal{CI}$ class expression can be constructed by a sequence of navigation steps starting from X.*

Summing up, our navigation paradigm is tuned in a way that favors construction of "meaningful" (in terms of satisfiability and balancedness) class descriptions but does not restrict expressivity otherwise.

6 The Possible World Explorer

We have developed a prototype of the Possible World Explorer (PEW[7] for short) that allows for both the exploration of possible worlds, and the assertion of negative axioms to eradicate possible worlds ("pew pew!"). On one hand, PEW is implemented on top of the OWL API[8] for handling ontologies, and on the HermiT reasoning engine [11] for checking the satisfiability of class expressions. On the other hand, PEW reuses the principles and user interface of SEWELIS[9] [2] for the interactive construction and display of class expressions and their possible adjuncts. The sources, executable, and screencasts of the system are available at `http://www.irisa.fr/LIS/softwares/pew/`.

Figure 1 shows a screenshot of PEW's user interface. It is composed of a toolbar (T) at the top, a class box (C) at the top left, an instance box (I) at the bottom left, and an adjunct box (A) on the right. The class box (C) displays the current pointed class expression $C(X)$, where the subexpression at the focus X is highlighted. The known instances of the class expression $C(\top)$ are listed in the instance box (I). The possible adjuncts of $C(X)$ are displayed in the adjunct box (A). Class names are displayed as a tree according to the underlying ontology's class hierarchy, and unqualified existential restrictions are displayed as a tree according to the property hierarchy. For each possible adjunct $D \in \mathsf{poss}^{\odot}_{KB}(C(X))$, both D and $\neg D$ are displayed, except for the necessary $D \in \mathsf{nec}_{KB}(C(X))$, for which only D is displayed but in a larger font. For the concrete syntax of the class expression and adjuncts, both DL notation and Manchester syntax are available. For better readability of complex expressions, we use indentation instead of brackets, and syntax highlighting (foreground color) to distinguish between class, role, and individual names.

From Figure 1, we can conclude a number of things about the pizza ontology [10]. From the class box, we conclude that a pizza *may* have no topping. From the emptiness of the instance box, we conclude that there is no known individual pizza without topping. From the adjunct box, we further conclude that such a

[7] We adopt the Semantic Web practice of flipping letters in acronyms.
[8] `http://owlapi.sourceforge.net/`
[9] `http://www.irisa.fr/LIS/softwares/sewelis/`
[10] `http://www.co-ode.org/ontologies/pizza/pizza.owl`

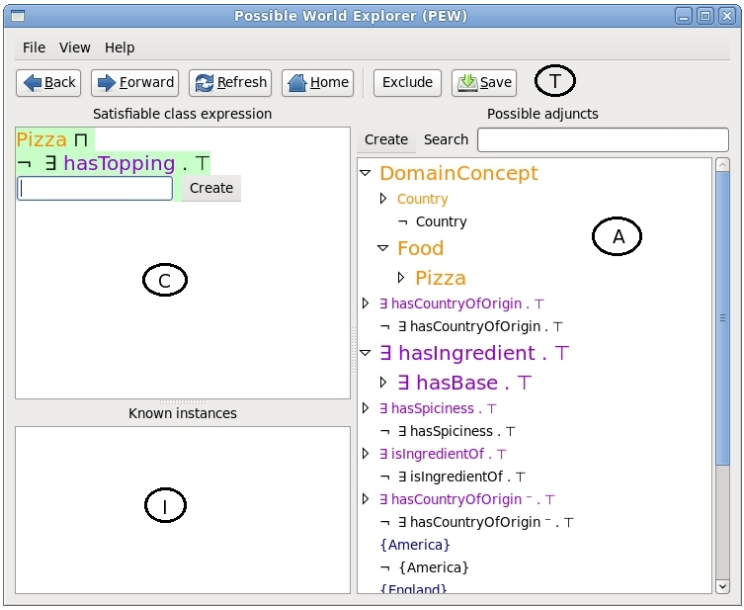


Fig. 1. Screenshot of the Possible World Explorer (PEW) showing that, according to the pizza ontology, a pizza can have no topping

pizza *must* be some food, that it *must* have some base as an ingredient, but that it *may* also be a country, and that it *may* have no country of origin.

Navigation from one (pointed) class expression to another is entirely performed in an interactive way. Double-clicking an adjunct *extends* the class expression by inserting it at the focus. Alternatively, adjuncts can be found by auto-completion in the text fields (one at the focus in the class box, and another above the adjunct box). The focus can be *moved* simply by clicking on various parts of the class expression. The contextual menu of the class box (C) provides the additional navigation steps: *inserting a disjunction*, and *deleting* the subexpression at focus. The toolbar (T) provides navigation in the history of class expressions, as well as the update of the ontology. The button "Exclude" adds the axiom $C(\top) \sqsubseteq \bot$ to the ontology, in order to rule out models in which the current class expression has instances. Figure 1 displays a situation where this operation would make sense. To provide feedback about the update, the focus background color switches from green (satisfiable class) to red (unsatisfiable class). In the case where the update would make the ontology inconsistent, the button "Exclude" triggers an error message. Finally, the button "Save" saves the updated ontology.

The current implementation is rather naive, and optimization is left for future work. For information, we here give the nature and number of performed reasoning tasks (OWL API calls to reasoner methods). First, the hierarchy of simple class expressions has to be computed, which amounts to 1 call to `getInstances` for the top class, 1 call to `getSubClasses` for each named class, and 1 call to

`getSubObjectProperties` for each named role. Then, at each navigation step, the known instances of the class expression $C(X)$ are computed with 1 call to `getInstances`, and the possible adjuncts by checking the possibility and necessity of $C(D)$, for each positive simple class expression D. Checking possibility amounts to $1+2d$ call to `isSatisfiable`, where d is the number of unions in the class expression; and checking necessity amounts to 1 call to `isSatisfiable`.

7 An Example Scenario

In this section, we describe an example scenario of exploration and completion of the pizza ontology. This ontology has the advantage of being well-known, covering a large subset of OWL constructs, and being representative for OWL ontologies. While the pizza ontology is often referred to and was subject to a number of refinements, we found in our exploration a number of unexpected possible worlds, and hence missing axioms. The following scenario describes a non-exhaustive exploration, and illustrates various situations that may arise.

7.1 First Steps in the Ontology

After launching PEW on the pizza ontology, the initial pointed class expression is $C(X) = X$, the instance box displays the list of all named individuals, and the adjunct box displays all simple class expressions. The latter means that every simple class and its complement are satisfiable, which generally holds in ontologies. Without prior knowledge, the user can then discover that the ontology is about food (in particular pizzas, pizza bases and pizza toppings), countries, and spiciness. The possible roles are "has ingredient" (refined into "has base" and "has topping"), "has spiciness", "has country of origin", and their inverses. Only 5 named individuals exist, namely for countries.

7.2 Class Exploration

In order to better understand the possible interactions between classes and properties, the user decides to navigate to each named class to discover what an instance of that class can be. For example, by selecting the adjunct `Country`, the pointed class expression becomes $\texttt{Country} \sqcap X$ (see Figure 2). The instance box says there are 5 known countries, namely America, England, France, Germany, and Italy. Surprisingly, the adjunct box says that a country can be some food (possible adjunct `Food`), or not (possible adjunct $\neg\texttt{Food}$). This implies we can further select the adjunct `Food` to reach the satisfiable class expression $\texttt{Country} \sqcap \texttt{Food} \sqcap X$. Obviously, such an individual should not be possible, and we exclude this possibility by pushing the "Exclude" button, which has the effect of adding the axiom $\texttt{Country} \sqcap \texttt{Food} \sqsubseteq \bot$ to the ontology. This illustrates the claimed fact that even very basic negative constraints such as disjointness axioms are often missing in ontologies. On the contrary, we found no missing positive axiom like subclass axioms.

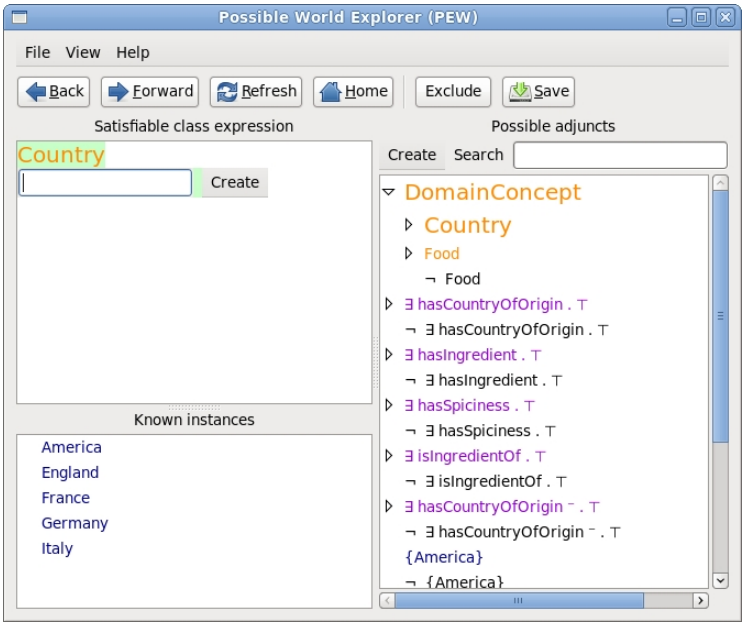


Fig. 2. A screenshot showing, among other things, that a country can be some food, and can have a country of origin

Navigating back to $\texttt{Country} \sqcap X$, the user can verify that a country cannot anymore be an instance of another class. However, looking at possible roles, she discovers that a country can not only be the country of origin of something (adjunct $\exists \texttt{hasCountryOfOrigin}^{-}.\top$), which is fine, but can also have a country of origin (adjunct $\exists \texttt{hasCountryOfOrigin}.\top$), and a spiciness. Those two undesirable possibilities can be ruled out by selecting an unexpected adjunct, and asserting a negative axiom with the "Exclude" button, and by repeating this sequence on the other unexpected adjunct. Note that selecting the two adjuncts simultaneously, and then asserting an axiom would not be equivalent because this would only exclude countries that have both a country of origin *and* a spiciness. Yet, it is possible to use only one axiom provided a class union is used between the two unexpected adjuncts. At this stage, the user can see no more undesirable adjuncts for countries, and move to other named classes.

Looking at food ($C(X) = \texttt{Food} \sqcap X$), the only undesirable adjunct is that some food can be the country of origin of something, which the user excludes. Looking at pizzas, she decides to exclude the possibility for a pizza to be an ingredient. So far, we have only excluded positive possibilities (e.g., a pizza can be an ingredient), but it is also possible to exclude negative possibilities. From the adjunct box, the user discovers that, while a pizza must have some ingredient and some base (the simple class $\neg\exists \texttt{hasBase}.\top$ is not a possible adjunct), it may have no topping (possible adjunct $\neg\exists \texttt{hasTopping}.\top$). This can be excluded simply by selecting the negative adjunct (instead of the positive one), and asserting

the axiom $\texttt{Pizza} \sqcap \neg\exists\texttt{hasTopping}.\top \sqsubseteq \bot$ (see Figure 1), which is equivalent to $\texttt{Pizza} \sqsubseteq \exists\texttt{hasTopping}.\top$ (every pizza has a topping).

Finally, looking at spiciness (degrees), the user excludes the following possibilities: a spiciness that has a country of origin, a spiciness that is the country of origin of something, and a spiciness that has a spiciness (degree). After those exclusions, a spiciness can only be the spiciness of something. The class `Spiciness` has three subclasses: `Hot`, `Medium`, and `Mild`. Selecting any of those classes shows that no other class is possible simultaneously, which means that disjointness axioms have already been asserted between them.

7.3 Exploring Roles

When exploring roles, we investigate for each role what can be at their range and domain. Class exploration has covered axiom schemas $A \sqcap B \sqsubseteq \bot$, $A \sqcap \exists r.\top \sqsubseteq \bot$ and $A \sqcap \neg\exists r.\top \sqsubseteq \top$. Here, we will cover axiom schemas $\exists r.\top \sqcap \neg A \sqsubseteq \bot$ and $\exists r.\neg A \sqsubseteq \bot$, which correspond, respectively, to domain and range axioms.

Looking at things that have a country of origin (with focus on the range, i.e., $C(X) = \exists\texttt{hasCountryOfOrigin}.X$), the user finds that the country of origin may be not a country (adjunct $\neg\texttt{Country}$). This means that the range axiom for role `hasCountryOfOrigin` is missing. It can be added by selecting the undesirable adjunct, and asserting the axiom $\exists\texttt{hasCountryOfOrigin}.\neg\texttt{Country} \sqsubseteq \bot$ which is equivalent to $\top \sqsubseteq \forall\texttt{hasCountryOfOrigin}.\texttt{Country}$.

Inspecting things having a spiciness ($C(X) = \exists\texttt{hasSpiciness}.\top \sqcap X$) with focus at the domain, it appears that those things may not be food (adjunct $\neg\texttt{Food}$). This can be excluded by asserting the axiom $\exists\texttt{hasSpiciness}.\top \sqcap \neg\texttt{Food} \sqsubseteq \bot$, which is equivalent to $\exists\texttt{hasSpiciness}.\top \sqsubseteq \texttt{Food}$, and defines a domain axiom.

7.4 Further Exploration

In this section, we provide an additional example to show that our approach does not only apply to simple interactions between named classes and roles. The class `Pizza` has a number of subclasses that are generally defined through equivalent classes axioms as pizzas satisfying certain criteria. For example, there is the `VegetarianPizza` class, and obviously it should not be possible to find a vegetarian pizza that contains some meat or fish as an ingredient. Following the *advocatus diaboli* approach, this is exactly what we are going to try and find. Starting from a vegetarian pizza ($C(X) = \texttt{VegetarianPizza} \sqcap X$), we find that it may (and must) have some ingredient (adjunct $\exists\texttt{hasIngredient}.\top$). By selecting this adjunct, we reach the class expression $C(X) = \texttt{VegetarianPizza} \sqcap \exists\texttt{hasIngredient}.X$, with the focus on the ingredient. As possible ingredients, we find only food, subdivided into pizza base and pizza topping. Under the class `PizzaTopping`, we find that both subclasses `MeatTopping` and `FishTopping` are possible! It is even possible to reach a vegetarian pizza that has both meat and fish as ingredients. The two undesirable possibilities can be excluded at once by navigating to the satisfiable and balanced class $\texttt{VegetarianPizza} \sqcap \exists\texttt{hasIngredient}.(\texttt{MeatTopping} \sqcup \texttt{FishTopping})$ (see Figure 3), and pushing the

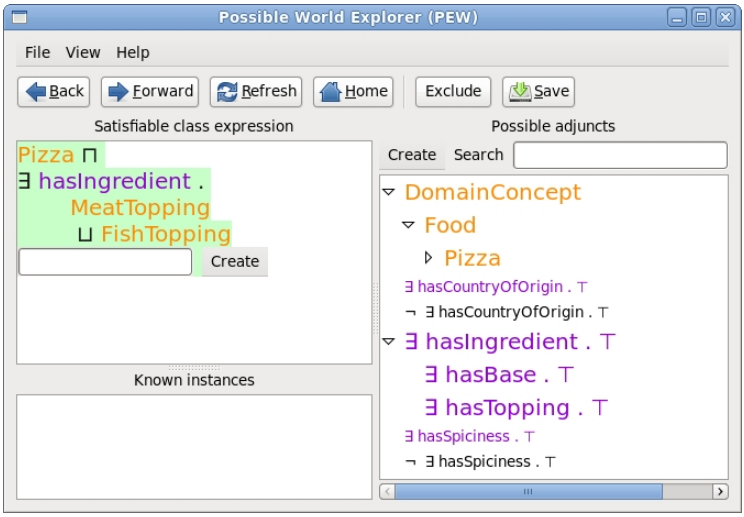


Fig. 3. A screenshot showing that a vegetarian pizza may contain some meat or fish

Exclude button. Looking at the ontology, we do find that a vegetarian pizza is defined as a pizza that contains neither meat nor fish. The problem is that in the respective axiom, the role "has topping" was used instead of the more general "has ingredient". Obviously, a vegetarian pizza should have no meat or fish, no matter what part of the pizza contains it!

8 Conclusion and Future Work

We have proposed an intuitive methodology for adding negative constraints to OWL ontologies in an exploratory way. To this end, we devised and implemented an interaction paradigm for constructing intuitive satisfiable class expressions in an interactive way reminiscent of faceted browsing, which–if found to be absurd–can be turned unsatisfiable by adding a corresponding negative constraint to the underlying ontology.

Future work on this subject clearly includes scalability and usability investigations and improvements. For seamless navigation and editing, the underlying reasoning steps must be performed in near-realtime which poses some restriction on the computational intricacy of the considered ontology. In order to enlarge the scope of applicability of our method, we will optimize PEW in terms of minimizing OWL API calls.

On the usability side, next to thorough user studies, we will further enrich the tool with further functionality beyond mere exclusion of unwanted class expressions. Ultimately, we plan to provide PEW as a Protégé plugin.

Acknowledgement. This work was performed in the course of a research visit of Sebastian Rudolph in Rennes supported by IRISA and University Rennes 1.

References

1. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Veloso, M.M. (ed.) IJCAI, pp. 230–235 (2007)
2. Ferré, S., Hermann, A.: Semantic Search: Reconciling Expressive Querying and Exploratory Search. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 177–192. Springer, Heidelberg (2011)
3. Ferré, S., Rudolph, S.: PEW! PEW! Tech. rep., Institut AIFB, KIT, Karlsruhe (April 2012), http://www.aifb.kit.edu/web/Techreport3024
4. Fleischhacker, D., Völker, J.: Inductive Learning of Disjointness Axioms. 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. 680–697. Springer, Heidelberg (2011)
5. Gómez-Pérez, A., Euzenat, J. (eds.): ESWC 2005. LNCS, vol. 3532. Springer, Heidelberg (2005)
6. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In: Gómez-Pérez and Euzenat [5], pp. 182–197
7. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press (2006)
9. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing ontology mappings. In: AAAI. pp. 1408–1413. AAAI Press (2007)
10. Meilicke, C., Völker, J., Stuckenschmidt, H.: Learning Disjointness for Debugging Mappings between Lightweight Ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 93–108. Springer, Heidelberg (2008)
11. Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. J. of Artificial Intelligence Research 36, 165–228 (2009)
12. Nikitina, N., Rudolph, S., Glimm, B.: Interactive ontology revision. Web Semantics: Science, Services and Agents on the World Wide Web 12-13(0), 118–130 (2012)
13. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-overview/
14. Rudolph, S.: Exploring Relational Structures Via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
15. Rudolph, S.: Acquiring Generalized Domain-Range Restrictions. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 32–45. Springer, Heidelberg (2008)
16. Rudolph, S.: Foundations of Description Logics. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 76–136. Springer, Heidelberg (2011)
17. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Gómez-Pérez and Euzenat [5], pp. 226–240
18. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning Disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 175–189. Springer, Heidelberg (2007)

Universal OWL Axiom Enrichment for Large Knowledge Bases

Lorenz Bühmann and Jens Lehmann

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany
{buehmann,lehmann}@informatik.uni-leipzig.de
http://aksw.org

Abstract. The Semantic Web has seen a rise in the availability and usage of knowledge bases over the past years, in particular in the Linked Open Data initiative. Despite this growth, there is still a lack of knowledge bases that consist of high quality schema information and instance data adhering to this schema. Several knowledge bases only consist of schema information, while others are, to a large extent, a mere collection of facts without a clear structure. The combination of rich schema and instance data would allow powerful reasoning, consistency checking, and improved querying possibilities as well as provide more generic ways to interact with the underlying data. In this article, we present a light-weight method to enrich knowledge bases accessible via SPARQL endpoints with almost all types of OWL 2 axioms. This allows to semi-automatically create schemata, which we evaluate and discuss using DBpedia.

1 Introduction and Motivation

The Semantic Web has recently seen a rise in the availability and usage of knowledge bases, as can be observed within the DataHub[1] and other repositories. Despite this growth, there is still a lack of knowledge bases that consist of sophisticated schema information and instance data adhering to this schema. Several knowledge bases, e.g. in the life sciences, only consist of schema information, while others are, to a large extent, a collection of facts without a clear structure, e.g. information extracted from databases. The combination of sophisticated schema and instance data would allow powerful reasoning, consistency checking, and improved querying. Schema enrichment, as described in this article, allows to create schemata base based on existing data.[2]

Example 1. As an example, consider a knowledge base containing a property `birthPlace` and subjects in triples of this property, e.g. Brad Pitt, Angela Merkel, Albert Einstein etc. Our enrichment algorithms could, then, suggest

[1] http://thedatahub.org/

[2] The approach of not creating schema upfront is sometimes referred to as "grass roots" approach or "after the fact" schema creation.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 57–71, 2012.

that the property `birthPlace` may be functional and has the domain `Person` as it is encoded via the following axioms in Manchester OWL syntax[3]:

```
ObjectProperty: birthPlace
    Characteristics: Functional
    Domain: Person
    Range: Place
    SubPropertyOf: hasBeenAt
```

Adding such axiom to a knowledge base can have several benefits: 1.) The axioms serve as documentation for the purpose and correct usage of schema elements. 2.) They improve the application of schema debugging techniques. For instance, after adding the above axioms the knowledge base would become inconsistent if a person has two different birth places due to the functionality axiom. Specifically for the DBpedia knowledge base, we observed an error in which a person was asserted to be born in Lacrosse, the game, instead of Lacross, the city in the United States. Such errors can be automatically detected when schema information such as the range restriction is present (assuming disjointness of the classes `Place` and `Game`). 3.) Additional implicit information can be inferred, e.g. in the above example the birth place of a person can be inferred to be one of the places a person has stayed at. The main purpose of our research is to reduce the effort of creating and maintaining such schema information.

We implemented our enrichment methods in the DL-Learner[4] framework [15] based on earlier work in [11,22,19] and the ORE tool [17] [5] contains a graphical interface for them. Whereas previously we focused on equivalence and subclass axioms, we describe how to support a broader range of OWL axioms in this article. In particular, we advance the current state of the art as follows:

- support for suggesting the following axioms to enrich a knowledge base:
  - class and property hierarchy (subsumption, equivalence, disjointness)
  - property characteristics (transitivity, (a)symmetry, (inverse)functionality, (ir)reflexivity)
  - inverse properties
- support for knowledge bases accessible via SPARQL endpoints
- scalability of algorithms via sampling
- DL-Learner command line interface and ORE web interface for the algorithms are available as open source

The article is structured as follows: we briefly described the term schema enrichment and give an overview of existing approaches in Section 2. The enrichment approach itself is described in Section 3. To be able to separate the process of generating enrichments from the process of manual supervision by a knowledge engineer, we need to store the suggestions. We do this via an ontology, which is described in Section 4. We then continue by giving some preliminary evaluation results for applying the algorithms on DBpedia in Section 5. Finally, we conclude and describe future work.

[3] For details on Manchester OWL syntax (e.g. used in Protégé, OntoWiki) see `http://www.w3.org/TR/owl2-manchester-syntax/`

[4] `http://dl-learner.org`

[5] `http://ore-tool.net`

2 Knowledge Base Enrichment Overview

The term *enrichment* in this article refers to the extension of a knowledge base schema. It describes the process of increasing the expressiveness and semantic richness of a knowledge base. Enrichment methods can typically be applied in a *grass-roots* approach to knowledge base creation. In such an approach, the whole ontological structure is not created upfront[6], but evolves with the data in a knowledge base. Ideally, this enables a more agile development of knowledge bases, which could become an interesting alternative to more traditional ontology engineering methods.

Knowledge base enrichment can be seen as a sub-discipline of ontology learning. Ontology learning is more general in that it can rely on external sources, e.g. written text, to create an ontology. The term knowledge base enrichment is typically used when already existing data in the knowledge base itself is analysed to improve its schema. Enrichment methods span several research areas like knowledge representation and reasoning, machine learning, statistics, natural language processing, formal concept analysis and game playing. Ontology enrichment usually involves applying heuristics or machine learning techniques to find axioms, which can be added to an existing ontology. Naturally, different techniques have been applied depending on the specific type of axiom.

One of the most complex tasks in ontology enrichment is to find *definitions* of classes. This is strongly related to Inductive Logic Programming (ILP) [26] and more specifically supervised learning in description logics. Research in this area is not purely focused on ontology enrichment, but has other applications, e.g. drug efficiency prediction in the life sciences. Work on learning in description logics goes back to e.g. [6,7], which used so-called *least common subsumers*. Later, [4] invented a refinement operator for $\mathcal{ALER}$ and proposed to solve the problem by using a top-down approach. [8,12,13] combine both techniques and implement them in the YINYANG tool. However, those algorithms tend to produce long and hard-to-understand expressions. The algorithms implemented in DL-Learner [20,21,14,22] overcome this problem and investigate the learning problem and the use of top down refinement in detail. DL-FOIL [9] is a similar approach, which is based on a mixture of upward and downward refinement of class expressions. They use alternative measures in their evaluation, which take the open world assumption into account, which was not done in ILP previously. Most recently, CELOE [16] implements appropriate heuristics and adaptations for learning definitions in ontologies. We use this algorithm for learning definitions, but go beyond it by including support for many different axiom types.

A different approach to learning the definition of a named class is to compute the so called *most specific concept* (msc) for all instances of the class. The most specific concept of an individual is the most specific class expression, such that the individual is instance of the expression. One can then compute the *least common subsumer* (lcs) [3] of those expressions to obtain a description of the named

[6] Talk by Tim Berners-Lee which advocates to get "raw data now": `http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html`

Table 1. Work in ontology enrichment grouped by type or aim of learned structures

Type/Aim	References
Taxonomies	[34,31]
Definitions	ILP approaches: [20,21,22,16,9,8,12,13,4], genetic approaches: [14]
Super Class Axioms	[16,31]
Rules in Ontologies	[23,24]
Disjointness	[33]
Property Chains	[31]
Alignment	challenges: [30], recent survey: [5]
Completion	formal concept analysis and relational exploration [2,32,29]

class. However, in expressive description logics, an msc does not need to exist and the lcs is simply the disjunction of all expressions. For light-weight logics, such as $\mathcal{EL}$, the approach appears to be promising. Other approaches, e.g. [23] focus on learning in hybrid knowledge bases combining ontologies and *rules*. Usually, hybrid approaches are a generalisation of concept learning methods, which enable powerful rules at the cost of efficiency (because of the larger search space). Similar as in knowledge representation, the tradeoff between expressiveness of the target language and efficiency of learning algorithms is a critical choice in symbolic machine learning.

Another enrichment task is *knowledge base completion*. The goal of such a task is to make the knowledge base complete in a particular well-defined sense. For instance, a goal could be to ensure that all subclass relationships between named classes can be inferred. The line of work starting in [28] and further pursued in e.g. [2] investigates the use of *formal concept analysis* for completing knowledge bases. It is promising, although it may not be able to handle noise as well as a machine learning technique. A Protégé plugin [29] is available. [32] proposes to improve knowledge bases through relational exploration and implemented it in the *RELExO framework*[7]. It focuses on simple relationships and the knowledge engineer is asked a series of questions. The knowledge engineer either must positively answer the question or provide a counterexample.

[33] focuses on learning *disjointness* between classes in an ontology to allow for more powerful reasoning and consistency checking. To achieve this, it can use the ontology itself, but also texts, e.g. Wikipedia articles corresponding to a concept. One of the closely related and most recent work in the area is statistical schema induction via *association rule mining* [31]. Association rules are a form of implication patterns and used to discover regularities in a data set. For instance, in [31] an association rule $A \implies B$ with sufficiently high confidence and support between two classes A and B indicates that introducing a subclass relationship $A \sqsubseteq B$ may be appropriate.

Another type of ontology enrichment is schema mapping. This task has been widely studied and will not be discussed in depth here. Instead, we refer to [5]

[7] http://code.google.com/p/relexo/

for a survey on ontology mapping. Schema mapping is not integrated in the presented prototype.

3 Enrichment with OWL Axioms

There is a large variety of axiom types in OWL, which we support in our enrichment tool. We first describe our general methodology for creating enrichment suggestions and then present details for each axiom type in separate sections.

3.1 General Method

In this part, we will describe the light-weight learning methods for obtaining enrichment suggestions. The methods usually take an entity (a class or property in our case) as input, generates a set of OWL axioms as output and proceeds in three phases (see Figure 1):

1. In the first phase, SPARQL queries are used to obtain general information about the knowledge base, in particular we retrieve axioms, which allow to construct the class hierarchy. It can be configured whether to use an OWL reasoner for inferencing over the schema or just taking explicit knowledge into account.[8] Naturally, the schema only needs to be obtained once and can then be re-used by all algorithms and all entities.
2. The second phase consists of obtaining data via SPARQL, which is relevant for learning the considered axiom. We will briefly describe this phase for each axiom type in the following sections.
3. In the third phase, the score of axiom candidates is computed and the results returned.

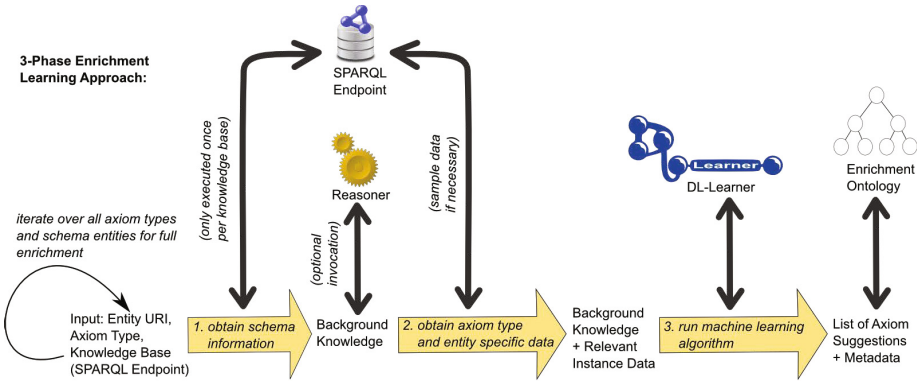


Fig. 1. 3-Phase Enrichment Workflow

[8] Note that the OWL reasoner only loads the schema of the knowledge base and, therefore, this option usually works even in cases with several hundred thousand classes in our experiments, which used the HermiT reasoner.

Many of our employed heuristics to suggest axioms are based on counting. For instance, when determining whether a class A is appropriate as domain of a property p, we count the number of triples using p in predicate position and the number of subjects in those triples which are instances of A. The latter value is divided by the first value to obtain a score. We illustrate this using a simple example. Let the following triples be given (Turtle syntax):

```
@prefix dbpedia: <http://dbpedia.org/resource/>.
@prefix dbo:     <http://dbpedia.org/ontology/>.
dbpedia:Luxembourg dbo:currency     dbpedia:Euro;
                   rdf:type         dbo:Country.
dbpedia:Ecuador    dbo:currency     dbpedia:US_dollar;
                   rdf:type         dbo:Country.
dbpedia:Ifni       dbo:currency     dbpedia:Spanish_peseta;
                   rdf:type         dbo:PopulatedPlace.
dbo:Country        rdfs:subClassOf  dbo:PopulatedPlace.
```

In the above example, we would obtain a score of 66,7% (2 out of 3) for the class `dbo:Country` and 100% (3 out of 3) for the class `dbo:PopulatedPlace`[9] as candidates for the range of the property `dbo:currency`.

A disadvantage of using this straightforward method of obtaining a score is that it does not take the *support* for an axiom in the knowledge base into account. Specifically, there would be no difference between having 100 out of 100 correct observations or 3 out of 3 correct observations.

For this reason, we do not just consider the count, but the average of the 95% confidence interval of the count. This confidence interval can be computed efficiently by using the improved Wald method defined in [1]. Assume we have m observations out of which s were successful, then the approximation of the 95% confidence interval is as follows:

$$\max(0, p' - 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}}) \text{ to } \min(1, p' + 1.96 \cdot \sqrt{\frac{p' \cdot (1-p')}{m+4}})$$

$$\text{with } p' = \frac{s+2}{m+4}$$

This formula is easy to compute and has been shown to be accurate in [1].

In the above case, this would change the score to 57.3% (previously 66,7%) for `dbo:Country` and 69.1% (previously 100%) for `dbo:PopulatedPlace`. This indicates that there is not much support for either of those choices in the knowledge base. 100 out of 100 correct observations would score much higher (97.8%). The actual scores for the DBpedia Live as of May 2012 are 99.1% for the class `dbo:PopulatedPlace` and 97.6% for `dbo:Country`.

Note that in this implementation, more general classes in the hierarchy would always score higher. It might be desirable to correct this by slightly penalising very general classes. The drawback of such a penalty could be that the scores would be more difficult to understand for users. We leave the decision on such a penalty and a possible implementation as an area for future work.

[9] If the reasoning option is turned off, the score would be 33,3%.

3.2 Learning Subclass Axioms

In this section and the following sections, we will just focus on phase 2 of the above described workflow. This phase consists of obtaining the data required for generating enrichment suggestions. Since we mainly expect the data to be available in triple stores, the data acquisition is implemented via SPARQL queries. We will briefly present the necessary SPARQL query (or queries) here.

The first axiom type, we consider, are subclass axioms. Generating suggestions for subclass axioms allows to create a taxonomy from instance data. Basically the data for this can be fetched in 2 different ways:

Single Query

```
SELECT ?type (COUNT(?ind) AS ?count) WHERE {
  ?ind a <$class>.
  ?ind a ?type.
} GROUP BY ?type
```

The query assumes a `$class` as input for which we want to learn superclasses. It retrieves all instances of a class and then counts the types for each of those instances. A higher count indicates better candidates for superclasses. The disadvantage of this approach is that it puts high computational load on the SPARQL endpoint in case of very large data sets. An alternative implementation is to iterate through all results as shown below.[10] This way, each individual query is inexpensive for the endpoint as the information is obtained in small chunks. Moreover, in DL-Learner, we impose runtime limits on algorithms. This means that we stop iterating through results once a configurable time threshold has been reached. The score for suggestions is then approximated from the obtained sample. The drawback of this method is that the score can only be computed on a subset of the knowledge base whereas in the first method the whole data set is taken into account.

Iterative Query

```
SELECT ?ind ?type WHERE {
  ?ind a <$class>.
  ?ind a ?type.
}
LIMIT $limit OFFSET $offset
```

3.3 Learning Disjointness

For disjointness, we can use the same query as above:

```
SELECT ?type (COUNT(?ind) AS ?count) WHERE {
  ?ind a <$class>.
  ?ind a ?type.
} GROUP BY ?type
```

The only difference in terms of the query is that this time, a lower count indicates a candidate for disjointness. When running enrichment in batch mode, the

[10] Correct pagination in SPARQL with `LIMIT` and `OFFSET` only works with the sorting of the results by using `ORDER BY`, but we omit this in this paper for simplicity.

number of suggested disjointness axioms is minimsed by moving disjointness as far up the class hierarchy as possible (see Section 3.8).

In addition, we draw on [33,10] for computing disjointness. Several criteria, specifically taxonomic overlap, existing subsumption axioms and semantic similarity are used in order to determine the most useful disjointness axioms.

3.4 Property Subsumption/Disjointness

For properties, learning subsumption and disjointness is analogous to learning this kind of axioms for classes. The difference is that we count how often subject `?s` and object `?o` in the triples for a given property `$property` are also related via other properties `?p`.

```
SELECT ?p (COUNT(?s) AS ?count) WHERE {
  ?s ?p ?o.
  ?s <$property> ?o.
} GROUP BY ?p
```

3.5 Property Domain and Range

For domains of object properties and data properties we count the occurrences of types in the subject position of triples having the property.

```
SELECT ?type COUNT(DISTINCT ?ind) WHERE {
  ?ind <$property> ?o.
  ?ind a ?type.
} GROUP BY ?type
```

For property ranges, we issue different queries depending on whether a resource is a data or object property.

Object Properties. The object property case is analogous to learning domains as shown above, except this time we pay attention to the triple objects.

```
SELECT ?type (COUNT(DISTINCT ?ind) AS ?cnt) WHERE {
  ?s <$property> ?ind.
  ?ind a ?type.
} GROUP BY ?type
```

Data Properties. For data properties, we make use of the fact that every triple is annotated with its datatype in RDF, i.e. we can just count occurring datatypes.

```
SELECT ?datatype COUNT(DISTINCT ?ind) WHERE {
  ?ind <$property> ?val.
} GROUP BY (DATATYPE(?val) AS ?datatype)
```

3.6 Inverse Properties

To generate axioms which state that a property `p1` is the inverse of a property `p2` we run the query below, which retrieves properties `p` having subject and object occurring in the triples for the given property `$property` in swapped positions and count how often this happens.

```
SELECT ?p (COUNT(*) AS ?cnt) WHERE {
  ?s <$property> ?o.
  ?o ?p ?s.
} GROUP BY ?p
```

3.7 Property Characteristics

Based on the axiom type which shall be learned for a given property `$property`, either the number of triples (symmetry, asymmetry), the number of distinct subjects (functionality, reflexivity, irreflexivity), the number of distinct objects (inverse-functionality) or the number of connected triple pairs (transitivity) is computed in a first step. This value is then combined with the result of the corresponding query in Table 2.

Table 2. SPARQL queries used to learn the different types of OWL 2 property characteristics

Functionality	`SELECT COUNT(DISTINCT ?s) AS ?functional WHERE { ?s <$property> ?o1. FILTER NOT EXISTS {?s <$property> ?o2. FILTER(?o1 != ?o2)} }`
Inverse-Functionality	`SELECT COUNT(DISTINCT ?o) AS ?inversefunctional WHERE { ?s1 <$property> ?o. FILTER NOT EXISTS {?s2 <$property> ?o. FILTER(?s1 != ?s2)} }`
Symmetry	`SELECT (COUNT(*) AS ?symmetric) WHERE { ?s <$property> ?o. ?o <$property> ?s. }`
Asymmetry	`SELECT (COUNT(*) AS ?asymmetric) WHERE { ?s <$property> ?o. FILTER NOT EXISTS {?o <$property> ?s.} }`
Reflexivity	`SELECT (COUNT(DISTINCT ?s) AS ?reflexive) WHERE { ?s <$property> ?s. }`
Irreflexivity	`SELECT (COUNT(DISTINCT ?s) AS ?irreflexive) WHERE { ?s <$property> ?o. FILTER NOT EXISTS {?s <$property> ?s.} }`
Transitivity	`SELECT (COUNT(*) AS ?transitive) WHERE { ?s <$property> ?o. ?o <$property> ?o1. ?s <$property> ?o1. }`

3.8 Batch Mode

While the described methodology is designed to be applicable for learning axioms involving specific classes or properties, it is also possible to run the enrichment script in batch mode. In this case, the script first detects all schema entities, loops over them and calls the learning methods for all axiom types. Our evaluation shows that the approach still scales to large knowledge bases, e.g. DBpedia. To prevent flooding of the SPARQL endpoints, the batch mode contains configurable options to delay the execution of successive queries, and for the case that a timeout occurs to rerun the query after some waiting period.

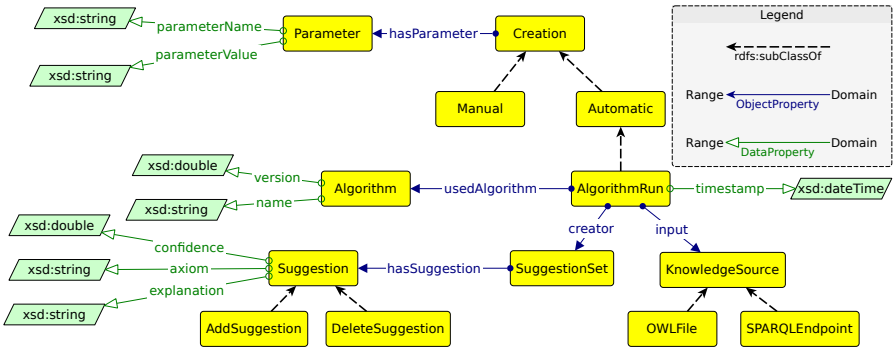


Fig. 2. Enrichment ontology used to store additional information occuring during learning proocess

Running the algorithms batched allows to add further optimisations, e.g. for disjointness between classes we give the opportunity to restrict the returned suggestions to pairs of most general classes, as due to inference disjointness is propagated to lower subclasses. Further possibilities we will investigate in the future are (1) the problem of coherency, i.e. some axiom types especially disjointness combined with e.g. subsumption can lead to unsatisfiable entities, (2) iterative creation of the knowledge base, i.e. taking into account earlier learned axioms for the generation of other axiom types, and (3) the minimization of the resulting ontology, i.e. finding an ontology in which none of the axioms can be inferred from other existing axioms.

4 Enrichment Ontology

As previously discussed, enrichment is usually a semi-automatic process. Each enrichment suggestion generated by an algorithm should be reviewed by a knowledge engineer who can then decide to accept or reject it. Because of this, there is a need for serialising enrichment suggestions such that the generation of them is independent of the process of accepting or rejecting them. Since all enrichment suggestions are OWL axioms, they could simply be written in an RDF or OWL file. However, this might be insufficient, because we lose a lot of metadata this way, which could be relevant for the knowledge engineer. For instance, algorithms may be able to store confidence values, statistics or even textual descriptions on why an enrichment is suggested. For this reason, we created an enrichment ontology[11], which is partially building on related efforts in [27] and `http://vocab.org/changeset/schema.html`. Such an interchange format is also relevant, because the process of generating enrichments for all schema elements in very large knowledge bases will often take several hours to complete. Furthermore, the metadata also simplifies reproducing algorithms results by storing algorithm versions, the used knowledge sources and time information. An overview of the ontology can be found in Figure 2.

[11] Available at `http://dl-learner.org/ontologies/enrichment.owl`

5 Preliminary Evaluation

To assess the feasibility of our approaches, we evaluated them on DBpedia [18]. We performed an enrichment on the DBpedia Live knowledge base [25], which at that time consisted of 385 million triples, 3.64 million things, 272 classes, 629 object properties and 706 data properties. We used a confidence threshold of 0.7 for the algorithm runs and showed at most 10 suggestions per entity and axiom type. Table 3 contains basic runtime information on the algorithms. It shows how many enrichment suggestions were made per axiom type, the runtime of the algorithm, the average score and the average of the maximum scores of each algorithm run. The algorithms require 10-161 seconds per ontology entity. To the best of our knowledge, there are no other approaches performing the same task to which we can compare our method in general. For a small number of the reported axiom types [31] performs a similar approach using association rule mining, which yields very similar results to our approach due to high similarity of the underlying heuristics for this axiom types.

Table 4 shows our evaluation results. In this evaluation we defined recall with respect to the existing DBpedia ontology. For instance, 180/185 in the subClassOf row indicates that we were able to re-learn 180 out of 185 such subclass relationships from the original DBpedia ontology. Higher numbers are an indicator that the methods do not miss many possible enrichment suggestions. The next column shows how many additional axiom were suggested, i.e. how

Table 3. Basic information on runtime and number of suggestions of the algorithms

algorithm	Avg. nr. of #suggestions	Avg. runtime in ms	timeout in %	Avg. score	Avg. max. score
disjoint classes	10.0	11957.0	0.00	1.00	1.00
subclass	2.5	98233.0	0.00	0.95	0.98
disjoint objectproperty	10.0	12384.0	0.16	1.00	1.00
equivivalent objectproperty	1.1	12509.0	0.16	0.96	0.96
functional objectproperty	1.0	19990.0	0.48	0.89	0.89
inv.funct. objectproperty	1.0	113590.0	4.29	0.86	0.86
objectproperty domain	3.1	11577.0	0.00	0.93	0.96
objectproperty range	2.6	14253.0	0.16	0.84	0.87
objectproperty subPropertyOf	1.1	56363.0	0.32	0.93	0.93
symmetric objectproperty	1.0	12730.0	0.32	0.80	0.80
transitive objectproperty	1.0	16830.0	1.91	0.84	0.84
irreflexive objectproperty	1.0	17357.0	0.16	0.97	0.97
reflexive objectproperty	0.0	10013.0	0.16	-	-
disjoint dataproperty	10.0	137204.0	3.97	1.00	1.00
equiv. dataproperty	1.0	161229.0	4.25	0.92	0.92
funct. dataproperty	1.0	18281.0	0.42	0.94	0.94
dataproperty domain	2.8	10340.0	0.28	0.94	0.96
dataproperty range	1.0	13516.0	0.42	0.96	0.96
dataproperty subPropertyOf	1.0	91284.0	4.11	0.88	0.88
OVERALL	3.0	55389.0	1.08	0.92	0.93

Table 4. Evaluation results

axiom type	recall	additional axioms	Estimated precision no	maybe	yes
SubClassOf	180/185	155	5	20	75
EquivalentClasses	0/0	1812	20	30	50
DisjointClasses	0/0	2449	0	0	100
SubObjectPropertyOf	0/0	45	18	9	18
EquivalentObjectProperties	0/0	40	40	0	0
DisjointObjectProperties	0/0	5670	0	0	100
ObjectPropertyDomain	385/449	675	10	22	68
ObjectPropertyRange	173/435	427	4	59	37
TransitiveObjectProperty	0/0	12	5	5	2
FunctionalObjectProperty	0/0	352	8	18	74
InverseFunctionalObjectProperty	0/0	173	72	3	25
SymmetricObjectProperty	0/0	3	0	0	3
ReflexiveObjectProperty	0/0	0	-	-	-
IrreflexiveObjectProperty	0/0	536	1	0	99
SubDataPropertyOf	0/0	197	86	8	6
EquivalentDataProperties	0/0	213	20	9	71
DisjointDataProperties	0/0	62	0	0	100
DataPropertyDomain	448/493	623	27	33	40
DataPropertyRange	118/597	79	0	0	100
FunctionalDataProperty	14/14	509	4	17	79

many axioms were suggested which are not in the original DBpedia ontology. The last three columns are the result of a manual evaluation. Both authors independently observed at most 100 axioms per type (possibly less, in case fewer than 100 suggestions were made) and evaluated them manually. Three different categories were used: "yes" indicates that it is likely that they would be accepted by a knowledge engineer, "maybe" are corner cases and "no" are those, which would probably be rejected. The evaluation at this stage is preliminary as it was only conducted by the authors. A full evaluation including several datasets and external reviewers is scheduled as future work.

In summary, we observed that axioms regarding the class hierarchy basically seem to be more easy to learn than axioms building the property hierarchy. We also noticed that we could suggest new axioms for all axiom types except for the ReflexiveObjectProperty ones. The reason is that DBpedia does not contain corresponding instance data. The low recall for the range axioms of object and data properties is mostly due to either missing or different type information on the triples' objects.

Below, we list some of the observations we made:

– The test set for irreflexive properties contained `dbo:isPartOf`, which is usually considered as a reflexive property. It is the only incorrect suggestion in this test set.

- The 5 missing subclass axioms are as follows:
  1. `dbo:Ginkgo subClassOf: dbo:Plant`
  2. `dbo:MixedMartialArtsLeague subClassOf: dbo:SportsLeague`
  3. `dbo:VoiceActor subClassOf: dbo:Actor` (each of those 3 axioms has only 1 triple and therefore too low support)
  4. `dbo:PoloLeague subClassOf: dbo:SportsLeague` (only 3 triples, therefore it had low support)
  5. `dbo:Bridge subClassOf: dbo:Building` (none of the bridges is actually a building according to DBpedia Live)
- As an example for the imperferct recall on object property domains, the results of the learning procedure for `dbo:hometown` are as follows: For the existing domain `dbo:Person` we only got a score of 0.3, whereas `dbo:Band` and `dbo:Organisation` achieved a score of approx. 0.7. This is because each `dbo:Band` was also a `dbo:Person` at this time in DBpedia Live.
- We discovered 3 symmetric object properties, namely `dbo:neighboringMunicipality`, `dbo:sisterCollege` and `dbo:currentPartner`.
- For most of the data properties, the learned axioms of the types SubDataPropertyOf and EquivalentDataProperties contained properties of the DBpedia namespace `http://dbpedia.org/property/`(dbp), e.g. EquivalentDataProperties(`dbo:drugbank`,`dbp:drugbank`). Mixing the two different ontologies is usually not desirable, hence the low precision in some of these cases. As a result, we now support more fine-grained control over the used ontology namespaces to avoid those problems.
- We missed some DataPropertyRanges, because sometimes the defined range in the ontology is a different datatype, compared to the one of the literal values in the triples. For instance `dbo:background` has a defined range `xsd:string`, but in the instance data the literals only have a language tag (which makes them implicit to `rdf:PlainLiteral`). `dbo:budget` (range in ontology: `xsd:double`, but `http://dbpedia.org/datatype/usDollar` used in the actual literals) is a different example. Clearly, in those cases data errors cause problems and our enrichment tool can be used to detect those.
- In some cases we learned a different datatype, so we missed the existing one and found an additional one. Most of these additional axioms would also be a reasonable but not optimal choice, e.g. for `dbo:populationTotal` we learned `xsd:integer`, whereas in the ontology `xsd:nonNegativeInteger` is defined.

6 Conclusion

We presented a set of approaches for schema enrichment, which covers most OWL 2 axioms. Those approaches were implemented and released in the DL-Learner machine learning framework. In our preliminary evaluation, we showed the feasibility of the methods for knowledge bases of the size of DBpedia.

In future work, we will investigate enhancements of the presented methods as indicated in the discussions of the respective approaches. In particular, we

closely collaborate with the authors of [31] to fine-tune the approach. One of the next steps will be to investigate how to generate a coherent ontology when combining all suggestions and how to use such an ontology for debugging large knowledge bases.

References

1. Agresti, A., Coull, B.A.: Approximate is better than "exact" for interval estimation of binomial proportions. The American Statistician 52(2), 119–126 (1998)
2. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. In: IJCAI 2007. AAAI Press (2007)
3. Baader, F., Sertkaya, B., Turhan, A.-Y.: Computing the least common subsumer w.r.t. a background terminology. J. Applied Logic 5(3), 392–420 (2007)
4. Badea, L., Nienhuys-Cheng, S.-H.: A Refinement Operator for Description Logics. In: Cussens, J., Frisch, A.M. (eds.) ILP 2000. LNCS (LNAI), vol. 1866, pp. 40–59. Springer, Heidelberg (2000)
5. Choi, N., Song, I.-Y., Han, H.: A survey on ontology mapping. SIGMOD Record 35(3), 34–41 (2006)
6. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: AAAI 1992, pp. 754–760 (1992)
7. Cohen, W.W., Hirsh, H.: Learning the CLASSIC description logic: Theoretical and experimental results. In: KR 1994, pp. 121–133. Morgan Kaufmann (1994)
8. Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-Intensive Induction of Terminologies from Metadata. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 441–455. Springer, Heidelberg (2004)
9. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL Concept Learning in Description Logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
10. Fleischhacker, D., Völker, J.: Inductive Learning of Disjointness Axioms. 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. 680–697. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-25106-1
11. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL class descriptions on very large knowledge bases. Int. J. Semantic Web Inf. Syst. 5(2), 25–48 (2009)
12. Iannone, L., Palmisano, I.: An Algorithm Based on Counterfactuals for Concept Learning in the Semantic Web. In: Ali, M., Esposito, F. (eds.) IEA/AIE 2005. LNCS (LNAI), vol. 3533, pp. 370–379. Springer, Heidelberg (2005)
13. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the semantic web. Applied Intelligence 26(2), 139–159 (2007)
14. Lehmann, J.: Hybrid Learning of Ontology Classes. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 883–898. Springer, Heidelberg (2007)
15. Lehmann, J.: DL-Learner: learning concepts in description logics. Journal of Machine Learning Research (JMLR) 10, 2639–2642 (2009)
16. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. Journal of Web Semantics 9, 71–81 (2011)
17. Lehmann, J., Bühmann, L.: ORE - A Tool for Repairing and Enriching Knowledge Bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 177–193. Springer, Heidelberg (2010)

18. Lehmann, J., Bizer, C., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Journal of Web Semantics 7(3), 154–165 (2009)
19. Lehmann, J., Haase, C.: Ideal Downward Refinement in the $\mathcal{EL}$ Description Logic. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 73–87. Springer, Heidelberg (2010)
20. Lehmann, J., Hitzler, P.: Foundations of Refinement Operators for Description Logics (Best Student Paper Award). In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 161–174. Springer, Heidelberg (2008)
21. Lehmann, J., Hitzler, P.: A Refinement Operator Based Learning Algorithm for the ALC Description Logic (Best Student Paper Award). In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 147–160. Springer, Heidelberg (2008)
22. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning Journal 78(1-2), 203–250 (2010)
23. Lisi, F.A.: Building rules on top of ontologies for the semantic web with inductive logic programming. Theory and Practice of Logic Programming 8(3), 271–300 (2008)
24. Lisi, F.A., Esposito, F.: Learning SHIQ+log rules for ontology evolution. In: SWAP 2008. CEUR Workshop Proceedings, vol. 426. CEUR-WS.org (2008)
25. Morsey, M., Lehmann, J., Auer, S., Stadler, C., Hellmann, S.: Dbpedia and the live extraction of structured data from wikipedia. Program: Electronic Library and Information Systems 46, 27 (2012)
26. Nienhuys-Cheng, S.-H., de Wolf, R. (eds.): Foundations of Inductive Logic Programming. LNCS, vol. 1228. Springer, Heidelberg (1997)
27. Palma, R., Haase, P., Corcho, Ó., Gómez-Pérez, A.: Change representation for OWL 2 ontologies. In: Hoekstra, R., Patel-Schneider, P.F. (eds.) OWLED. CEUR Workshop Proceedings, vol. 529. CEUR-WS.org (2009)
28. Rudolph, S.: Exploring Relational Structures Via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS (LNAI), vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
29. Sertkaya, B.: Ontocom P system description. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK. CEUR Workshop Proceedings, vol. 477, CEUR-WS.org (2009)
30. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. Technical report (August 01, 2008)
31. Völker, J., Niepert, M.: Statistical Schema Induction. 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. 124–138. Springer, Heidelberg (2011)
32. Völker, J., Rudolph, S.: Fostering web intelligence by semi-automatic OWL ontology refinement. In: Web Intelligence, pp. 454–460. IEEE (2008)
33. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning Disjointness. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 175–189. Springer, Heidelberg (2007)
34. Wu, H., Zubair, M., Maly, K.: Harvesting social knowledge from folksonomies. In: Proceedings of the Seventeenth Conference on Hypertext and Hypermedia, HYPERTEXT 2006, pp. 111–114. ACM, New York (2006)

Proxemic Conceptual Network Based on Ontology Enrichment for Representing Documents in IR

Chiraz Latiri[1], Lamia Ben Ghezaiel[2], and Mohamed Ben Ahmed[2]

[1] LIPAH Research Laboratory, Computer Sciences Department
Faculty of Sciences of Tunis,
El Manar University, 1068, Tunis, Tunisia
chiraz.latiri@gnet.tn
[2] Computer Sciences School
RIADI-GDL Research Laboratory,
Manouba University, 2010, Tunis, Tunisia
lamiagh@gmail.com
mohamed.benahmed@riadi.rnu.tn

Abstract. In this paper, we propose the use of a minimal generic basis of association rules (ARs) between terms, in order to automatically enrich an existing domain ontology. The final result is a proxemic conceptual network which contains additional implicit knowledge. Therefore, to evaluate our ontology enrichment approach, we propose a novel document indexing approach based on this proxemic network. The experiments carried out on the OHSUMED document collection of the TREC 9 filtring track and MeSH ontology showed that our conceptual indexing approach could considerably enhance information retrieval effectiveness.

Keywords: Text mining, Ontology enrichment, Information retrieval, Association rule, Generic bases, Similarity measures, Conceptual Indexing.

1 Introduction

Recently, several research communities in text mining and semantic web spent a determined efforts to conceptualize competencies of a given domain through the definition of a domain ontology. However, in order to make that ontology actually of use in applications, it is of paramount importance to enrich its structure with concepts as well as instances identifying the domain.

Many contributions in the literature related to Information Retrieval (IR) and text mining fields proved that domain ontologies are very useful to improve several applications such as ontology-based IR models [1]. While several ontology learning approaches extract concepts and relation instances directly from unstructured texts, in this paper, we show how an existing ontology can be automatically enriched by the use of text mining techniques. Especially, we are interested in mining a specific domain document collections in order to extract valid association rules [2] between concepts/terms . Thus, we propose to use a minimal generic basis of association rules, called $\mathcal{MGB}$, proposed in [3], to detect additional concepts for expanding ontologies. The result of our

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 72–86, 2012.

enrichment process is a proxemic conceptual network, denoted $\mathcal{O}_{\mathcal{MGB}}$, which unveils the semantic content of a document. To show the benefits of this proxemic conceptual network in the IR field, we propose to integrate it in a document conceptual indexing approach.

The remainder of the paper is organized as follows: Section 2 recalls paradigms for mining association rules between terms. In Section 3, we introduce a novel automatic approach of ontology enrichment based on the generic basis $\mathcal{MGB}$. Section 4 presents a document conceptual indexing approach based on the enriched ontology. Section 5 is devoted to the experimental evaluation, in which the results of the carried out experiments on OHSUMED collection and MeSH ontology are discussed. The conclusion and work in progress are finally presented in Section 6.

2 Mining Association Rules between Terms

In a previous work [3], we used in the text mining field, the theoretical framework of Formal Concept Analysis (FCA), presented in [4], in order to propose the extraction of a minimal generic basis of irredundant association rules between terms, named $\mathcal{MGB}$. In this respect, we formalize an extraction context made up of documents and index terms, called *textual context* [3].

Definition 1. *A textual context is a triplet $\mathfrak{M} = (\mathcal{C}, \mathcal{T}, \mathcal{I})$ where:*

- $\mathcal{C} := \{d_1, d_2, \ldots, d_n\}$ *is a finite set of n documents of a collection.*
- $\mathcal{T} := \{t_1, t_2, \cdots, t_m\}$ *is a finite set of m distinct terms in the collection. The set $\mathcal{T}$ then gathers without duplication the terms of the different documents which constitute the collection.*
- $\mathcal{I} \subseteq \mathcal{C} \times \mathcal{T}$ *is a binary (incidence) relation. Each couple $(d,\ t) \in \mathcal{I}$ indicates that the document $d \in \mathcal{C}$ has the term $t \in \mathcal{T}$.*

We called *termset*[1], a set of terms $T \subset \mathcal{T}$. Its support is formally defined as follows[2]:

$$Supp(T) = |\{d \mid d \in \mathcal{C} \ \wedge \ \forall\, t \in T : (d,\ t) \in \mathcal{I}\}| \tag{1}$$

Given a textual context $\mathfrak{M} = (\mathcal{C}, \mathcal{T}, \mathcal{I})$, we consider that an association rule R between terms is an implication of the form R: $T_1 \Rightarrow T_2$, where T_1 and T_2 are subsets of $\mathcal{T}$, and $T_1 \cap T_2 = \emptyset$. The termsets T_1 and T_2 are, respectively, called the *premise* and the *conclusion* of R. The rule R is said to be based on the termset T equal to $T_1 \cup T_2$. The *support* of a rule R: $T_1 \Rightarrow T_2$ is then defined as [3]:

$$Supp(R) = Supp(T), \tag{2}$$

while its *confidence* is computed as:

$$Conf(R) = \frac{Supp(T_1)}{Supp(T)}. \tag{3}$$

[1] By analogy to the *itemset* terminology used in data mining for a set of items.

[2] In this paper, we denote by $|X|$ the cardinality of the set X.

An association R is said to be *valid* if its confidence value, *i.e.*, *Conf*(R), is greater than or equal to a user-defined threshold denoted *minconf*. This confidence threshold is used to exclude non valid rules.

Given a document collection, the problem of mining association rules between terms consists in generating all association rules given user-defined *minsupp* and *minconf* thresholds. Several approaches in the literature deal with the redundancy problem. More advanced techniques that produce only a limited number of rules rely on Galois closure [4]. These techniques focus on extracting irreducible nuclei of all association rules, called *generic basis*, from which the remaining association rules can be derived [5]. An interesting discussion about the main generic bases of association rules is proposed in [3].

The huge number of irredundant association rules constitutes a real hamper towards many text mining-based applications. To overcome this problem, we proposed in [3] the use of a *minimal generic basis*, called $\mathcal{MGB}$. This basis involves only valid association rules, w.r.t. *minsupp* and *minconf*, having minimal premises and which maximize the number of terms in the conclusion part. In other words, for each rule $R \in \mathcal{MGB}$, there is no rule $R' \in \mathcal{MGB}$ such that $R' \neq R$, R and R' have the same premise, and the conclusion part of R' subsumes that of R. The conclusion part of R is thus a largest possible one for the associated premise. Please refer to [3] for a detailed description of $\mathcal{MGB}$ construction.

In this paper, we propose to disclose how that can be achieved when a domain ontology is enriched using irredundant association rules belonging to $\mathcal{MGB}$.

3 Ontology Enrichment Based on a Generic Basis of Association Rules

In the context of ontology maintenance, we tackle in this paper, the problem of enrichment of an existing ontology with additional concept derived from a compact set of irredundant association rules, *i.e.,* the generic basis $\mathcal{MGB}$ [3]. We aim to enhance the knowledge captured in a domain ontology and lead to a proxemic conceptual network. The main motivation behind the idea is to prove the ability to select automatically only relevant concepts for enrichment by using uniquely irredundant association rules between terms.

3.1 Related Works to Enrichment Ontology

In the literature, there is no common formal definition of what an ontology is. However, most approaches share a few core items: concepts, a hierarchical *is-a*-relation, and further relations. For sake of generality, we formalize an ontology in the following way [6]:

Definition 2. *An ontology is a tuple* $\mathcal{O} = \langle \mathcal{C}_\mathcal{D}, \leq_\mathcal{C}, \mathcal{R}, \leq_\mathcal{R} \rangle$*, where* $\mathcal{C}_\mathcal{D}$ *is a set whose elements are called concepts of a specific domain,* $\leq_\mathcal{C}$ *is a partial order on* $\mathcal{C}_\mathcal{D}$ (i.e., *a binary relation* is-a $\subseteq \mathcal{C}_\mathcal{D} \times \mathcal{C}_\mathcal{D}$*),* $\mathcal{R}$ *is a set whose elements are called relations, and* $\leq_\mathcal{R}$ *is a function which assigns to each relation name its arity.*

Throughout this paper, we will consider the Definition 2 to designate a specific domain ontology. Naturally, the construction of an ontology is hard and constitutes an expensive task, as one has to train domain experts in formal knowledge representation. This knowledge is usually evolvable and therefore an ontology maintenance process is required and plays a main role as ontologies may be misleading if they are not up to date [7].

Roughly speaking, ontology enrichment process is performed in two main steps namely: a *learning step* to detect new concepts and relations and a *placement step* which aims to find the appropriate place of these concepts and relations in the original domain ontology. In this work, we focus on methods dedicated to the discovery of new candidate terms from text and their relation with initial concepts in a domain ontology. Thus, two main classes of methods have been explored for detecting candidate terms, namely: (*i*) *Statistical based methods* which consist in counting the number of occurrences of a given term in the corpus [7]. They allow to identify the new concepts but they are not able to add them into the original ontology without the help of the domain expert; and (*ii*) *Syntactic based methods* which require a grammatical sentence analysis. Most of these methods include upstream a part of speech tagging process [9]. In [10], authors introduced the use of lexico-syntactic patterns to detect ontological relations. The main advantage of the syntactic based methods compared to statistical based ones is that they allow to put automatically new terms into the existing ontology. Nevertheless, they do not label new relations.

3.2 Enrichment Ontology Process

We assume that we have, for a given domain, a document collection denoted $\mathcal{C}$. Hence, in order to derive the generic basis of association rules between terms $\mathcal{MGB}$ from the collection $\mathcal{C}$, we used the algorithm GEN-MGB to get out irredundant associations between terms [3].

Furthermore, we consider an initial domain ontology denoted by $\mathcal{O}$ such as the medical ontology MeSH [11], and $\mathcal{C}_{\mathcal{O}}$ the set of its original concepts. A such ontology includes the basic primitives of an ontology which are concepts and taxonomic relations such as the subsomption link *is-a*. Then, to evaluate the strength of the semantic link between two concepts inside the ontology $\mathcal{O}$, we use a knowledge-based similarity measure such as the edge-based one of *Wu and Palmer* [12]. It is a measure between concepts in an ontology restricted to taxonomic links. Given two concepts C_1 and C_2, *Wu and Palmer's* similarity measure is defined as follow [12]:

$$Sim_{WP}(C_1, C_2) = \frac{2 \times depth(C)}{depth(C_1) + depth(C_2)} \tag{4}$$

where $depth(C_i)$ is the distance which separates the concept C_i to the ontology root and C is the common concept ancestor of C_1 and C_2 in the given ontology.

Our enrichment process aims to bring closer the original ontology $\mathcal{O}$ of the terms contained in the premises of $\mathcal{MGB}$ association rules. Once the new concepts placed in the ontology, we calculate the similarity measure which evaluate the semantic links existing between the concepts of enriched ontology, denoted in the sequel by $\mathcal{O}_{\mathcal{MGB}}$.

The $\mathcal{MGB}$-based enrichment process iterates through the following steps.

Step 1: Detecting Candidate Concepts for Enrichment. We calculate for each concept C_O in the initial ontology $\mathcal{O}$, the set of candidate concepts to be connected to C_O. This set includes the terms in the conclusion parts of valid association rules in $\mathcal{MGB}$, whose premise is C_O.

Step 2: Placement of New Concepts. In this step, we add the candidate concepts to the initial ontology $\mathcal{O}$, while maintaining existing relations. This allows to avoid adding redundant links in the case of a concept candidate to be linked to several concepts in the initial ontology $\mathcal{O}$. In other words, given a valid association rule $R : C_O \Rightarrow C_j$ in $\mathcal{MGB}$, we select the candidate concept C_j in the association rule R related to the initial concept $C_O \in \mathcal{O}$ where R has the greater confidence, among those in $\mathcal{MGB}$ and having C_O as premise.

Step 3: Computing of C_i Neighborhood and Similarity Measure. It is worth noting that in our work, we looked for a similarity measure between concepts, which is not a distance (does not satisfy similarity nor triangle inequality). Thus, we need a similarity function which expresses how much a concept is similar to another within the enriched ontology. In this respect, given an ontology $\mathcal{O}$ and $\mathcal{C}_\mathcal{O}$ the set of its concepts, we consider a similarity function, denoted $Sim_C : \mathcal{C}_\mathcal{O} \rightarrow [0, 1]$, if and only if $Sim_C(C) = 1$ and $0 \leq Sim_C(C_j) < 1$ for all $C_j \neq C \in \mathcal{C}_\mathcal{O}$.

Among extracted terms as conclusions of valid association rules in $\mathcal{MGB}$, there are some already known terms, as they are already referenced as concepts by the initial ontology $\mathcal{O}$, belonging to the set $\mathcal{C}_\mathcal{O}$. In order to link only new terms extracted with existing concepts, we propose to define the neighborhood of these concepts as follows:

Definition 3. *Let C_i be a concept in $\mathcal{C}_\mathcal{O}$, the neighborhood of C_i, denoted by $\mathcal{N}_{C_i}$, is the set of concepts in the ontology $\mathcal{O}$ that can be accessed from C_i by using the hierarchical link or by using one or more valid irredundant associations rules in $\mathcal{MGB}$.*

The relations between a concept C_i in the set $\mathcal{C}_\mathcal{O}$ and its neighborhood are evaluated through a statistical measure called *similarity measure* between C_i and its neighborhood, denoted $Sim_{\mathcal{O}_{\mathcal{MGB}}}$.

In our enrichment approach, three configurations are possible to evaluate the similarity measure $Sim_{\mathcal{O}_{\mathcal{MGB}}}$ between two concepts C_i and C_j in the enriched ontology $\mathcal{O}_{\mathcal{MGB}}$, given in the following definition:

Definition 4. *Given a concept C_i in the set $\mathcal{C}_\mathcal{O}$.*

1. *For a concept $C_j \in \mathcal{C}_\mathcal{O}$ and linked to C_i, we have :*

$$Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_i, C_j) = Sim_{WP}(C_i, C_j) \tag{5}$$

2. *If it exists a link between C_i and a concept C_j derived from an association rule in the generic basis $\mathcal{MGB}$, then:*

$$Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_i, C_j) = Conf(R : C_i \Rightarrow C_j) \tag{6}$$

3. *If C_j is a concept added to the enriched ontology $\mathcal{O}_{\mathcal{MGB}}$ and linked to C_i where $Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_i, C_j) = Conf(R : C_i \Rightarrow C_j) = \beta$, then each concept C_k in $\mathcal{C}_{\mathcal{O}}$ in relation with C_i where $Sim_{WP}(C_i, C_k) = \alpha$, is also in relation with C_j. The similarity measure is a mixte one and it is calculated as follows:*

$$Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_i, C_j) = \alpha \times \beta. \tag{7}$$

Thereby, we consider that the neighborhood $\mathcal{N}(C_i)$ of a concept C_i involves the set of k concepts belonging to the conceptual proxemic network $\mathcal{O}_{\mathcal{MGB}}$, in relation with the concept C_i, where the similarity measure between them is greater than or equal to a user-defined θ. Formally, we have:

$$\mathcal{N}(C_i) = \{C_j \mid Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_i, C_j) \geq \theta,\ j \in [1..k]\} \tag{8}$$

Example 1. The three configurations of the similarity measure evaluation are depicted in Figure 1, namely:

1. The two concepts C_1 and C_2 belong to the initial ontology $\mathcal{O}$, so $Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_1, C_2) = Sim_{WP}(C_1, C_2) = \alpha$ (*cf.* Equation (5)).
2. The concept C_{10} is selected from an association rule in $\mathcal{MGB}$, so: $Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_1, C_{10}) = Conf(C_1 \Rightarrow C_{10}) = 0.98$ (*cf.* Equation (6)).
3. A mixte similarity measure is computed between C_2 and C_{10} since C_1 is linked to C_{10} thanks to the valid association rule $R : C_1 \overset{0.98}{\Rightarrow} C_{10}$ and an initial relation exits in $\mathcal{O}$ between C_1 and C_2, so: $Sim_{\mathcal{O}_{\mathcal{MGB}}}(C_2, C_{10}) = 0.98 \times \alpha$ (*cf.* Equation (6)).

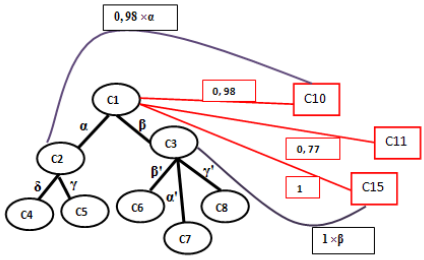


Fig. 1. Examples of similarity measure evaluation between concepts in $\mathcal{O}_{\mathcal{MGB}}$

The generated result, *i.e.*, the enriched ontology $\mathcal{O}_{\mathcal{MGB}}$, is then explored as a proxemic conceptuel network to represent the domain knowledge.

3.3 $\mathcal{O}_{\mathcal{MGB}}$: A Proxemic Conceptuel Network for Knowledge Representation

In what follows, we describe an original proposal for knowledge representation, namely a proxemic conceptual network resulting from the enriched ontology $\mathcal{O}_{\mathcal{MGB}}$. The relationships between the concepts of the conceptual network is quantified by the similarity measures introduced in Definition 4. The originality of our proxemic conceptual network is its completeness thanks to the combination, on the one hand of knowledge stemming from the initial ontology, *i.e.,* concepts and semantic relations, and, on the other hand implicit knowledge extracted as association rules between terms.

Thus, a concept in our proxemic conceptual network has three levels of semantic proximities, namely:

1. **A referential semantic:** assignes to each concept of $\mathcal{O}_{\mathcal{MGB}}$ an intensional reference, *i.e.,* its best concept sense through a disambiguation process.
2. **A differential semantic:** associates to each concept its neighbors concepts, *i.e.,* those correlated with it in the local context according to the Equation (8).
3. **An inferential semantic:** induced by irredundant association rules between terms that allow to link the initial ontology concepts to ones contained in valid association rules of $\mathcal{MGB}$ with respect to a minimum threshold of confidence $minconf$ and the proposed similarity measure.

Indeed, around each concept C_i in $\mathcal{O}_{\mathcal{MGB}}$, there is a semantic proxemic subspace which represents the different relations by computing similarity measure $Dist_{\mathcal{O}_{\mathcal{MGB}}}$ between its different neighbors concepts, and between concepts and their extensions. This results in an enriched knowledge representation.

In order to prove that the proxemic conceptual network can have a great interest in IR and can contribute to improve the retrieval effectiveness, we propose a document conceptual indexing approach based on $\mathcal{O}_{\mathcal{MGB}}$.

4 Evaluation of $\mathcal{O}_{\mathcal{MGB}}$ in Information Retrieval

Several ways of introducing additional knowledge into Information Retrieval (IR) process have been proposed in recent literature. In the last decade, ontologies have been widely used in IR to improve retrieval effectiveness [13]. Interestingly enough, such ontology-based formalisms allow a much more structured and sophisticated representation of knowledge than classical thesauri or taxonomy [14]. They represent knowledge on the semantic level thanks to concepts and relations instead of simple words.

Indeed, main contributions in *document conceptual indexing* issue are based on detecting new concepts from ontologies and taxonomies and use them to index documents instead of using simple lists of words [14, 15, 16]. Roughly, the indexing process is performed in two steps, namely (*i*) first detecting terms in the document text by mapping document representations to concepts in the ontology; then, (*ii*) disambiguating the selected terms.

In the remainder, we introduce a novel conceptual documents indexing approach in IR, based on the proxemic network $\mathcal{O}_{\mathcal{MGB}}$ which is the result of the enrichment of an initial ontology $\mathcal{O}$ with the generic basis $\mathcal{MGB}$. Our strategy involves three steps detailed below: (1) Identification and weighting representative concepts in the document through the conceptual network $\mathcal{O}_{\mathcal{MGB}}$; (2) Concepts disambiguation using the enriched ontology; and, (3) Building the document semantic kernel, denoted Doc-$\mathcal{O}_{\mathcal{MGB}}$, by selecting the best concepts-senses.

4.1 Identification and Weighting of Representative Concepts in the Document

We assume that a document d is represented by a set of terms, denoted by $d = \{t_1, \ldots, t_m\}$ and resulting from the terms extraction stage. A term t_i of a document d, denoted

$t = \{w_1, \ldots, w_n\}$, is composed of one or more words and its length $|t|$ is defined as the number of words in t.

This step aims to identify and weight, for each index term of the document, the corresponding concept in the proxemic conceptual proxemic $\mathcal{O}_{\mathcal{MGB}}$. Thus, the process of identification and weighting of representative concepts in a document d proceeds as follows:

1. **Projection of the document index on $\mathcal{O}_{\mathcal{MGB}}$:** It allows to identify mono-words or multi-words concepts which correspond to index terms with respect to their occurrence frequencies [17]. We select then the set of terms t_i characterizing the document d, denoted by $T(d)$, namely:

$$T(d) = \{(t_1, f(t_1)), \ldots, (t_n, f(t_n))\} \text{ such that } t_i \in d, \tag{9}$$

   where $f(t_i)$ means the occurrence frequency of t_i in d.
2. **Concepts weighting:** A widely used strategy in IR for weighting terms is $tf \times idf$ and its variants, expressed as $W(t,d) = tf(t) \times idf(t,d)$ [18]. In this formula, tf represents term frequency and idf is the inverse document frequency. In [15], authors proposed, for the case of multi-word terms, a statistical weighting method named $cf \times idf$ and based both on classical $tf \times idf$ measure and the length of the terms. So, for an extracted concept t composed of n words, its frequency in a document d is equal to the number of occurrences of the concept itself t, and the one of all its sub-concepts. The frequency is calculated as follows [15]:

$$cf(t) = f(t) + \sum_{t_i \in sub(t)} \left(\frac{|t_i|}{|t|} \times f(t_i)\right) \tag{10}$$

   where $sub(x)$ is the set of all possible sub-sets which can be derived from a term x, $|x|$ represents the number of words in x and $f(t)$ is the occurrence frequency of t in d.

   In this step of our conceptual indexing process, concepts weighting assignes to each concept a weight that reflects its importance and representativity in a document. We propose a new weighting measure which considers both *statistical* and *semantic* representativities of concepts in a given document.

   The key feature of the weighting way that we propose is to consider for a term t, in addition to its weight given by $cf(t) \times idf(t,d)$, the weights of concepts C_i belonging to its neighborhood.

   Hence, the *statistical representativity*, denoted W_{Stat}, is computed by using Equation (10), namely:

$$W_{Stat}(t,d) = cf(t) \times idf(t,d) \tag{11}$$

   Moreover, while considering the proxemic conceptuel network $\mathcal{O}_{\mathcal{MGB}}$, we propose that the *semantic representativity* of a term t in a document d, denoted $W_{Sem}(t,d)$, takes into account the different links in $\mathcal{O}_{\mathcal{MGB}}$ between each occurrence of t and other concepts in its neighborhood. This semantic representativity is computed by

using the semantic similarity measure $Dist_{\mathcal{O}_{\mathcal{MGB}}}$ as defined in Definition 4, between each occurrence C_i of a term t in $\mathcal{O}_{\mathcal{MGB}}$ and the concepts in its neighborhood $\mathcal{N}(C_i)$ (*cf.* Equation (8)). It is computed as follows:

$$W_{Sem}(t,d) = \sum_{C_i \in S_t} \sum_{C_j \in \mathcal{N}(C_i)} Dist_{\mathcal{O}_{\mathcal{MGB}}}(C_i, C_j) \times f(C_j) \tag{12}$$

such that $S_t = \{C_1, \ldots, C_n\}$ is the set of all concepts linked to the term t, *i.e.,* occurrences of t.

The underlying idea is that the global representativity of a term t in a document d, *i.e.,* its weight, further denoted W_{Doc}, is formulated as the combination between the statistical representativity and the semantic one, and is computed as:

$$W_{Doc}(t,d) = W_{Stat}(t,d) + W_{Sem}(t,d) \tag{13}$$

The document index, denoted $Index(d)$, is then generated by selecting only terms whose global representativity, *i.e.,* $W_{Doc}(t,d)$, is greater than or equal to a minimal representativity threshold.

4.2 Concepts Disambiguation

We assume that each term t_i in a document d can have multiple senses, denoted by S_i= $\{C_1^i,\ldots,C_n^i\}$, and are represented by corresponding concepts in $\mathcal{O}_{\mathcal{MGB}}$. Thus, a term t_i $\in Index(D)$ has $|S_i|$=n senses, *i.e.,* it represents n concepts in $\mathcal{O}_{\mathcal{MGB}}$.

Given a term t in the document index $Index(d) = \{t_1, \ldots, t_m\}$, the disambiguation aims to identify and to assign it the appropriate sens with respect to its context. We propose in the following an $\mathcal{O}_{\mathcal{MGB}}$-based disambiguation approach. In this regard, we consider that each index term in $Index(d)$ contributes to the semantic representation of d with only one sense even if a term can have different senses in the same document [17]. Hence, disambiguation task consists to select for each index term in $Index(d)$, its best sense in d, with respect to a computed score for each concept-sens in $\mathcal{O}_{\mathcal{MGB}}$.

In the literature, various methods and metrics have been proposed for disambiguating words in text [19]. In our case, we got inspired by the approach described in [15] which is based on the computation of a score for every concept-sense linked to an index term and using WordNet ontology.

Our disambiguation approach differs from that one proposed in [15] in the way of calculating the score. Indeed, we believe that only considering the semantic proximity between concepts is insufficient to detect the best sense of a term. We therefore suggest that the best sens to be assigned to a term t_i in the document d shall be strongly correlated with other elements, namely:

1. *The local context of the term t_i in the document d*: This means that the disambiguation of t_i considers its neighbors terms in the document d. We define the local context of a term t_i as follows:

   Definition 5. *The local context of a term t_i in a document d, denoted $Context_d(t_i)$, is a termset in $T(d)$ belonging to the same sentence where appears t_i.*

2. *The context of each sens in* $\mathcal{O}_{\mathcal{MGB}}$: The disambiguation of a concept C_i considers its neighborhood, *i.e.,* $\mathcal{N}_{C_i}$.
3. *Term representativity in the document context*: The best sense for a term t_i in d is that which is highly correlated with the most important sense in d. To do this, we integrate the term weight in the document, *i.e.,* its global representativity, computed w.r.t Equation (13).

In our disambiguation approach, we firstly define the weight of a concept-sense C_j^i in S_i as the weight of its associated index term t_i. That is, for a term t_i, the score of its j^{th} sense, denoted by C_j^i, is computed as:

$$C\_Score(C_j^i) = \sum_{C_v \in \mathcal{N}(C_j^i) \cup C_j^i} \sum_{t_l \in Context_d(t_i), l \neq i} Score_{Doc}(t_i, t_l) \times Dist(C_v, t_l) \tag{14}$$

where:

$$Score_{Doc}(t_i, t_l) = W_{Doc}(t_i, d) \times W_{Doc}(t_l, d) \tag{15}$$

and

$$Dist(C_v, t_l) = \sum_{k \in [1..n_l]} Dist_{\mathcal{O}_{\mathcal{MGB}}}(C_v, C_k^l) \tag{16}$$

such that n_l represents the number of senses in $\mathcal{O}_{\mathcal{MGB}}$ which is proper to each term t_l, $W_{Doc}(t_i, d)$ and $W_{Doc}(t_l, d)$ are the weights associated to t_i and t_l in the document d.

The concept-sense C_i which maximizes the score $C\_Score(C_j^i)$ is then retained as the best sense of the term t_i. Formally, we have:

$$C_i = \underset{j \in [1..n_i]}{\arg\max} \{C\_Score(C_j^i)\} \tag{17}$$

Indeed, by performing the different formulas (14), (15), (16) and (17), we have disambiguated the concept C_i which will be a node in the proxemic semantic network of the document d.

4.3 Building the Proxemic Semantic Network $Doc\text{-}\mathcal{O}_{\mathcal{MGB}}$

The last step of our document conceptual indexing process is the building of the proxemic network representing a document content, denoted in the sequel $Doc\text{-}\mathcal{O}_{\mathcal{MGB}}$. To do so, we select its nodes, *i.e.,* concept senses, by computing for each of them the best score C_Score. The nodes of $Doc\text{-}\mathcal{O}_{\mathcal{MGB}}$ are initialized with the selected concepts in the disambiguation step. Then, each concept of $Doc\text{-}\mathcal{O}_{\mathcal{MGB}}$ is declined in more intensions and extensions thanks to the structure of $\mathcal{O}_{\mathcal{MGB}}$, which are themselves linked to other concepts of the generic basis of association rules $\mathcal{MGB}$.

Therefore, thanks to the obtained structure, *i.e.,* $Doc\text{-}\mathcal{O}_{\mathcal{MGB}}$, we move from a simple index containing single index terms to a proxemic three-dimensional indexing space, synthesizing the three semantics as explained in Sub-section 3.3. The expected advantage on this novel document representation is to get a richer and more precise meaning representation in order to obtain a more powerful identification of relevant documents matching the query in an IR system.

5 Experimental Evaluation

In order to evaluate our ontology enrichment approach based on the generic basis $\mathcal{MGB}$, we propose to incorporate the generated conceptual proxemic network $Doc\text{-}\mathcal{O}_{\mathcal{MGB}}$ into a conceptual document indexing framework. For this purpose, we consider the well known medical ontology MeSH as domain ontology [11]. Indeed, in MeSH, each concept is described by a main heading (preferred term), one or many concept entries (non-preferred terms), qualifiers, scope notes, etc. Thus, we used main headings and qualifiers as indexing features in our evaluation.

5.1 Test Collection

We used the OHSUMED test collection, which is a MEDLINE sub-collection used for biomedical IR in TREC9 filtering Track, under the Terrier IR platform (http://terrier.org/). Each document has been annotated by human experts with a set of MeSH concepts revealing the subject matter(s) of the document. Some statistical characteristics of the OHSUMED collection are depicted in Table 2.

Table 1. OHSUMED test collection statistics

Number of documents	348, 566
Average document length	100 tokens
Number of queries	63
Average query length	12 terms
Average number of relevant docs/query	50

5.2 Experimental Setup

For measuring the IR effectiveness, we used exact precision measures P@10 and P@20, representing respectively the mean precision values at the top 10 and 20 returned documents, and MAP representing the *Mean Average Precision* computed over all topics. The purpose of our experimental evaluation is to determine the utility of our ontology enrichment approach on the MeSH ontology using irredundant association rules between terms which are derived from the document collection OHSUMED. Hence, we propose to assess the impact of exploiting the conceptual proxemic network $\mathcal{O}_{\mathcal{MGB}}$ in document indexing, on the retrieval effectiveness.

Therefore, we carried out two series of experiments applied on the articles titles and abstracts. The first one is based on the classical document indexing using the state-of-the-art weighting scheme OKAPI BM25 [20], as the baseline, denoted BM25. The second one concerns our conceptual indexing approach and consists of four scenarios, namely:

1. The first one concerns the document expansion using concepts manually assigned by human experts, denoted I_{Expert}.
2. The second one concerns the document expansion using only preferred concepts of the MeSH ontology before any enrichment, denoted I_{MeSH}.

3. The third one concerns the document expansion based on additional terms derived from valid association rules of the $\mathcal{MGB}$ generated from the document collection OHSUMED, denoted $I_{\mathcal{MGB}}$. Notice that we set up minimal support threshold $minsupp$ and minimal confidence threshold $minconf$, respectively to, 0.05 and 0.3.
4. The last one concerns the document expansion using concepts identified from the proxemic conceptual network $Doc\text{-}\mathcal{OMGB}$, denoted $I_{\mathcal{O}_{\mathcal{M}}\mathcal{GB}}$, which is the result of the MeSH enrichment with the generic basis of irredundant association rules $\mathcal{MGB}$ derived from OHSUMED collection.

5.3 Results and Discussion

We now present the experimental results of the proposed document indexing strategies. We assess the IR effectiveness using the extracted concepts and our proposed disambiguation approach.

Table 2. IR effectiveness (% change) over 63 queries

Strategies	MAP	P@10	P@20
Baseline (BM25)	23.96	41.9	35.00
I_{Expert}	29.55 (**+23.33**)	45.08 (+7.59)	39.92 (+14.06)
I_{MeSH}	24.73 (+3.21)	41.27 (-1.50)	35.87 (+2.49)
$I_{\mathcal{MGB}}$	24.96 (**+4.17**)	42.77(+2.08)	36.08 (+3.09)
$I_{\mathcal{O}_{\mathcal{MGB}}}$	28.17 (**+17.57**)	44.33 (+5.80)	38.17 (+10.86)

Table 3 depicts the IR effectiveness of the I_{Expert}, I_{MeSH} and our two conceptual indexing approaches based on the generic basis of association rules $\mathcal{MGB}$ and the proxemic conceptuel network $\mathcal{O}_{\mathcal{MGB}}$. We observe that in an automatic setting, our best indexing method, namely $I_{\mathcal{O}_{\mathcal{MGB}}}$, provides the highest improvement rate ($+17.57\%$) while the $\mathcal{MGB}$ based method only gives $+4.17\%$ in terms of MAP over the baseline BM25. This proves the interest to take into account both terms from association rules and the concepts selected from enriched ontology during the concept extraction process. Results highlight that using only concepts extracted from the MeSH ontology lead to a small improvement of IR effectiveness in the case of document indexing. Furthermore, we see that the I_{Expert}, $I_{\mathcal{MGB}}$ and $I_{\mathcal{O}_{\mathcal{MGB}}}$ methods consistently outperform the baseline BM25.

Although the gain of the $I_{\mathcal{O}_{\mathcal{MGB}}}$ method is smaller than the I_{Expert} method, which represents the best scenario, in terms of MAP (17.57% *vs.* $23,33\%$) (*cf.* Table 2), the former yields improvement in terms of P@10 and P@20, which is less significant in the others methods, namely I_{MeSH} and $I_{\mathcal{MGB}}$.

Figure 2 sheds light on the advantage of the insight gained through the $\mathcal{MGB}$ irredundant association rules and the conceptual proxemic network $\mathcal{O}_{\mathcal{MGB}}$ in the context of conceptual document indexing. We note that the increase of the precision at 11 points of recall with the $I_{\mathcal{MGB}}$ method is not so important with respect to the baseline BM25. This can be explained by the fact that OHSUMED is a scientific medical collection

where terms have very weak distributions and marginally co-occur. Moreover, an important part of the vocabulary is not used, since it is not correctly analyzed, due to the used tagger which does not identify specific and scientific terms of OHSUMED. Moreover, we notice in our experiments that the improvement of the average precision is less significant for high support values. Indeed, extracting association rules, when considering a high support value, leads to some trivial associations between terms that are very frequent in the document collection.

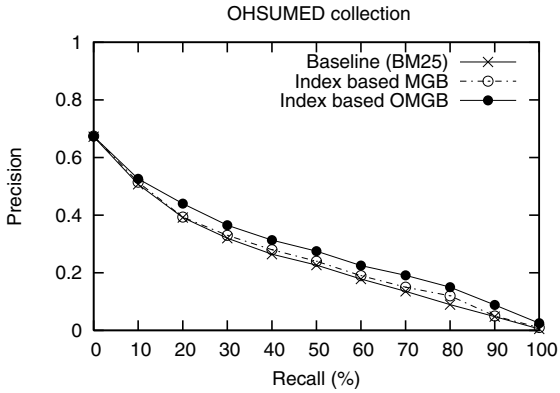


Fig. 2. Precision-recall curves corresponding to the baseline *vs* index based on $\mathcal{MGB}$ and $\mathcal{O}_{\mathcal{MGB}}$

In order to show how our indexing approach based on the conceptual proxemic network $\mathcal{O}_{\mathcal{MGB}}$ is statistically significant, we perform the Wilcoxon signed rank test [21] between means of each ranking obtained by our indexing method and the baseline BM25. The reason for choosing the Wilcoxon signed rank test is that it is more powerful and indicative test as it considers the relative magnitude in addition to the direction of the differences considered. Experimental results for a significance level $\alpha = 1\%$, show with the paired-sample Wilcoxon-test, that our based $\mathcal{O}_{\mathcal{MGB}}$ document indexing approach is statistically significant ($p\text{-}value < 0.01\%$) compared to the baseline BM25. Thus, we conclude that conceptual indexing by an enriched ontology with irredundant association rules between terms, would significantly improve the biomedical IR performance.

6 Conclusion

The work developed in this paper lies within the scope of domain ontologies enrichment and their application in IR field. We have introduced an automatic enrichment approach based on a generic basis of association rules between terms [3] to identify additional concepts linked to ontology concepts. The placement of new concepts is carried out through the defined similarity measure and the neighborhood concept. The result is a proxemic conceptual network where nodes represent disambiguated concepts and edges are materialized by the value of similarity measure between concepts. To evaluate the contribution of the conceptual proxemic network in the retrieval effectiveness, we integrate it in a conceptual document indexing. In this regard, the carried out experiments

using OHSUMED collection and MeSH ontology which highlighted an improvement in the performances of the information retrieval system, in terms of both recall and precision metrics. As work in progress, we focus on enrichment of multilingual ontologies by means of inter-lingual association rules between terms introduced in [22].

Acknowledgements. This work was partially supported by the French-Tunisian project CMCU-UTIQUE 11G1417.

References

1. Song, M., Song, I., Hu, X., Allen, R.B.: Integration of association rules and ontologies for semantic query expansion. Data and Knowledge Engineering 63(1), 63–75 (2007)
2. Agrawal, R., Skirant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Databases (VLDB 1994), Santiago, Chile, pp. 478–499 (September 1994)
3. Latiri, C., Haddad, H., Hamrouni, T.: Towards an effective automatic query expansion process using an association rule mining approach. Journal of Intelligent Information Systems (2012), doi: 10.1007/s10844–011–0189–9
4. Ganter, B., Wille, R.: Formal Concept Analysis. Springer (1999)
5. Zaki, M.J.: Mining non-redundant association rules. Data Mining and Knowledge Discovery 9(3), 223–248 (2004)
6. Cimiano, P., Hotho, A., Stumme, G., Tane, J.: Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 189–207. Springer, Heidelberg (2004)
7. Di-Jorio, L., Bringay, S., Fiot, C., Laurent, A., Teisseire, M.: Sequential Patterns for Maintaining Ontologies over Time. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1385–1403. Springer, Heidelberg (2008)
8. Parekh, V., Gwo, J., Finin, T.W.: Mining domain specific texts and glossaries to evaluate and enrich domain ontologies. In: Proceedings of the International Conference on Information and Knowledge Engineering, IKE 2004, pp. 533–540. CSREA Press, Las Vegas (2004)
9. Bendaoud, R., Napoli, A., Toussaint, Y.: Formal Concept Analysis: A Unified Framework for Building and Refining Ontologies. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 156–171. Springer, Heidelberg (2008)
10. Navigli, R., Velardi, P.: Ontology Enrichment Through Automatic Semantic Annotation of On-Line Glossaries. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 126–140. Springer, Heidelberg (2006)
11. Díaz-Galiano, M.C., García-Cumbreras, M.Á., Martín-Valdivia, M.T., Montejo-Ráez, A., Ureña-López, L.A.: Integrating MeSH Ontology to Improve Medical Information Retrieval. In: Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D.W., Peñas, A., Petras, V., Santos, D. (eds.) CLEF 2007. LNCS, vol. 5152, pp. 601–606. Springer, Heidelberg (2008)
12. Wu, Z., Palmer, M.: Verb semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, New Mexico, USA, pp. 133–138 (June 1994)
13. Vallet, D., Fernández, M., Castells, P.: An Ontology-Based Information Retrieval Model. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 455–470. Springer, Heidelberg (2005)
14. Andreasen, T., Bulskov, H., Jensen, P.A., Lassen, T.: Conceptual Indexing of Text Using Ontologies and Lexical Resources. In: Andreasen, T., Yager, R.R., Bulskov, H., Christiansen, H., Larsen, H.L. (eds.) FQAS 2009. LNCS, vol. 5822, pp. 323–332. Springer, Heidelberg (2009)

15. Baziz, M., Boughanem, M., Aussenac-Gilles, N., Chrisment, C.: Semantic cores for representing documents in IR. In: Proceedings of the 2005 ACM Symposium on Applied Computing, SAC 2005, pp. 1011–1017. ACM Press, New York (2005)
16. Dinh, D., Tamine, L.: Combining Global and Local Semantic Contexts for Improving Biomedical Information Retrieval. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 375–386. Springer, Heidelberg (2011)
17. Amirouche, F.B., Boughanem, M., Tamine, L.: Exploiting association rules and ontology for semantic document indexing. In: Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU 2008), Malaga, Espagne, pp. 464–472 (June 2008)
18. Salton, G., Buckely, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
19. Navigli, R.: Word sense disambiguation: A survey. ACM Comput. Surv. 41, 1–69 (2009)
20. Jones, K.S., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments. Information Processing and Management 36(6), 779–840 (2000)
21. Smucker, M.D., Allan, J., Carterette, B.: A comparison of statistical significance tests for information retrieval evaluation. In: Proceedings of the 16th International Conference on Information and Knowledge Management, CIKM 2007, pp. 623–632. ACM Press, Lisboa (2007)
22. Latiri, C., Smaïli, K., Lavecchia, C., Langlois, D.: Mining monolingual and bilingual corpora. Intelligent Data Analysis 14(6), 663–682 (2010)

Extracting Multilingual Natural-Language Patterns for RDF Predicates

Daniel Gerber and Axel-Cyrille Ngonga Ngomo

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany
{dgerber,ngonga}@informatik.uni-leipzig.de
http://aksw.org

Abstract. Most knowledge sources on the Data Web were extracted from structured or semi-structured data. Thus, they encompass solely a small fraction of the information available on the document-oriented Web. In this paper, we present BOA, a bootstrapping strategy for extracting RDF from text. The idea behind BOA is to extract natural-language patterns that represent predicates found on the Data Web from unstructured data by using background knowledge from the Data Web. These patterns are then used to extract instance knowledge from natural-language text. This knowledge is finally fed back into the Data Web, therewith closing the loop. The approach followed by BOA is quasi independent of the language in which the corpus is written. We demonstrate our approach by applying it to four different corpora and two different languages. We evaluate BOA on these data sets using DBpedia as background knowledge. Our results show that we can extract several thousand new facts in one iteration with very high accuracy.

1 Introduction

The population of the Data Web has been mainly carried out by transforming semi-structured and structured data available on the Web into RDF. Yet, while these approaches have successfully generated the more than 30 billion triples currently available on the Linked Open Data Cloud [1], they rely on background data that encompasses solely 15-20% [5] of the information on the Web, as the rest of the information in the document-oriented Web is only available in unstructured form. Consequently, the data in the Linked Data Web suffers from a lack of coverage and actuality that has been eradicated from the Web by Web 2.0 and crowdsourcing approaches. For example, while the Wikipedia text fragment "`... reputedly designed by Robert Mills, architect of the Washington Monument ...`" states that the triple "`dbr:Washington_Monument dbo:architect dbr:Robert_Mills`" holds, this triple is not included in DBpedia 3.7. In addition, being able to convert natural language to structured data makes manifold applications such as using SPARQL for question answering [14] possible by allowing that the pattern `born in` from questions such as `Which actors were born in Germany?` to be mapped explicitly to the relation `dbo:birthPlace`.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 87–96, 2012.

In this paper, we extend the BOA framework[1] presented in [6]. The goal of the BOA framework is to allow extracting structured data as RDF from unstructured data. Unlike many approaches (e.g., [2]) which start with their own ontologies and background knowledge as seeds, BOA makes use of the Linked Data Web to retrieve high-confidence natural-language patterns that express the predicates available in the Data Web. In contrast to its previous model, BOA uses a supervised machine learning approach trained on a set of manually annotated patterns to recognize high-confidence patterns. Based on these patterns, BOA can extract new instance knowledge (i.e., both new entities and relations between these new entities) from the Human Web with high accuracy. Our approach is completely agnostic of the knowledge base upon which it is deployed. It can thus be used on the whole Data Web. In addition, our extension of BOA implements generic pattern extraction algorithms that can be used to retrieve knowledge from sources written in different languages. Consequently, it can also be used on the whole Human Web.

The main contributions of this paper are as follows: (1) We present the novel approach implemented by the BOA framework and apply it to corpora written in English and in German. (2) We provide a multilingual library of natural-language representations of predicates found on the Data Web (especially in DBpedia). (3) We present a set of features that can be used to distinguish high-quality from poor natural-language patterns for Data Web predicates. (4) We evaluate our machine-learning approach and the BOA framework on 4 text datasets against DBpedia and show that we can achieve a high-accuracy extraction in both languages. The rest of this paper is structured as follows: In Section 2, we give an overview of previous work that is related to our approach. Thereafter, in Section 3, we present our bootstrapping framework and several insights that led to the approach currently implemented therein. In Section 4 we evaluate our approach on two different data sets and show its robustness and accuracy. Finally, we sum up our results and conclude.

2 Related Work

BOA is related to a large number of disciplines due to the different areas of knowledge from which it borrows methods. Like Information Extraction approaches, BOA aims to detect entities in text. Three main categories of natural language processing (NLP) tools play a central role during the extraction of knowledge from text: Keyphrase Extraction (KE) [8], Named Entity Recognition (NER) [4] and Relation Extraction (RE) [10]. While these three categories of approaches are suitable for the extraction of facts from NL, the use of the Data Web as source for background knowledge for fact extraction is still in its infancy. [10] coined the term "distant supervision" to describe this paradigm but developed an approach that led to extractors with a low precision (approx. 67.6%). Services

[1] A demo of the framework can be found at `http://boa.aksw.org`. The code of the project is at `http://boa.googlecode.com`

such as Alchemy[2], FOX [12] and Spotlight [9] reach better precision scores and allow to extract entities and relations from text. Yet, they do not rely on the Data Web as training data and are thus restricted with respect to the number of relations they can detect. The problem of extracting knowledge from the Web at large scale, which is most closely related to this paper, has been the object of recent research, especially in the projects ReadTheWeb and PROSPERA. The aim of the ReadTheWeb project[3] [2] is to create the never-ending language learner NELL that can read webpages. To achieve this goal, NELL is fed with the ClueWeb09[4] data set and an initial ontology. In each iteration, NELL uses the available instance knowledge to retrieve new instances of existing categories and relations between known instances by using pattern harvesting. The approach followed by PROSPERA [11] is similar to that of NELL but relies on the iterative harvesting of n-grams-itemset patterns that allow generalizing NL patterns found in text.

Our approach goes beyond the state of the art in two key aspects. First, it is the first approach to extract multi-lingual natural-language patterns from the Linked Data Web. In addition, it makes use of the Data Web as background knowledge, while the approaches ReadTheWeb and PROSPERA rely on their own ontology for this purpose. Moreover, BOA can generate RDF and can thus be used to populate a knowledge base that can be readily made available for querying via SPARQL, integrating and linking. Finally, our experiments show that our approach can extract a large number of statements (like PROSPERA and [10]) with a high precision (like ReadTheWeb).

3 Approach

An overview of the workflow implemented by BOA is given in Figure 1. The input for the BOA framework consists of a set of knowledge bases, a text corpus (mostly extracted from the Web) and (optionally) a Wikipedia dump[5]. When provided with a Wikipedia dump, the framework begins by generating surface forms for all entities in the source knowledge base. The surface forms used by BOA are generated by using an extension of the method proposed in [9]. For each predicate p found in the input knowledge sources, BOA carries out a sentence-level statistical analysis of the co-occurrence of pairs of labels of resources that are linked via p. Instead of using a hard-coded evaluation function like in previous work, BOA then uses a supervised machine-learning approach to compute the score of patterns for each combination of corpus and knowledge bases. In a final step, our framework uses the best-scoring patterns for each relation to generate RDF data. This data and the already available background knowledge can now be used for a further iteration of the approach. In the following, we describe the core steps of BOA in more detail. Throughout this description, we will use the example of generating new knowledge for the `dbpedia:architect` relation.

[2] http://www.alchemyapi.com
[3] http://rtw.ml.cmu.edu
[4] http://lemurproject.org/clueweb09
[5] http://dumps.wikimedia.org/

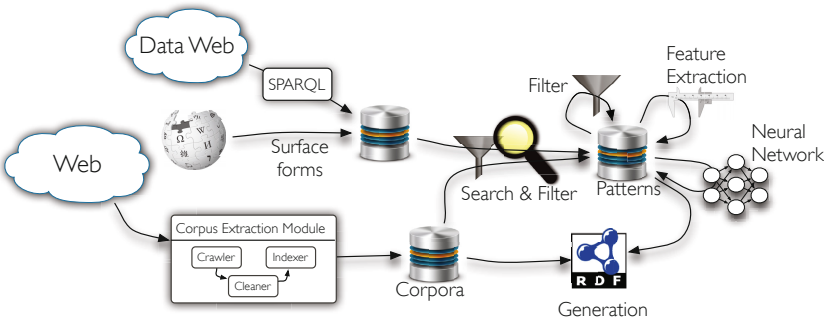


Fig. 1. Overview of the BOA approach

3.1 Pattern Extraction

Let $\mathcal{K}$ be the knowledge base that is used as background knowledge. The first and optional step of the pattern extraction is the computation of surface forms $\mathcal{S}_r$ for the subject and objects of a relation p for which patterns are to be extracted. To extract surface forms for resources $r \in \mathcal{K}$, we use Wikipedia's redirect and disambiguation pages as described in [9]. Overall, the average number of surface forms per resource was 1.66 for German and 2.36 for English. The pattern search is carried out independently for each predicate. Let $\mathtt{p} \in \mathfrak{P}$ be a predicate whose natural-language representations are to be detected, where $\mathfrak{P}$ is the set of all predicates. We use the symbol "$\in$" between triples and knowledge bases to signify that a triple can be found in a knowledge base. The starting point for the pattern search for $\mathtt{p}$ is the set of pairs $\mathcal{I}(\mathtt{p}) = \{(s, o) : (s\ \mathrm{p}\ o) \in \mathcal{K}\}$ that instantiate $\mathtt{p}$. In the following, we use $\lambda(x)$ to signify the set of labels of any resource x and $\mu(x)$ to signify x's URI. The pattern search process begins with the even distribution of the set $\mathcal{I}(\mathtt{p})$ across pattern search threads. Each of these threads then retrieves all sentences which contain both labels of all combination of $(\lambda(s), \lambda(o))$ from the input corpus. If a thread finds a sentence σ that contains a label $l_s \in \lambda(s)$ and $l_o \in \lambda(o)$, it deletes all tokens that are not found between l_s and l_o in σ. The labels are then replaced with the placeholders D for l_s and R for l_o. We call the resulting string a *natural-language representation* (NLR) of $\mathtt{p}$ and denote it with θ. Each θ extracted is used to create a new instance of a BOA pattern.

Definition 1 (BOA Pattern). *A BOA pattern is a pair* $\mathcal{P} = (\mu(\mathtt{p}), \theta)$*, where* $\mu(\mathtt{p})$ *is* $\mathtt{p}$*'s URI and* θ *is a natural-language representation of* $\mathtt{p}$*.*

Definition 2 (BOA Pattern Mapping). *A BOA pattern mapping is a function* $\mathcal{M}$ *such that* $\mathcal{M}(\mathtt{p}) = \mathfrak{S}$ *, where* $\mathfrak{S}$ *is the set of natural-language representations for* $\mathtt{p}$*.*

```
dbr:Empire_State_Building dbo:architect dbr:Shreve,_Lamb_and_Harmon
dbr:Empire_State_Building rdfs:label 'Empire State Building'@en
dbr:Shreve,_Lamb_and_Harmon rdfs:label 'Shreve, Lamb and Harmon'@en
```

Listing 1. RDF snippet used for pattern search

Table 1. Example sentences for pattern search

Sentence with $\lambda(s)$ before $\lambda(o)$	Sentence with $\lambda(o)$ before $\lambda(s)$
"... **Shreve, Lamb and Harmon** *also designed the* **Empire State Building** ."	"The **Empire State Building** *was designed by* **William F. Lamb** ..."

For example, consider the RDF snippet from Listing 1 derived from DBpedia. Querying the index of an underlying corpus for sentences which contain both entity labels returns the sentences depicted in Table 1 amongst others. We can replace "Empire State Building" with D, because it is a label of the subject of the `:architect` triple, as well as replace "Shreve, Lamb and Harmon" and "William F. Lamb" (a surface form of l_r) with R because it is one label of the object of the same triple. These substitutions lead to the BOA patterns (`:architect`, "D *was designed by* R") and (`:architect`, "R *also designed the* D"). For the sake of brevity and in the case of unambiguity, we also call θ "pattern". Patterns are only considered for storage and further computation if they withstand a first filtering process. For example, they must contain more than one non stop-word, have a token count between certain thresholds and may not begin with "and" or ", and". In addition to $\mathcal{M}(\mathtt{p})$ for each $\mathtt{p}$, we compute the number $f(\mathcal{P}, s, o)$ of occurrences of $\mathcal{P}$ for each element (s, o) of $\mathcal{I}(\mathtt{p})$ and the ID of the sentences in which $\mathcal{P}$ was found. Based on this data, we can compute (1) the total number of occurrences of a BOA pattern $\mathcal{P}$, dubbed $f(\mathcal{P})$; (2) the number of sentences that led to θ and that contained $\lambda(s)$ and $\lambda(o)$ with $(s, o) \in \mathcal{I}(\mathtt{p})$, which we denote $l(s, o, \theta, \mathtt{p})$ and (3) $\mathcal{I}(\mathtt{p}, \theta)$ is the subset of $\mathcal{I}(\mathtt{p})$ which contains only pairs (s, o) that led to θ. Thereafter we apply a second filtering process, where patterns which do not abide a threshold for $|\mathcal{I}(\mathtt{p}, \theta)|$, $\max(l(s, o, \theta, \mathtt{p}))$ and $f(\mathcal{P})$ are removed. We denote the set of predicates such that the pattern $\theta \in \mathcal{M}(\mathtt{p})$ by $\mathfrak{M}(\theta)$. Note that pattern mappings for different predicates can contain the same pattern.

3.2 Feature Extraction

The feature extraction is applied on all patterns which overcome both filtering processes. Note that although BOA is designed to work independently from the language of the underlying corpus, it can be tailored towards a given language. For example the ReVerb and IICM feature exploit knowledge that is specific to English. The first three features BOA relies upon are the support, specificity and typicity as described in [6]. In addition, we rely on the three supplementary features dubbed IICM, ReVerb and Tf-Idf. The **I**ntrinsic **I**nformation **C**ontent **M**etric (IICM) captures the semantic relatedness between a pattern's NLR and the property it expresses. This similarity measure was introduced in [13] and is based on the Jiang-Conrath similarity measure [7]. We apply this measure to each BOA pattern mapping independently. First we retrieve all synsets for each token of the pattern mappings associated *rdfs:label* from WordNet. If no such synsets are found we use the tokens of the *rdfs:label* of $\mathcal{M}(\mathtt{p})$. We then apply the IICM measure pairwise to these tokens and the tokens derived from one $\mathcal{M}(\mathtt{p})$

assigned pattern's NLR. The IICM score for one pattern is then the maximal value of the similarity values of all pairs. **ReVerb** has been introduced in [3] and distinguishes good from bad relation phrases by measuring how well they abide to a predefined part-of-speech-based regular expression. Since the input of ReVerb is a POS-tagged sentence, but a pattern is only a substring of a sentence, we use all sentences we found the pattern in (see Section 3.1) as ReVerbs input. For all of ReVerb's extracted relations of a particular sentence we check if it matches the pattern in question and use ReVerb's trained logistic regression classifier to assign a confidence score to this extraction. Note that BOA focuses on the relation between two given resources and discards all other extractions, since those are not mappable to the background knowledge. Finally, we calculate a patterns ReVerb feature as the average of all scored extractions. The **Tf-Idf** features are an adaption of the tf-idf score used in information retrieval and text mining. The intuition behind this feature is to distinguish relevant from irrelevant patterns for a given pattern mapping $\mathcal{M}(\mathtt{p})$. In the BOA case a document is considered to be all tokens of all patterns (without stop-words and the placeholders "D" and "R") of one pattern mapping. In other words, the total number of documents is equal to the number of pattern mappings with patterns. We then calculate the features $idf(\mathtt{p})$ and $tf(\mathtt{p})$ for each token of the patterns NLR as follows:

$$idf(\mathtt{p}) = \sum_{t \in \mathcal{T}(\mathtt{p})} log\left(\frac{|\mathcal{M}(\mathtt{p})|}{df(t)+1}\right) + 1 \qquad tf(\mathtt{p}) = \sum_{t \in \mathcal{T}(\mathtt{p})} \sqrt{f(t)}$$

Where $df(t)$ is the document frequency of t, $f(t)$ the term frequency of t and $\mathcal{T}(\mathtt{p})$ the set of tokens for a pattern $\mathtt{p}$.

3.3 Scoring Approach

Given the number of features that characterize the input data, devising a simple scoring function transforms into a very demanding task. In this work, we address the problem of computing a score for each BOA pattern by using feedforward neural networks. The input layer of our network consists of as many neurons as features for patterns exist while the output neuron consists of exactly one neuron whose activation was used as score. We used the sigmoid function as transfer function. For each data set, we trained the neural network by using manually annotated patterns (200 in our experiments). The patterns were extracted from the set of all patterns generated by BOA by first randomly sampling the same number of patterns for each predicate (7 in our experiments) and selecting a subset of these patterns for annotation.

3.4 RDF Generation

The generation of RDF out of the knowledge acquired by BOA is the final step of the extraction process and is carried out as follows: For each pattern θ and each predicate $\mathtt{p}$, we first use the Lucene index to retrieve sentences that contain θ stripped from the placeholders "D" and "R". These sentences are subsequently processed by a NER tool that is able to detect entities that are of the

`rdfs:domain` and `rdfs:range` of `p`. Thereafter, the first named entities within a limited distance on the left and right of θ which abide by the domain and range restrictions of `p` are selected as labels for subject and object of `p`. Each of the extracted labels is then fed into the URI retrieval and disambiguation service implemented by the FOX framework [12]. If this service returns a URI, then we use it for the label detected by BOA. Else, we create a new BOA URI. By applying our approach, we were able to extract the triples shown in Listing 2 from the text fragment " `... reputedly designed by Robert Mills, architect of the Washington Monument.`".

```
dbr:Washington_Monument dbo:architect dbr:Robert_Mills .
dbr:Washington_Monument rdf:type dbo:Building .
dbr:Washington_Monument rdfs:label "Washington Monument"@en .
dbr:Robert_Mills rdf:type dbo:Architect .
dbr:Robert_Mills rdfs:label "Robert Mills"@en .
```

Listing 2. RDF snippet generated by BOA

Note that `dbr:Washington_Monument dbo:architect dbr:Robert_Mills` is not included in DBpedia but explicitly stated in Wikipedia [6].

4 Evaluation

The aim of our evaluation was three-fold. First, we aimed at testing how well BOA performs on different languages. To achieve this goal, we applied BOA to German and English. Our second goal was to determine the accuracy of BOA's extraction. For this purpose, we sample 100 triples from the data extracted by BOA from each corpus and had two annotators measure the precision of these samples manually. Finally, we wanted to compute the amount of (new) knowledge that can be extracted by BOA. For this purpose, we compute the number of new triples that we were able to extract. We excluded temporal properties from the evaluation as BOA does not yet distinguish between different time expressions and conjugations. We evaluated our approach on the 4 corpora described in Table 2.

Table 2. Statistical overview of German and English text corpus

Corpus	Sentences	Tokens	Unique Tokens	Tokens per Sentence
en-wiki	58.0M	1,240.6M	6.8M	21.4
en-news	214.3M	4,745.1M	17.6M	22.1
de-wiki	24.6M	428.4M	6.7M	17.4
de-news	112.8M	2,062.1M	18.0M	18.3

[6] The functionality used to extract these triples is publicly available via a REST webservice described at `http://boa.aksw.org`

4.1 Score Function

We began the evaluation by annotating 200 patterns per corpus by hand. Each training data set was annotated independently by the authors, who agreed on the annotations in approximately 90% of the cases. The annotations upon which the authors disagreed were resolved by both authors. High-quality patterns were assigned a score of 1, else they were assigned a 0. We then trained four different neural networks (one for each dataset) to distinguish between the high-precision and poor patterns. In our experiments, we varied the size of the hidden layer between one and three times the size of the input layer. In addition, we varied the error rate to which they were trained. The maximal number of training epochs was set to 10000. The accuracy of the networks was measured by using a 10-fold cross-validation. Patterns whose score was above 0.5 were considered to be good patterns, while all other were considered to be poor. The best neural network was set to be the smallest network that reaches the maximal accuracy. The resulting learning curves are shown in Figure 2. It is easy to see that networks trained to achieve an error rate of maximally 5% performed best in our experiments.

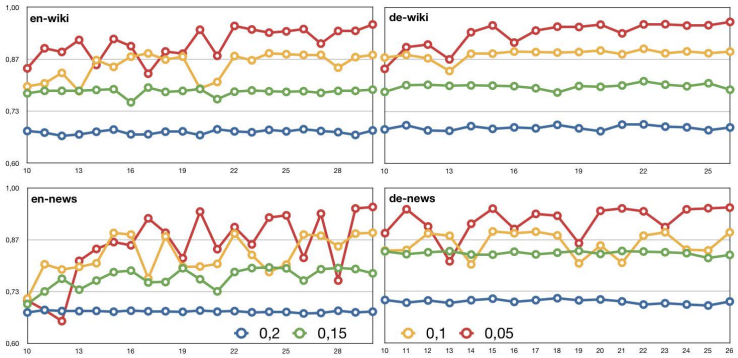


Fig. 2. Learning curves of BOA's neural networks. The x-axis shows the size of the hidden layer while the y-axis shows the accuracy achieved by the neural network. The different colors stand for different maximal error rates.

4.2 Multi-linguality

Enabling BOA to process languages other than English requires solely the alteration of the NER tools and POS taggers. As the results on German show, languages with a wide range of morpho-syntactical variations demand the analysis of considerably larger corpora to enable the detection of meaningful patterns. For example, while we trained the neural network by using the same number of patterns, we were not able to detect any triples with a score above 0.5 when using the German Wikipedia and DBpedia. Yet, when using a larger German Newscorpus data set, we were able to detect new patterns with an acceptable precision (see subsequent section).

4.3 Accuracy

The results of our experiments on accuracy are shown in Table 3. We measured the precision of the extraction carried out by BOA as well as the number of new triples that we were able to extract in one iteration. For the top-100 scored triples we achieved a precision superior to 90% overall on the English data sets. This value is comparable to that achieved by the previous versions of BOA [6]. Yet, the addition of surface forms for the extraction yields the advantage of achieving a considerably higher recall both with respect to the number of patterns extracted as well as with respect to the total number of triples extracted. For example, when using the English Wikipedia, we can extract more than twice the amount of triples. The same holds for the number of patterns and pattern mappings as shown in Table 3.

Table 3. Results of one iteration of the BOA framework

	en-wiki	**de-wiki**	**en-news**	**de-news**
Number of pattern mappings	125	44	66	19
Number of patterns	9551	586	7366	109
Number of new triples	78944	22883	10138	883
Number of known triples	1,829	798	655	42
Number of found triples	80773	3081	10793	925
Precision top-100 triples	92%	70%	91%	74%

An excerpt of the new knowledge extracted by BOA is shown in Listing 3. Note that the triple `Iomega subsidiary ExcelStor_Technology` is wrong. Although Iomega planned to buy ExcelStor, the deal was never concluded. Our approach finds the right patterns in the sentences describing the deal and thus extract this triple.

```
1 Chinese spokenIn Malaysia .
2 Chinese spokenIn  China .
3 Weitnau administrativeDistrict boa:Oberallgäu .
4 Memmingerberg administrativeDistrict boa:Unterallgäu .
5 ESV_Blau-Rot_Bonn ground Bonn .
6 TG_Würzburg ground Würzburg .
7 Intel_Corporation subsidiary McAfee .
8 Iomega subsidiary ExcelStor_Technology .
```

Listing 3. RDF extracted by BOA. If not stated otherwise, all instances and properties use the DBpedia namespace.

5 Conclusion and Future Work

In this paper, we presented BOA, a framework for the extraction of RDF from unstructured data. We presented the components of the BOA framework and applied it to English and German corpora. We showed that in all cases, we

can extract RDF from the data at hand. Our extraction strategy was to only integrate RDF triples that were generated by at least two patterns. By these means we were able to achieve a high precision on all data sets. The precision of German was smaller than that on English because of the rich morphology and syntax of the German language. Overall, the new version of BOA achieves a significantly higher recall by using surface forms to retrieve entities. In future work, we will aim to apply our approach to Asian languages whose grammar differ completely from that of the language we processed so far. In addition, we will consider the use of crowd-sourcing to improve our scoring approach. Finally, we will take temporal markers into consideration so as to be able to process predicates such as `dbpedia:formerTeam`.

References

1. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to Linked Data and Its Lifecycle on the Web. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 1–75. Springer, Heidelberg (2011)
2. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI (2010)
3. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: EMNLP, pp. 1535–1545. ACL (2011)
4. Finkel, J.R., Manning, C.D.: Hierarchical joint learning: improving joint parsing and named entity recognition with non-jointly labeled data. In: ACL 2010, pp. 720–728 (2010)
5. Gaag, A., Kohn, A., Lindemann, U.: Function-based solution retrieval and semantic search in mechanical engineering. In: IDEC 2009, pp. 147–158 (2009)
6. Gerber, D., Ngonga Ngomo, A.-C.: Bootstrapping the linked data web. In: 1st Workshop on Web Scale Knowledge Extraction ISWC (2011)
7. Jiang, J.J., Conrath, D.W.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: ROCLING X, p. 9008 (September 1997)
8. Kim, S.N., Medelyan, O., Kan, M.-Y., Baldwin, T.: Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In: SemEval 2010 (2010)
9. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In: I-SEMANTICS, pp. 1–8. ACM (2011)
10. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: ACL, pp. 1003–1011 (2009)
11. Nakashole, N., Theobald, M., Weikum, G.: Scalable knowledge harvesting with high precision and high recall. In: WSDM, Hong Kong, pp. 227–236 (2011)
12. Ngonga Ngomo, A.-C., Heino, N., Lyko, K., Speck, R., Kaltenböck, M.: SCMS – Semantifying Content Management Systems. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 189–204. Springer, Heidelberg (2011)
13. Seco, N., Veale, T., Hayes, J.: An intrinsic information content metric for semantic similarity in WordNet. In: Proc. of ECAI, vol. 4, pp. 1089–1090 (2004)
14. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., Cimiano, P.: Sparql template-based question answering. In: Proceedings of WWW (2012)

Improving the Performance of a Named Entity Recognition System with Knowledge Acquisition

Myung Hee Kim and Paul Compton

The University of New South Wales, Sydney, NSW, Australia
{mkim978,compton}@cse.unsw.edu.au

Abstract. Named Entity Recognition (NER) is important for extracting information from highly heterogeneous web documents. Most NER systems have been developed based on formal documents, but informal web documents usually contain noise, and incorrect and incomplete expressions. The performance of current NER systems drops dramatically as informality increases in web documents and a different kind of NER is needed. Here we propose a Ripple-Down-Rules-based Named Entity Recognition (RDRNER) system. This is a wrapper around the machine-learning-based Stanford NER system, correcting its output using rules added by people to deal with specific application domains. The key advantages of this approach are that it can handle the freer writing style that occurs in web documents and correct errors introduced by the web's informal characteristics. In these studies the Ripple-Down Rule approach, with low-cost rule addition improved the Stanford NER system's performance on informal web document in a specific domain to the same level as its state-of-the-art performance on formal documents.

Keywords: Ripple-Down Rules, Named Entity Recognition.

1 Introduction

The Web contains a vast amount of information mainly in natural language that has been increasing exponentially. Most Web documents are relatively unstructured, with considerable noise and they change dynamically; therefore it is important to develop tools to manage unstructured data on the Web. A number of Natural Language Processing (NLP) applications have been developed to reduce the amount of time necessary to find the desired information from the Web, including Web Information Extraction (WIE), Automatic Text Summarization (ATS) Information Retrieval (IR) and Question-Answering (QA) systems.

Named Entity Recognition (NER) is one of key tasks in these NLP applications [1, 2]. It automatically identifies proper names in text and classifies them into a set of categories such as persons, geographical locations, names of organizations, dates, times, amounts of money etc. NER has mainly adopted two approaches. One is referred to as knowledge-based using explicit resources like rules and gazetteers, which usually are handcrafted by experienced language experts [4]. This achieves good performance but the development can be very time-consuming. The other

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 97–113, 2012.

approach is learning-based and uses statistics or machine learning. Supervised learning techniques learn automatically on large corpora of annotated text. [7]. While this approach does not need language engineering expertise, it requires large amounts of annotated training data. Such training corpora are available from evaluation forums but there are limitations in the amount of annotated data and coverage of the domain. Recent studies have explored semi-supervised [8] and unsupervised learning techniques [9], which do not require annotated corpora.

Current NER systems are trained mainly on journalistic documents such as news articles. Consequently they have not been trained to deal with the informality of Web documents, resulting in dramatic performance drops on Web documents. For these reasons, some studies comment that NER is a major source of errors for Web Information Extraction (WIE) [10, 11]. Recent WIE systems [11, 15] have avoided the NER process and instead utilized only shallow features like part-of-speech (POS) tags and chunk-phrase tags for entity extraction and extraction pattern generation and then relied on the Web's redundancy to improve accuracy. This approach has limitations for less redundant informal Web documents such as blogs and comment. Thus, developing NER systems for the Web is important for efficient information extraction from informal Web documents.

There are a number of studies on Web-scale NER. One approach relies on large Web resources. The lack of annotated corpora for Web documents and the cost of creating annotated corpora to cover highly heterogeneous Web documents lead to semi-supervised learning (SSL), which relies on very large collections of the Web documents and information redundancy [9]. Another approach develops gazetteers from Web data. Some studies have focused on automatic extraction of known entities from the Web or Wikipedia to build gazetteers appropriate for the web [13, 14].

However, most of these studies focused on the Web's heterogeneity rather than its informality. While machine learning (ML) based approaches are good for scaling domain coverage and achieves state-of-the-art performance, they have critical limitations in interpreting and fixing specific errors as they are discovered. Particularly, it is difficult to overcome errors caused by the Web's informal characteristics. Further, while gazetteers, which were automatically extracted from Web resources, could improve the coverage of Web vocabulary, they contain high levels of noise, which confuses context analysis NER techniques. Riloff et al. [16] demonstrated that when noise is introduced in gazetteers, the performance of an NER system degrades rapidly.

Web-scale NER is a tough challenge due to following characteristics of the Web's informality.

Informal Writing Styles. Huge amounts of Web documents are written informally not following strict writing styles like journalistic text [3]. Many NER techniques rely on title and trigger words. For example, in journalistic texts, person names are generally preceded by titles (Mr., Dr.), organization names are usually followed by trigger words (Inc., Ltd.) and location names are often identified by keywords (mountain, city). As these markers are often absent in Web documents, NER techniques, relying on such indicators, do not work efficiently and cause a significant numbers of errors.

Spelling Mistakes and Incomplete Sentences. Web documents often include spelling mistakes and incomplete sentences, which hinder the syntactic analysis of NER systems and cause extraction errors, since most of the existing NER systems are trained with formal texts with an assumption that the content of texts follows strict writing guidelines.

Large Amount of Newly and Informally Generated Vocabulary. Web documents contain a large number of newly generated unknown words, informal slang and short abbreviations which cannot be found in the formal dictionaries generally utilized by NER systems.

In order to tackle the above challenges, we propose a Ripple-Down Rules based Named Entity Recognition (RDRNER) system. The RDRNER system employs the Stanford NER system as a base system and applies the Ripple-Down Rules technique to deal with the Web's informality. The Ripple-Down Rules technique supports incremental knowledge acquisition (KA) and efficient knowledge base (KB) maintenance. The benefit of the RDR technique is that the KB is built incrementally while the system is already in use, so errors can be corrected whenever they are identified.

The underling idea of the RDRNER system is that it takes state-of-the-art performance from a machine learning technique but then corrects any errors due to the Web's informality. With the RDRNER system, the user creates rules when the classification result provided by the system is incorrect. Since the rules are generated by humans, the RDRNER system is more likely to be able to handle NER errors caused by informally written Web documents. The average F-score (F1) of the Stanford NER [17], trained on the CoNLL03 shared task dataset is 90.8% [18] but dropped to 76.96% on our Web dataset. After 4 hours knowledge acquisition with 200 sentences, the RDRNER system improved this performance to 90.48% similar to the Stanford NER's best performance on formal documents. The RDRNER system described below achieved 92.12% precision and 88.68% recall after training for a Web subdomain and then testing on other sentences from the same subdomain.

It should be noted that Stanford NER system and the RDRNER systems have quite different roles. The Stanford NER system is intended to be applied to any text and. In contrast because the RDRNER system is designed to fix the errors that occur, it is intended be used in specific domains of interest. That is, the user looking to identify entities in a particular domain will write rules for the errors that occur in that domain. If the scope of the application domain expands, the user writes more rules if and when needed. The strength of the approach comes from the ease and speed for which new rules can be added. This may seem a fairly limited solution, but we suggest that in practice organizations generally require technology like NER to deal with specific application domains, so that rapidly developing rules to tune a general purpose system for a particular domain may be a very attractive solution.

Our contributions can be summarized as follows:

- We have developed an RDRNER system that employs Ripple-Down Rules' incremental learning technique as an add-on to the state-of-the-art Stanford NER system in order to handle the problems of the Web's informality.

- We have evaluated the state-of-the-art Stanford NER system on a Web dataset with fair level of Web's informality and categorized error sources that critically degrade performance.
- We have demonstrated how the RDRNER system handles informally written Web documents and improves the performance of the Stanford NER system on informal documents in a restricted domain to be equivalent to its best performance on formal documents

The remainder of this paper is structured as follows. Section 2 presents related work and section 3 formally defines the task and presents the Web's informality challenges. Section 4 explains our RDRNER system in detail, section 5 presents the experimental setting and results and section 6 discusses the results and future work.

2 Related Work

2.1 Web Scale Named Entity Recognition

Since a large hand-annotated corpus is usually expensive and has coverage limitations, semi-supervised or unsupervised learning adopts categorised lists of known entities (called *gazetteers*) with lookup-based methods to recognize named entities [19]. This method needs much less time and labor effort since gazetteers can be generated using automated or semi-automated techniques [9].

As many tagging systems utilise gazetteers, some research has focused on creating gazetteers automatically using web page sources [9] or Wikipedia [13]. While Mikheev et al. [20] have shown that such gazetteers did not necessarily result in increased NER performance, Nadeau et al. [19] used gazetteers in an unsupervised named entity recognizer, and outperformed some other methods on the MUC Named Entity test dataset. Kazama et al. [14] also achieved a 3% F-score improvement using Wikipedia-based gazetteers in their named entity recognizer.

Liu et al. [21] studied NER performance on tweets that contains high level of noise such as excessive informal abbreviations and short hand. They proposed a combination of a K-Nearest Neighbours classifier and Conditional Random Fields and showed the effectiveness of KNN and semi-supervised learning through extensive experiments.

2.2 Ripple-Down Rules (RDR)

The basic idea of RDR is that cases are processed by the knowledge based system and when the output is not correct or missing one or more new rules are created to provide the correct output for that case. The knowledge engineering task in adding rules is simply selecting conditions for the rule. The rule is automatically located in the knowledge base with new rules placed under the default rule node for newly seen cases, and exception rules located under the fired rules. The system also stores cornerstone cases, cases that triggered the creation of new rules. If a new rule is fired by any cornerstone cases, the cornerstones are presented to the expert to select further

differentiating features for the rule or to accept that the new conclusions should apply to the cornerstone. Experience suggests this whole process takes at most a few minutes. A recent study of a large number of RDR knowledge bases used for interpreting diagnostic data in chemical pathology, showed from logs that the median time to add a rule was less than 2 minutes across 57,626 rules [22].

The RDR approach has also been applied to a range of NLP applications. For example, Pham et al. developed KAFTIE, an incremental knowledge acquisition framework to extract positive attributions from scientific papers [23] and temporal relations that outperformed machine learning algorithms [24]. Relevant to the work here, RDR Case Explorer (RDRCE) [25] combined Machine Learning and manual Knowledge Acquisition for NLP problems. It automatically generated an initial RDR tree using transformation-based learning, but then allowed for corrections to be made. They applied RDRCE to POS tagging and achieved a slight improvement over state-of-the-art POS tagging after 60 hours of KA. We have recently demonstrated improved open information extraction using Ripple-Down Rules [26]. In this work we used the Stanford NER system and the limitations we found led to the work described here.

The idea of using an RDR system as a wrapper around a more general system was suggested by work on detecting duplicate invoices where the RDR system was used to clean up false positive duplicates from the general system [28].

3 The Web's Informality and NER

In this section, we illustrate the performance drop of the Stanford NER system on a Web dataset and categorise the type of NER errors that occur due to the Web's informality.

Table 1. The performance of the Stanford NER system on a Web dataset

	Person	Organization	Location	Money	Date	Time	Percent	All
P	90.4%	86.2%	84.4%	97.2%	87.2%	100%	100%	87.1%
R	75.3%	62.5%	81.4%	85.4%	85.0%	100%	66.7%	69.2%
F1	81.8%	72.7%	82.5%	90.6%	86.0%	100%	80.2%	77.0%

Table1 presents the performance of the Stanford NER system on seven NE classes in the Web dataset. The details of the Web dataset are discussed in section 5.1. CONLL evaluation methodologies were used for the experiment. Overall, the Stanford NER system achieved a 77.0% F1 score on the Web data, while it achieved state-of-the-art 90.8% F1 score on the CONLL corpus [18]. This is about 14% performance drop on informal Web documents compared to formal journalistic documents without noise. On average, the Stanford NER system achieved quite reasonable precision but low recall on the informal Web documents. The difference between precision and recall is around 18%.

Table 2. The Stanford NER system's error sources on the Web dataset

New vocabulary	39.53%
Machine learning inconsistency	25.97%
Informal capital letter usage	21.71%
Lack of trigger word	10.08%
Web noise	2.71%

Table 2 presents the causes of error for the Stanford NER system on the Web dataset. Error sources were classified into five categories: new vocabulary, ML inconsistency, informal capital letter usage, lack of trigger word and Web noise.

39.53% of errors were caused by new vocabulary that had not seen during training and were not contained in dictionaries used by the system. As noted, in order to solve this problem, some research has focused on creating gazetteers automatically from the Web [9] or Wikipedia [13] increasing the size of dictionary. Since the size of these Web gazetteers is quite large, most systems take a 'bag of words' approach that can cause confusion as its size increases. Another problem is the uncontrolled noise contained in the Web gazetteers. Liu et al. [21] have shown that the noise contained in the Web gazetteers reduces the NER performance. 25.97% of the errors were caused by Machine-Learning (ML) producing inconsistent annotation. For example, in one short Web sentence, '(**/O** Encyclopedia**/ORG**)**/O** Kafka**/PER**', Kafka was annotated as a PERSON. However, in another short sentence, '(**/O** Almanac**/ORG** –**/O** People**/O**)**/O** Kafka**/LOC**', Kafka was tagged as a LOCATION, although the second sentence has the evidence of the word 'People'. It is difficult to understand why such errors are made. 21.71% of the error is caused by informal Web capital letter usage. On the Web, capital letters are often used informally in order to emphasize the content, but this causes critical errors for current NER systems. For example, in the sentence 'Google**/ORG** Acquires**/ORG** YouTube**/ORG**', because 'Acquires' started with capital letter, the system annotated all three token as a one ORGANIZATION NE tag. 10.08% of errors were caused by lack of trigger words expected such as 'Ltd., Dr. and city'. For instance, in a sentence 'Franz**/PER** Kafka**/PER** –**/O** Prague**/LOC** wax**/O** museum**/O**', 'wax museum' was not tagged as LOCATION NE because museum is not recognised as a sort of trigger word. 2.71% of errors were caused by Web noise such as spelling errors, various symbols, excessive abbreviation and informal short hand. For instance, in a sentence, 'This**/O** morning**/O** Googld**/O** held**/O** a**/O** webcast**/O** and**/O** conference**/O** call**/O** session**/O** with**/O** Eric**/PER** Schmidt**/PER** (**/O** Google**/ORG** CEO**/O**) **/O**', 'Googld' was not tagged due to a spelling error.

4 RDRNER System

The RDR-based Named Entity Recognition (RDRNER) system shown in Figure 1 consists of three main components: preprocessor, Stanford NER system and RDR KB learner. In section 4.1, the implementation details of the three components are explained; the RDR rule syntax is described in section 4.2 and RDR KB construction

demonstrated in section 4.3. RDRNER rule examples to handle the Web's informality are presented in section 4.4 and finally the user interface is shown in section 4.5.

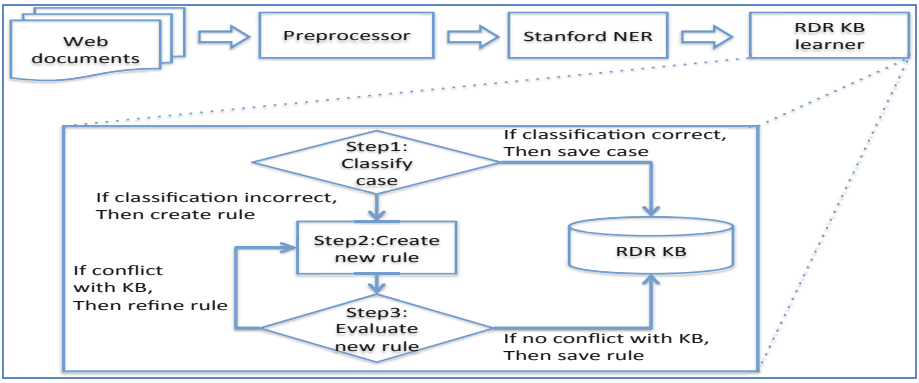


Fig. 1. Architecture of the RDRNER system

4.1 Implementation

Preprocessor. The preprocessor converts raw Web documents into a sequence of sentences, and annotates each token for Part-Of-Speech (POS) and noun and verb phrase chunk using the OpenNLP system. Annotated NLP features are utilized when creating RDR rules.

Stanford NER System. The Stanford NER system was chosen as the base NER system because it is well known as one of the most robust NER systems. It is based on the use of Gibbs sampling for inference in a Conditional Random Field (CRF) machine learning technique [17]. The system models were trained on data from CONLL, MUC6, MUC7, and ACE named-entity corpora. Each type of serialized classifier is provided with one more version that uses Clark's distributional similarity code to improve performance [27]. These versions of the classifier have additional features which provide better performance but require more memory. It achieves a 1.5% F-measure compared to the other versions. We used the classifier 'muc.7class.distsim.crf.ser.gz' that was trained on the MUC corpora and classifies 7 types of NE categories: PERSON, ORGANIZATION, LOCATION, DATE, TIME, MONEY, and PERCENT.

RDR KB Learner. The RDRNER KB is built incrementally while the system is in use with the base NER system. In the RDRNER system, the user gets the NER classification results from the base NER system and adds rules when the classification results are not correct. There are three steps:

Step1: RDRNER classification
The RDR KB takes the given preprocessed sentence and the NER classification results from the Stanford NER system then returns RDRNER classification results. If RDR rules fired, the system replaces the Stanford NER system result with the fired RDR rule's conclusion. Otherwise, the Stanford NER results are given.

Step2: Create RDR rule
Whenever incorrect classification results are given (by the Stanford NER or the RDR KB add-on), the user adds rules to correct the classification results.

Step3: Evaluate and refine RDR rule
Once the new rule is created, the system automatically checks whether the new rule affects KB consistency by evaluating all the previously stored cornerstone cases that may fire the new rule. To assist the expert, the user interface displays not only the rule conditions of previously stored cases but also the features differentiating the current case and any previously stored cases, which also satisfy the new rule but have a different conclusion. The expert must select at least one differentiating feature, unless they decide that new conclusion should apply to the previous case.

4.2 RDRNER's Rule Description

An RDR rule has a condition part and a conclusion part: 'IF (*condition*) THEN (*conclusion*)'. A *condition* consists of three components: (ATTRIBUTE, OPERATOR, VALUE). ATTRIBUTE refers to tokens of the given sentence. Currently the RDRNER system provides 8 types of OPERATOR as follows:

- hasPOS: whether a certain part of speech matches with the attribute token
- hasChunk: whether a certain noun/verb chunk matches with the attribute token
- hasNE: whether a certain named entity matches with the attribute token
- hasGap: skip a certain number of tokens or spaces to match the pattern
- notHasPOS: whether a certain part of speech does not match with the attribute token
- notHasNE: whether a certain named entity does not match with the attribute token
- beforeWD(+a): checks tokens positioned before the given attribute token by +a
- afterWD(+a): checks tokens positioned after the given attribute token by +a

VALUE is derived automatically from the given sentence corresponding to the attribute and operator chosen by the user in the user interface. Conditions may connected with an 'AND' operation. A sequence of conditions using 'SEQ' operator is used to identify a group of words in sequence order, so patterns can be detected. For instance, the sequence condition '*IF SEQ(('Prague' hasNE 'LOC') AND ('wax' hasNE 'O') AND ('museum' hasNE 'O'))*' detects 'Prague**/LOC** wax**/O** museum**/O**'.

Rule's *conclusion* part has the following form:

```
(fixTarget,     // target word to amend NE classification
 fixType,       // amendment type (currently by default 'NE type' used)
 fixFrom,       // incorrect classification to be amended
 fixTo)         // correct classification which is amended
```

4.3 RDR KB Construction

The RDRNER system is based on Multiple Classification RDR (MCRDR) [5]. Figure 2 demonstrates MCRDR KB construction as the RDRNER system processes the following three cases starting with an empty KB.

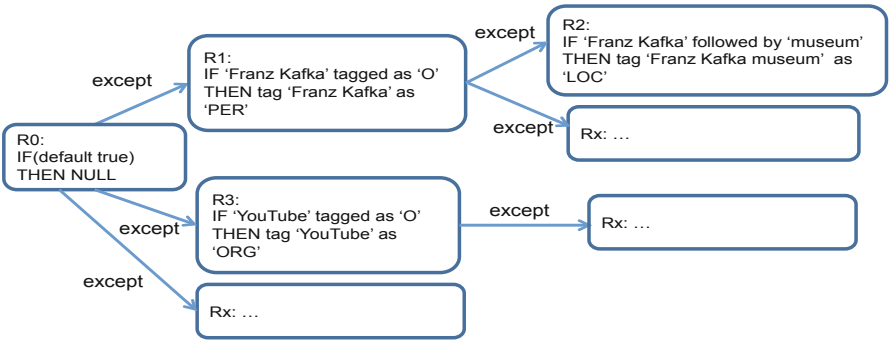


Fig. 2. MCRDR structure of the RDRNER system

Case1: Franz**/O** Kafka**/O** was**/O** born**/O** into**/O** a**/O** Jewish**/O** family**/O** in**/O** Prague**/LOC** in**/O** 1883**/DATE**

1. The default rule R0 fired and the KB system returns a NULL classification so the Stanford NER output is not changed. The user decides this is incorrect as 'Franz Kafka' should be tagged as a PERSON NE.
2. A user adds a new rule R1 under the default rule R0.

Case2: Franz**/O** Kafka**/O** Museum**/O** (**/O** Easy**/ORG** Prague**/ORG** Travel**/ORG** Agency**/ORG** -**/O** Accommodation**/O** and**/O** Sightseeing**/O** in**/O** Prague**/LOC**)**/O**

1. Rule R1 fired but the user decides this is as an incorrect result because 'Franz Kafka Museum' should be tagged as an ORGANIZATION NE instead of tagging 'Franz Kafka' as a PERSON NE.
2. A user adds an exception rule R2 under the parent rule R1.

Case3: Google**/ORG** Buys**/O** YouTube**/O** for**/O** 1.65**/MONEY** Billion**/MONEY** Netscape.com**/O**

1. The default rule R0 fired and the KB system returns a NULL classification, which the user decides is an incorrect result as 'YouTube' is not tagged as an ORGANIZATION NE.
2. A user adds new rule R3 under the default rule R0.

This process continues with a new rule being added whenever the system does not give the correct classification.

4.4 RDR Rules to Handle the Web's Informality

In this section, we show examples of how the RDR rules handle five types of error: new vocabulary, ML inconsistency, informal capital letter usage, lack of trigger words and web noise, which were discussed in section 3.

The following is an example of an error caused by uncovered new vocabulary. The word 'YouTube' was not tagged by the Stanford NER system since it was not contained in the existing dictionary. A simple RDR rule is created to tag 'YouTube' as an ORGANIZATION NE that in effect extends the existing dictionary.

Error source	New vocabulary
Problem Case	Google**/ORG** Buys**/O** YouTube**/O** for**/O** 1.65**/MONEY** Billion**/MONEY** Netscape.com**/O** → NE classification error: 'YouTube' should be tagged as ORGANIZATION
RDR rule	*IF('YouTube' hasNE 'O')* *THEN ('YouTube', 'NE type', 'O', 'ORG')*
Resolved Case	Google**/ORG** Buys**/O** YouTube**/ORG** for**/O** 1.65**/MONEY** Billion**/MONEY** Netscape.com**/O**

The following is an instance of an error caused by ML inconsistency. In case 1, the word 'Kafka' was tagged correctly as PERSON NE but in case 2 it was tagged as LOCATION NE the Stanford NER system without obvious reason. A simple RDR rule is created to annotate 'Kafka' as a PERSON NE that amends the classification derived from the ML-based Stanford NER system.

Error source	ML inconsistency
Problem Case	Case1: (**/O** Encyclopedia**/ORG**)**/O** Kafka**/PER** Case2: (**/O** Almanac**/ORG** –**/O** People**/O**)**/O** Kafka**/LOC** → NE classification error: In case2, 'Kafka' is tagged as 'LOCATION'
RDR rule	*IF ('Kafka' hasNE 'LOC')* *THEN ('Kafka', 'NE type', 'LOC', 'PER')*
Resolved Case	(**/O** Almanac**/ORG** -**/O** People**/O**) **/O** Kafka**/PER**

The following is an example of an error due to informal capital letters. Because the word 'Acquire' started a capital letter, it was treated as part of an ORGANIZATION NE and resulted in an NE boundary error from the Stanford NER system. An RDR rule is created to 'Acquire' as a no NE which resolves the NE boundary error and correctly annotates two ORGANIZATION NE: Google and YouTube.

Error source	Informal capital letter usage
Problem Case	Google**/ORG** Acquire**/ORG** YouTube**/ORG** → NE boundary error: Because 'Acquire' is tagged as 'ORG', 'Google Acquire YouTube' treated as one ORGANIZATION NE tag instead of tagging 'Google' and 'YouTube' as ORGANIZATION separately
RDR rule	*IF ('Acquire' hasNE 'ORG')* *THEN ('Acquire', 'NE type', 'ORG', 'O')*
Resolved Case	Google**/ORG** Acquire**/O** YouTube**/ORG**

The following is an instance of an error caused by lack of trigger words in informal Web documents. Because there are no trigger words available to tag 'Prague wax museum' as one LOCATION NE, only 'Prague' was tagged as a LOCATION NE by the Stanford NER system. Two simple RDR rules are presented to extend the LOCATION NE boundary from 'Prague' to 'museum'.

Error source	Lack of trigger words
Problem Case	Franz**/PER** Kafka**/PER** –**/O** Prague**/LOC** wax**/O** museum**/O** → NE boundary error: 'Prague wax museum' should be tagged as one LOCATION NE
RDR rule	Rule1: *IF SEQ((‘Prague’ hasNE ‘LOC’) AND (‘wax’ hasNE ‘O’) AND (‘museum’ hasNE ‘O’))* *THEN (‘wax’, ‘NE type’, ‘O’, ‘LOC’) AND (‘museum’, ‘NE type’, ‘O’, ‘LOC’)* Rule2: *IF ((‘Prague’ hasNE ‘LOC’) AND (‘Prague’ afterWD(+1) ‘wax’) AND (‘Prague’ afterWD(+2) ‘museum’))* *THEN (‘wax’, ‘NE type’, ‘O’, ‘LOC’) AND (‘museum’, ‘NE type’, ‘O’, ‘LOC’)*
Resolved Case	Franz**/PER** Kafka**/PER** –**/O** Prague**/LOC** wax**/LOC** museum**/LOC**

A more general rule for such a case might be that if a LOCATION NE is followed by the word museum within three words then the sequence is a location. This would implicitly recognise that museum in this context is a trigger word. It is entirely up to the person building rules whether they build specific or more general rules.

The following shows three examples of errors caused by noise. Case 1 shows 'Googld' not tagged due to a spelling error. Rule 1 is generated to tag the 'Googld' as an ORGANIZATION NE when there is the word 'Google' in the same sentence tagged as ORGANIZATION NE. Future conflict caused by this rule could be resolved by adding an exception rule under the rule as explained in section 4.3. Case 2 demonstrates that 'Google' was not tagged due to symbol '+'. Two simple rules (Rule 2 and Rule 3) are created to amend it. Case 3 presents a NE boundary error due to unknown abbreviation 'Bn.'. Rule 4 is generated to fix the boundary error adding the 'Bn.' as part of MONEY NE when there is a $ sign within two tokens before 'Bn.'.

Error source	Web noise (spelling error, various symbols and abbreviations)
Problem Case	Case 1: spelling error This**/O** morning**/O** Googld**/O** held**/O** a**/O** webcast**/O** and**/O** conference**/O** call**/O** session**/O** with**/O** Eric**/PER** Schmidt**/PER** (**/O** Google**/ORG** CEO**/O**) **/O** → NE classification error: 'Googld' tagged as 'O' due to spelling error Case 2: symbols Google**/O** +**/O** YouTube**/O** :**/O** Day**/O** Two**/O** → NE classification error: 'Google' and 'YouTube' should be tagged as ORGANIZATION NE Case 3: abbreviations Google**/ORG** bought**/O** YouTube**/ORG** for**/O** $**/MONEY** 1.6**/MONEY** Bn.**/O** Sweet**/O** → NE boundary error: 'Bn.' Should be tagged as a part of MONEY NE

RDR rule	Rule 1 for case 1: *IF((‘Google’ hasNE ‘ORG’) AND (‘Googld’ hasNE ‘O’))* *THEN (‘Googld’, ‘NE type’, ‘O’, ‘ORG’)* Rule 2 and 3 for case 2: *IF(‘Google’ hasNE ‘O’)* *THEN(‘Google’, ‘NE type’, ‘O’, ‘ORG’)* *IF(‘YouTube’ hasNE ‘O’)* *THEN(‘YouTube’, ‘NE type’, ‘O’, ‘ORG’)* Rule 4 for case 3: *IF((‘Bn.’ beforeWD(+2) ‘$’) AND (‘Bn.’ hasNE ‘O’))* *THEN (‘Bn.’, ‘NE type’, ‘O’, ‘MONEY’)*
Resolved Case	Case 1: This**/O** morning**/O** Googld**/ORG** held**/O** a**/O** webcast**/O** and**/O** conference**/O** call**/O** session**/O** with**/O** Eric**/PER** Schmidt**/PER** (**/O** Google**/ORG** CEO**/O**)**/O** Case 2: Google**/ORG** +**/O** YouTube**/ORG** :**/O** Day**/O** Two**/O** Case 3: Google**/ORG** bought**/O** YouTube**/ORG** for**/O** $**/MONEY** 1.6**/MONEY** Bn.**/MONEY** Sweet**/O**

Again, we do not suggest that these are ideal rules. The user can add whatever rules they like and the RDR approach simply ensures that the rules added do not degrade the performance of the knowledge base to date

4.5 RDRNER User Interface

The RDRNER system provides a graphic interface for creating and adding RDR rules and enabling the KB to be maintained by end-users. Because most of the relevant values are displayed automatically and the system is built based on the normal human process of identifying distinguishing features to justify a different conclusion, a user should be able to manage the system after few hours training. Industrial experience in complex domain supports this [22]. Figure 3 present the RDRNER user interface and the process flow.

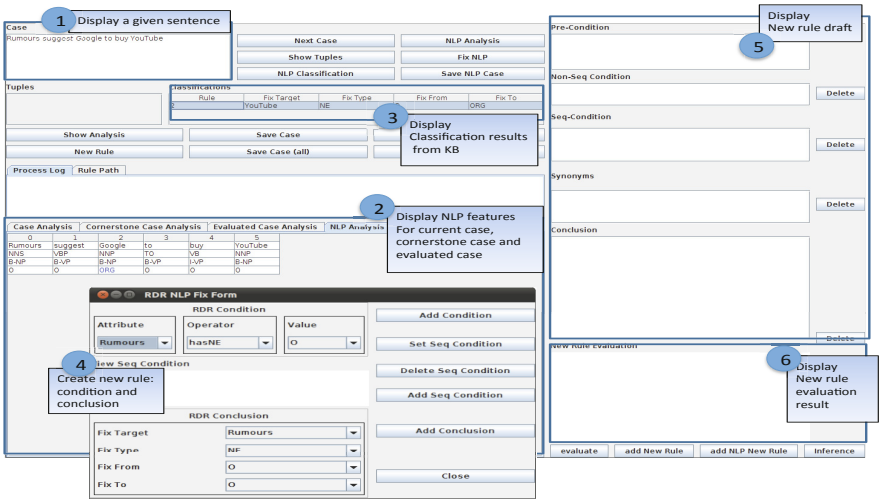


Fig. 3. User Interface of the RDRNER system

5 Experiments

Section 5.1 describes the Web dataset used. Section 5.2 shows the initial knowledge base construction of the RDRNER system and the section 5.3 present the results achieved by the RDRNER system and discusses how the system improved the existing performance of the Stanford NER system on the Web data for this particular domain. The CONLL evaluation methodology was used for experiments. The CONLL evaluation uses an 'exact-match' scoring methodology. That is, it only counts scores for named entity recognition, which satisfies both type classification and boundary detection.

5.1 Web Datasets

Web datasets were prepared from the MIL dataset developed by Bunescu et al. [6]. Bunescu et al. [6] collected a bag of sentences from the Google search engine by submitting a query string 'a1 ******* a2' containing seven wildcard symbols between the given pair of arguments. The pairs of arguments used were 'adobe systems and macromedia', 'google and youtube', 'novartis and eon labs', 'pfizer and rinat neuroscience', 'viacom and dreamworks', 'yahoo and inktomi', 'andre agassi and las vegas', 'chalie chaplin and london', franz kafka and prague', 'george gershwin and new york', 'luc besson and paris', and 'marie antoinette and vienna'. In the MIL dataset, each sentence has one pair of entities manually identified for their intended extraction task, but these entity tags were removed for our NER task experiment. That is, there are no pre-defined tags in our Web dataset. Among 4260 sentences from the positive example folder of the MIL dataset, we randomly selected 200 sentences as a training dataset and 341 different sentences as a test dataset.

The MIL dataset is widely used to evaluate Web Information Extraction (WIE) [11, 15]. We suggest that the MIL dataset represents a reasonable approximation to how an NER system would be used in a specific application. We expect in practice that documents of likely interest would be retrieved because they contained keywords and an NER would then applied to those documents as part of the information extraction process. The MIL dataset provides an approximation of this, while at the same time being an accepted evaluation data set. It should be noted that the MIL sentence selection selects far more entities than just those directly relating to the a1 and a2 tokens. For example the a1 and a2 tokens used included 6 person names but the sentences in the test dataset contained 60 different person names, and so on with the other entity types.

5.2 RDR Initial KB Constructions

In practice, the RDRNER system would be used case by case, but in the experiments here we first tagged the 200 sentences using the Stanford NER system. 231 NE errors were identified and used to develop RDR rules. In processing the error 231 cases and adding rules as required, 44 new rules were created for the cases which received a NULL classification result and 39 exception rules were created for cases which

received an incorrect classification result because of rules added earlier. In total, 83 rules were added within four hours. KB construction time covered from when a case is called up until a rule is accepted and this time is logged automatically. With the RDR approach this time covers all the time spent on knowledge acquisition, and no other time is spent validating or rule debugging outside of this.

We note that the approach may have introduced errors into cases correctly classified by the Stanford NER system. These would have been picked up and corrected with further rules if the normal RDR protocol had been used and cases were processed through both systems one by one. The effect of this is that there may be some errors in the test data which would have been corrected using the normal RDR protocol.

5.3 RDRNER Performance

After the initial KB was built, the 341 sentences of the test dataset were run on the RDRNER system. The test dataset contains 857 NEs including 150 PERSON, 499 ORGANIZATION, 112 LOCATION, 41 MONEY, 40 DATE, 9 PERCENT, 2 TIME.

Table 3. The performance of the RDRNER system on each named entity class

	Person	Organization	Location	Money	Date	Time	Percent	All
P	95.1%	91.5%	86.4%	100%	89.7%	100%	100%	92.1%
R	92.0%	88.6%	84.8%	92.7%	87.5%	100%	88.9%	88.7%
F1	93.5%	90.5%	85.5%	96.4%	89.0%	100%	94.2%	90.5%

Table 3 presents the performance of the RDRNER system on seven NE classes on the test dataset. Overall, the RDRNER system achieved a 90.5% F1 score, while the Stanford NER system achieved a 77.0% F1 score on the same dataset (see table 1). That is, the RDRNER system improved the F1 score by 13.5% compared to the Stanford NER system. The 90.5% F1 score of the RDRNER system is close to the 90.8% F1 score, achieved by the Stanford NER system on the formal CONLL corpus [18]. On average, the RDRNER system achieved both high precision and recall. The difference between precision and recall is around 4%.

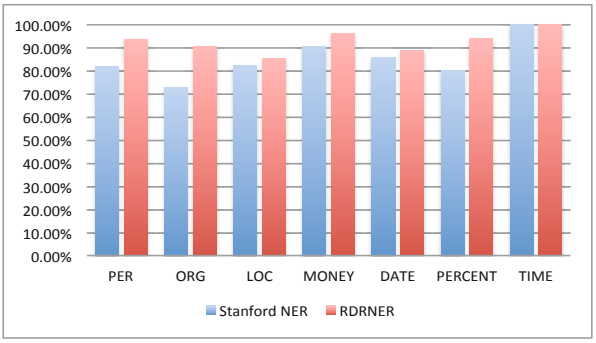


Fig. 4. Performance improvement of the RDRNER system over the Stanford NER system

Figure 4 presents the performance improvement of the RDRNER system over the Stanford NER system on each of the 7 NE classes in F1-score. For all classes the RDRNER system improved the Stanford NER performance except for 'TIME' where no rules were required as the Stanford NER system was already at 100%. ORGANIZATION and PERSON classes had the biggest improvement.

6 Discussion

Table 3, shows that the RDRNER system improves the Stanford system to give a high precision and recall after only 4 hours initial KB construction by a user working through a small training dataset. Given the very rapid training time we propose that rather than simply having to accept an inadequate level of performance delivered by a general purpose NER tool, and the results of this flow on through the whole information extraction task; it is a worthwhile and viable alternative to very rapidly add rules to specifically cover the domain of interests. Such a solution may not be as elegant as an improved general purpose NER, but given the rapid knowledge acquisition time is certainly a practical solution for organisations needing higher NER in a domain of interest. The approach is not specific to the Stanford NER and can be used on top of better NER systems as they become available, and the better the general system the less knowledge acquisition time to tune it to the particular domain.

For NER, the use of lexical features such as dictionary and trigger words is a main technique. As discussed earlier, recent studies have focused on creating gazetteers automatically using Web sources, increasing the size of gazetteers dramatically. With the current 'bag of words' approach, the increased size of gazetteers cannot be utilised efficiently because word sense disambiguation problems. The lack of trigger words on the Web also make difficult to utilise lexical features by pre-defining trigger words.

The RDRNER system, however, can efficiently utilise lexical features for informal Web documents. Section 3 showed examples of the RDRNER system using lexical features in rule creation. One could perhaps characterise some of the rules we added as equivalent to adding words to a gazetteer; however, because rules are used any combination of conditions can be used to provide a much more sophisticated function than adding words to a gazetteer. Secondly, if the vocabulary causes an incorrect classification result with other cases seen later, we can simply add an exception rule to correct the classification, ending up with something much more sophisticated than a gazetteer. Similarly, when there is a lack of pre-defined trigger words in Web documents, the RDRNER system can be used to identify new triggers, but again with a rule structure allowing a conjunction of conditions and exceptions to be used.

The RDRNER system is designed to be trained on a specific domain of interest. It would also be interesting to see if it was possible to extend the domain coverage to use crowd sourcing, whereby large numbers of people on the web might contribute rules, but there would major issues in how to combine such rules.

The RDRNER system required very little effort; rule creation is simple and rapid. In the study here it took about three minutes on average to build a rule. Experience suggests that knowledge acquisition with RDR remains very rapid even for large

knowledge bases [22]. Rules can be updated as errors are uncovered, or when new vocabularies are created, or new meanings are attached to existing terminologies, or new colloquialisms come into use. The alternative is to accumulate enough training data to retrain a system, but this has its own demands. An error can be corrected manually the first time it occurs with an RDR system but with a machine learning system, one needs to keep monitoring cases until sufficient examples of each type of error have been accumulated for the learning system. We suggest it is simpler to correct the error when it first occurs so the user is then monitoring a continuously improving system. When errors are sufficiently infrequent, monitoring can stop.

References

1. Califf, M.E., Mooney, R.J.: Relational Learning of Pattern-Match Rules for Information Extraction. In: ACL 1997 Workshop in Natural Language Learning (1997)
2. Rozenfeld, B., Feldman, R.: Self-supervised relation extraction from the Web. Knowl. Inf. Syst. 17, 17–33 (2008)
3. Collot, M., Belmore, N.: Electronic Language: A New Variety of English. In: Computer-Mediated Communications: Linguistic, Social and Cross-Cultural Perspectives. John Benjamins, Amsterdam/Philadelphia (1996)
4. Rau, L.F.: Extracting Company Names from Text. In: 6th IEEE Conference on Artificial Intelligence Applications. IEEE Computer Society Press, Miami Beach (1991)
5. Kang, B.H., Compton, P., Preston, P.: Multiple Classification Ripple Down Rules: Evaluation and Possibilities. In: 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop (1995)
6. Bunescu, R.C., Mooney, R.J.: Learning to Extract Relations from the Web using Minimal Supervision. In: 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic (2007)
7. Asahara, M., Matsumoto, Y.: Japanese Named Entity Extraction with Redundant Morphological Analysis. In: Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics (2003)
8. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Linguisticae Investigationes 30, 3–26 (2007)
9. Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A.: Unsupervised named-entity extraction from the Web: An experimental study. Artif. Intell. 165, 91–134 (2005)
10. Nguyen, D.P.T., Matsuo, Y., Ishizuka, M.: Relation extraction from wikipedia using subtree mining. In: 22nd National Conference on Artificial Intelligence, vol. 2, pp. 1414–1420. AAAI Press (2007)
11. Zhu, J., Nie, Z., Liu, X., Zhang, B., Wen, J.R.: StatSnowball: a statistical approach to extracting entity relationships. In: 18th International Conference on World Wide Web, pp. 101–110. ACM, Madrid (2009)
12. Zacharias, V.: Development and Verification of Rule Based Systems — A Survey of Developers. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2008. LNCS, vol. 5321, pp. 6–16. Springer, Heidelberg (2008)
13. Toral, A., Muñoz, R.: A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. In: 11th Conference of the European Chapter of the Association for Computational Linguistics, Trento, Italy (2006)

14. Kazama, J.i., Torisawa, K.: ExploitingWikipedia as External Knowledge for Named Entity Recognition. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, Czech Republic (2007)
15. Banko, M., Etzioni, O.: The Tradeoffs Between Open and Traditional Relation Extraction. In: ACL 2008: HLT (2008)
16. Riloff, E., Jones, R.: Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In: 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence (1999)
17. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: The Association for Computer Linguistics (2005)
18. Ratinov, L., Roth, D.: Design Challenges and Misconceptions in Named Entity Recognition. In: CONLL 2009 (2009)
19. Nadeau, D., Turney, P.D., Matwin, S.: Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In: Lamontagne, L., Marchand, M. (eds.) Canadian AI 2006. LNCS (LNAI), vol. 4013, pp. 266–277. Springer, Heidelberg (2006)
20. Mikheev, A., Moens, M., Grover, C.: Named Entity recognition without gazetteers. In: 9th Conference on European Chapter of the Association for Computational Linguistics, pp. 1–8. Association for Computational Linguistics, Bergen (1999)
21. Liu, X., Zhang, S., Wei, F., Zhou, M.: Recognizing Named Entities in Tweets. In: 49th Association for Computational Linguistics, pp. 359–367 (2011)
22. Compton, P., Peters, L., Lavers, T., Kim, Y.S.: Experience with long-term knowledge acquisition. In: 6th International Conference on Knowledge Capture, pp. 49–56. ACM, Banff (2011)
23. Pham, S.B., Hoffmann, A.: Extracting Positive Attributions from Scientific Papers. In: Discovery Science Conference (2004)
24. Pham, S.B., Hoffmann, A.: Efficient Knowledge Acquisition for Extracting Temporal Relations. In: 17th European Conference on Artificial Intelligence, Riva del Garda, Italy (2006)
25. Xu, H., Hoffmann, A.: RDRCE: Combining Machine Learning and Knowledge Acquisition. In: Pacific Rim Knowledge Acquisition Workshop (2010)
26. Kim, M.H., Compton, P., Kim, Y.S.: RDR-based Open IE for the Web Document. In: 6th International Conference on Knowledge Capture, Banff, Alberta, Canada (2011)
27. Clark, A., Tim, I.: Combining Distributional and Morphological Information for Part of Speech Induction. In: 10th Annual Meeting of the European Association for Computational Linguistics (2003)
28. Ho, V.H., Compton, P., Benatallah, B., Vayssiere, J., Menzel, L., Vogler, H.: An incremental knowledge acquisition method for improving duplicate invoices detection. In: Proceedings of the International Conference on Data Engineering, Shanghai, China, pp. 1415–1418 (2009)

Knowledge Extraction Based on Discourse Representation Theory and Linguistic Frames

Valentina Presutti, Francesco Draicchio, and Aldo Gangemi

STLab, ISTC - CNR, Roma, Italy
{valentina.presutti,aldo.gangemi}@cnr.it, draicchi@cs.unibo.it
http://stlab.istc.cnr.it

Abstract. We have implemented a novel approach for robust ontology design from natural language texts by combining Discourse Representation Theory (DRT), linguistic frame semantics, and ontology design patterns. We show that DRT-based frame detection is feasible by conducting a comparative evaluation of our approach and existing tools. Furthermore, we define a mapping between DRT and RDF/OWL for the production of quality linked data and ontologies, and present FRED, an online tool for converting text into internally well-connected and linked-data-ready ontologies in web-service-acceptable time.

Keywords: frame detection, discourse representation theory, robust ontology design, knowledge extraction.

1 Introduction

The problem of knowledge extraction from text is still only partly solved, in particular if we consider it as a means for populating the Semantic Web. Being able to automatically and fastly produce quality linked data and ontologies from an accurate and reasonably complete analysis of natural language text would be a breakthrough: it would enable the development of applications that automatically produce machine-readable information from Web content as soon as it is edited and published by generic Web users, e.g. through a content management system plugin for automatically annotating HTML pages with RDFa on-the-go.

This simple vision is still far from being reached with state of the art tools. Approaches that tackle this task are referred to in the literature as *ontology learning and population* (OL&P) [8]. Ontology learning can be described as "the acquisition of a domain model from data", hence it targets TBox production. Typical ontology learning tasks include concept extraction, relation extraction, and taxonomy induction. Learning the extension of concepts and relations (individuals and facts) is the typical task of ontology population, hence targeting ABox production. OL&P is meant to support ontology engineers in defining the appropriate ontology and filling its knowledge base according to the context given by a corpus of documents covering a certain knowledge domain.

In the context of knowledge management for large organizations, existing OL&P approaches can be used as drafting ontology engineering tools, but they

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 114–129, 2012.

show some limitations when used to produce linked data on the Web: they usually need a training phase, which can take a long time; their output form needs further elaboration to be put in a logical form; they do not exploit full OWL expressivity, since they typically focus on specific aspects e.g. taxonomy creation, disjointness axioms, etc.; in most cases they lack implementation of ontology design good practices; linking to existing linked data vocabularies and datasets is usually left as a separate task or not considered at all. In other words, existing tools focus mainly on the needs of (possibly large) organizations, where users would substantially refine the resulting ontology, as opposed to focusing on producing ontologies and linked data for the Web. The current trend to abridge organizational knowledge (specially from public administrations) with public semantic datasets is an additional motivation for handling OL&P in a way that works well with Semantic Web.

An important aspect of ontology learning is the design quality of the resulting ontology: this is related to representing the results in a logical form such as OWL by ensuring that modeling good practices are followed. Approaches such as [3] show that augmenting the learning cycle with selection and reuse of ontology design patterns, or with the detection of linguistic frames, improves results of existing OL methods.

Based on the above considerations, we summarize a set of requirements for a method that enables robust OL&P, the output of which can be readily published on the Web:

- ability to capture accurate semantic structures, and to produce good quality schemas e.g. representing complex relations;
- exploitation of general purpose resources i.e. no need of large-size domain-specific text corpora and training sessions;
- minimal time of computation;
- ability to map natural language to RDF/OWL representations;
- easiness of adaption to the principles of linked data publishing [15].

We present a novel approach and an online tool, FRED,[1], which performs robust OL&P according to those requirements: FRED performs deep parsing of natural language and extracts complex relations based on Discourse Representation Theory (DRT) [16]. We use Boxer [5] that implements a DRT-compliant deep parser, which saves FRED from the typical training phase of machine-learning-based information extraction tools. We extend Boxer in order to detect the most appropriate linguistic frames [21] capturing complex relations expressed in the input text. This ensures good quality of design of the resulting ontology (cf. [10]). The logical output of Boxer with frames is transformed into RDF/OWL by means of a mapping model and a set of heuristics that follow good practices of OWL ontologies and RDF data design, e.g. we avoid blank nodes and create domain-oriented relation names. The produced RDF complies with linked data principles as we reuse existing vocabularies when possible, resolve named entities over resources existing in RDF datasets of the linked data cloud (LOD)[2], and

[1] FRED demonstrator is available at `http://wit.istc.cnr.it/fred`

[2] Named entity resolution relies on an external system, as described in Section 5.

disambiguate domain terminology against WordNet and foundational ontologies. We are able to prove that our method has high potential for being the enabler of robust knowledge extraction, since it guarantees good quality of design, and fast computational performance.

The paper is structured as follows: Section 2 discusses related work. Section 3 shows a new approach to frame detection and its evaluation, Section 4 discusses a set of heuristics that we have defined for transforming a DRT-based logical form into a well designed RDF/OWL ontology, and Section 5 describes FRED, the system that implements the overall method. In Section 6 we briefly conclude and discuss future work.

2 Related Work

OL&P is concerned with the (semi-)automatic generation of ontologies from textual data (cf. [8]). Typical approaches to OL&P are implemented on top of Natural Language Processing (NLP) techniques, mostly machine learning methods, hence they require large corpora, sometimes manually annotated, in order to induce a set of probabilistic rules. Such rules are defined through a training phase that can take long time. OL&P systems are usually focused on either ontology learning (OL) for TBox production, or ontology population (OP) for ABox production.

Examples of OL systems include [17], which describes an ontology-learning framework that extends typical ontology engineering environments by using semiautomatic ontology-construction tools, and Text2Onto [9], which generates a class taxonomy and additional axioms from textual documents.

Examples of OP systems include: [23], which presents a GATE plugin that supports the development of OP systems and allows to populate ontologies with domain knowledge as well as NLP-based knowledge; [22] describes a weakly supervised approach that populate an ontology of locations and persons with named entities; [18] introduces a sub-task of OP restricted to textual mentions, and describes challenging aspects related to named entities. More complete reviews of state-of-art research methods and tools for OL&P are given in [8,24,13].

The method and tool (FRED) that we present in this paper differs from most existing approaches, because it does not rely on machine learning methods. Instead, it is based on a logical interpretation of natural language given by Discourse Representation Theory, a formal theory of linguistic semantics originally designed by Hans Kamp to cope both with linguistic phenomena – such as donkey sentences, anaphoric resolution, ellipsis and presupposition – and temporal relations [16]. Another distinguishing component of our approach is frame-based ontology design combined with a set of mapping and heuristic rules. We argue that automatic frame-based ontology design simulates the typical approach of ontology engineers when using textual documents as requirements for the ontology to be designed. Our method addresses the overall conceptualization expressed by a document, instead of only focusing on specific tasks such as named entity recognition or taxonomy induction.

We use Boxer [5] as an implementation of DRT, and we prove, by comparing it to Semafor [7], that it can be used for detecting frames with good performance, especially in terms of computational time. The choice of performing frame-based ontology design is based on the evidence given in [3] that OL methods performances improve if the learning cycle is augmented with ontology design patterns, and on the work by [10,19] that ontology design patterns can be easily derived from frames. Additional related work on using frame semantics when integrating structured knowledge in NLP can be found in [20]. A notable work, which is also an inspiration for FRED, is AURA [6], which uses a library of frame-like knowledge engineering components (CLib) for automatic formalization of specialized texts into the KM language.

The core of the work on FRED presented in this paper was originally developed as a master's thesis [12]. Approximately in the same period, an approach similar to ours, named LODifier [1], has been developed. Unfortunately, we could not compare the performances of our system to LODifier's because the latter is not available at this time. Nevertheless, from the paper we could understand the main differences. LODifier reuses Boxer, however it does not exploit it as a frame-detector and does not follow a frame-based approach to ontology design. Their basic conversion table from DRT to RDF is similar to FRED's, but the result of Boxer is serialized to RDF without applying specific rules to maximizing design choices according to Semantic Web and OWL ontology design principles (e.g. it produces blank nodes for all variables and DRSs, does not consider terminological issues, does not try to reduce redundant structures, etc.). In our opinion, although it is in general recommended to publish linguistic linked data without changing the original data structure significantly, in the case of Boxer, we are not publishing established lexical data, but extracted knowledge in logical form that is not targeted at linguists, therefore there is no advantage in preserving the original data structure. LODifier makes a smart usage of NER and WSD (Word Sense Disambiguation) to WordNet to provide better linked data. For NER, we use Stanbol enhancers[3], and for WSD we have reused UKB[4] in a similar way as LODifier does, but also adding top-level mappings to DOLCE+DnS foundational ontology and to WordNet "super senses" (see Section 5). Our frame-based tools can be tested online[5].

3 DRT-Based Frame Detection

Robust OL&P requires special attention to design quality. [3] proves that augmenting the learning cycle with ontology design patterns improves existing OL&P tool performances in terms of ontology enrichment. [10] shows that detecting the most appropriate frames from the input text leads to improving the design quality of the resulting ontology because frames can be directly mapped to an

[3] Cf. http://incubator.apache.org/stanbol/docs/trunk/enhancer/

[4] http://ixa2.si.ehu.es/ukb/

[5] FRED is cited; for NER and WSD over FRED see our Wikipedia typer *Tìpalo* at http://wit.istc.cnr.it/tipalo

important variety of ontology design patterns (cf. [19]) based on *n-ary relations*, the most critical logical form used in domain ontologies. We take these results as assumptions in our work, and design our workflow so that it includes a frame detection step that we use as a means for selecting ontology modeling choices.

Frames. Frame Semantics [14] is a formal theory of meaning: its basic idea is that humans can better understand the meaning of a single word by knowing the contextual knowledge related to that word. For instance, the sense of the word *buy* is clarified by knowing about the context of a commercial transfer that involves certain individuals, e.g. a seller, a buyer, goods, money, etc. Linguistic frames are referred to by sentences in text that describe different situations of the same type i.e. frame occurrences. The words in a sentence "evoke" concepts as well as the perspective from which the situation expressed is viewed.

In the previous example, the word *sell* evokes a situation from the perspective of the seller, and the word *buy* evokes it from the perspective of the buyer. This fact explains the observed asymmetries in many lexical relations that need a particular design involving roles in order to be represented. Frame semantics allows real-world knowledge to be captured by semantic frames, which describe particular types of situations, objects or events, and their participants characterized by specific semantic roles. FrameNet[21] is a lexical resource that collects linguistic frames, each described with its semantic roles, called frame elements, and lexical units (the words evoking a frame).

Frame Detection. The frame detection (or frame recognition) task [10] has the goal of recognizing complex relations in natural language text. There are a number of systems that perform frame detection with reasonable performances, however they all require a training phase and the availability of a large annotated corpus. One of our goals is to realize a system that can be used also in interactive applications, in other words it has to be as fast and simple as possible.

A problem of machine learning-based systems is that their output needs significant intervention in order to be transformed into a logical form. This is an issue in our case, as we want to reuse the frame structure (e.g. frame roles) in order to reflect it in the ontology we produce. That is why the detection task per-se addresses only partially our requirements.

In summary, we need to answer the following questions:

1. How can we map natural language to a logical form?
2. How can we perform frame detection without ad-hoc training?

3.1 Discourse Representation Theory and Boxer

The answer to the first question is "Discourse Representation Theory" (DRT). DRT is a formal theory of meaning originally described in [16], and is equivalent to first-order logic (FOL). DRT uses an explicit semantic structured language called Discourse Representation Structure (DRS): standard representations corresponding to natural language sentences, which constitute the core of DRT languages. The flavour of DRT we are interested in provides an event-based,

Neo-Davidsonian (based on reified n-ary relations just as frames are) model to represent natural language.

Boxer [5], an implementation of compositional semantics of language that produces a DRS output, is especially suited to our task. It is open-source software that performs deep parsing of natural language: it uses Combinatory Categorial Grammar (CCG) and produces event-based, verb-centric, semantic representations of natural language complying with DRT semantics. These representations are expressed in the form of DRS using the VerbNet[6] inventory of thematic roles. Boxer provides us with an important component of our workflow: it produces, without any additional training, a logical representation of natural language text. The fact that Boxer exploits VerbNet is helps answering our second question: since VerbNet is linked to FrameNet [21], we exploit such mappings in order to perform frame-detection without any training.

3.2 Using Boxer as a Frame Detection System

All frame detection systems address their task with a probabilistic approach. The formal structure of a FrameNet frame (defined at frame-creation time) can be defined as its proper semantic footprint [21]. Based on this observation, we had the intuition that a frame can be detected with a different approach from the probabilistic one: a rule-based approach whose output can be compared with the syntactic and semantic structure of frames, i.e. their typical syntactic manifestation in language, and with the involved roles that characterize them, in order to identify the best frame candidates.

This intuition can be experimented by using Boxer with a different purpose than its usual one, and the potential of the approach can be evaluated by comparing its performances against Semafor [11], to our knowledge the best performing tool for frame detection so far.

Boxer exploits VerbNet in order to identify the roles involved in a sentence, and roles in turn are used to detect a corresponding frame. However, as it is not developed with this task in mind, its coverage with respect to FrameNet frames is limited. Hence, we have adapted Boxer for tackling the frame detection task by integrating it with a resource that provides the most complete mapping between VerbNet and FrameNet.[7].

Evaluation. In order to evaluate Boxer as a frame detection tool, we compare its performances against Semafor's for *Task 19* of *Semeval'07* [2], defined as a *frame recognition task*: given a textual sentence, the system has to automatically extract facts from it, and predict FrameNet frame structures that best fit those facts. Although Semeval provides a set of benchmarks for evaluating the results of the tests, we could not use them because Semafor has been trained on their annotated corpus after the challenge.

[6] VerbNet, http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

[7] http://verbs.colorado.edu/~mpalmer/projects/verbnet/downloads.html

Table 1. Performances for frame recognition task

Tool	Precision	Recall	F-Score	Coverage	Elapsed Time
Boxer	69.693	53.223	60.354	76.367	2m 45.21s
SEMAFOR	72.370	71.875	72.122	99.316	20m 14.27s

To address this problem, we have built a new benchmark[8] that is based on the FrameNet-annotated corpus of sample frame sentences. The dataset consists of 1214 sentences, each fully annotated with at least one frame. The benchmark is automatically generated by randomly selecting dataset sentences among the whole set of annotated samples so as to keep the same data distribution. The evaluation is performed in terms of precision, recall, coverage, and time-efficiency. We have built an evaluation component by implementing the same criteria used in the Semeval task evaluation. Such component is used for comparing the results of the two systems against the gold-standard file provided by the benchmark. Given a sentence, when a predicted frame matches exactly the one indicated by the gold-standard, we assign a full score (1). Additionally, by following criteria defined in [3], when a mismatch occurs, we assign a partial score (0.8) if the predicted frame is conceptually close, or semantically related to the one indicated by the gold-standard. For the sake of this evaluation we have used Semafor for frame detection only from verbs, because Boxer currently identifies frames expressed by verbs. While this is surely a gap in frame detection, consider that the explicit roles for non-verbal frames are usually very limited.

Table 1 shows the results for the frame recognition task performed by Boxer and Semafor by considering only exact match (full scores). Precision, recall, f-score and coverage values are expressed in percentage, while the computation time is expressed in minutes and seconds. From the results, we notice that Boxer is much faster than Semafor, however the latter still performs better than Boxer in terms of accuracy, although precision values are comparable. We claim that this is a good result as the goal of our evaluation is to understand the feasibility of the rule-based approach for frame detection. Our focus has been so far the adaptation of Boxer to the frame detection task (by enriching its reference knowledge base through the integration of Semlink mappings), hence we are aware that there is room for improving the detection algorithm. We leave this issue to future work that includes the implementation of a frame disambiguation method; at the moment Boxer selects the first candidate among a list of frames that it identifies as possible targets without performing any ranking.

While precision values are comparable, we observe a significant gap in recall, which is mainly due to the difference in coverage of the two considered systems. While Boxer covers around 76% of answers, Semafor scores 99%. This difference can be further analyzed by considering performance with partially correct answers (partial scores), reported in Table 2.

[8] The benchmark is available online at `http://tinyurl.com/fd-benchmark`

Table 2. Performances for frame recognition task taking into account partially correct results

Tool	Precision	Recall	F-Score
Boxer	75.320	57.519	65.227
Semafor	75.325	74.797	75.060

In this case, we notice a substantial increase of Boxer performances over all the evaluation parameters, while Semafor shows a less significant improvement. An important result of this evaluation is that a rule-based approach can easily reach performances that are comparable to those of the best frame detection tool currently available based on a probabilistic approach. We claim that these results are promising, they demonstrate the high potential of our method, especially if we consider that the detection algorithm can be further improved by enhancing it for frame disambiguation, and by further extending the set of Semlink mappings between VerbNet and FrameNet.

A number of additional consideration can be made for supporting our claim: Boxer frame detection system is indirectly related to FrameNet, hence it currently exploits FrameNet knowledge only partially. As future work, we aim at improving this aspect by exploiting the FrameNet-LOD datasets [19], which encodes the full knowledge of FrameNet including semantic relations between frames.

Probabilistic models such as those used by Semafor encode most of the knowledge expressed by FrameNet as a consequence of being trained over a large portion of FrameNet data. In other words, the scope of information used for building the inductive models in Semafor is almost complete with respect to FrameNet knowledge. This is the main motivation for the reduced coverage of Boxer, and therefore also of the high amount of partially correct answers. We expect that by increasing Boxer's coverage, its accuracy will increase as well. Two further important aspects are: time taken to compute predictions and output form.

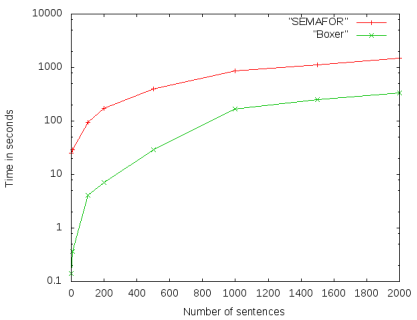


Fig. 1. Time taken to provide answers in function of the number of sentences per document

Figure 1 shows that Boxer is much faster than Semafor, as also reported in Table 1. This is probably due to the fact that Semafor algorithm has a very expensive complexity: it also requires significant resources (8 GB RAM and CPU cycles). As far as the output form is concerned, Semafor does not provide a logical representation, which is one of our core requirements.

Finally, we remark that the benchmark used here put Semafor in a condition of advantage with respect to Boxer. The benchmark is built on FrameNet sample sentences, and Semafor is trained over FrameNet full text annotations. If we would use Semafor on a corpus independent on FrameNet, it would need a proper training, while Boxer could be directly executed without any preparing activity. These characteristics, and the good performances of Boxer support our claim of its suitability for performing frame detection on any Web content item, and for being employed in the context of interactive Semantic Web applications.

4 Transforming DRT Forms to OWL/RDF Ontologies

A key step in our OL&P approach is performing good quality ontology design. We assume, supported by [10,3], that a frame-based approach to design helps quality assurance. Intuitively, a frame provides a means for representing knowledge boundaries, which is a desideratum, since it allows to associate a context to data in a knowledge base. In this work, we consider frames identified by verbs, which is also the case in most modeling situations. For example, if an ontology engineer needs to store in a knowledge base the knowledge expressed by the following sentence:

> The statement by China Foreign Ministry on Friday signaled a possible breakthrough in a diplomatic crisis that has threatened American relations with Beijing.[9]

she would probably model at least two situations (or events): one expressed by the verb *signal*, the other expressed by the verb *threaten*. The two situations create boundaries for the *statement*, its content, and the *Ministry*, and link them to *diplomatic crisis*, *America*, and *Beijing*, which in turn are kept together by the *threaten* situation.

Thanks to the frame-based approach integrated in our method, as described in Section 3, we can automatically design an ontology by following this good modeling practice. However, although Boxer gives us a logical form, its constructs – syntactically, lexically, and semantically – differ from RDF or OWL ones, and the heuristics that it implements for interpreting a natural language and transforming it to a DRT-based structure can be sometimes awkward when directly translated to ontologies for the Semantic Web.

[9] Taken from the New York Times, May 4th 2012, http://www.blogrunner.com/snapshot/D/7/2/chen_guangcheng_can_apply_to_study_abroad_china_says/

Table 3. The main translation rules from DRS to OWL

DRT construct	Boxer syntax	FOL construct	OWL construct
Predicate	pred(x)	Unary predicate ϕ	`rdf:type`
Relation	rel-name(x,y)	Binary relation	`owl:ObjectProperty`
Eq Rel	eq(x,y)	Identity	`owl:sameAs`
Named Entity	named(<var>, <name>, <type>)	Unary predicate ϕ	`owl:NamedIndividual`
Discourse Referent	(<var>)	Quantified Variable	(generated) `owl:NamedIndividual`
DRS	<drs> with event E	Proposition P with predicate ϕ_E	RDF graph G_P with class `E`
Negated DRS	not(<drs>)	Negated Proposition $\neg P$	G_P with `NotE owl:disjointWith E`

Table 4. Boxer built-in types and relations

Boxer built-in type	Label	Semantic Web entity
Per	Person	foaf:Person
Org	Organisation	foaf:Organisation
Loc	Location	dbpedia:Place
Tim	Time	to:Interval
Ttl	Title	dul:Role
Event	Event	dul:Event
Eq	Equal to	owl:sameAs

In this section we show, through a set of examples, the rules that we have defined in order to transform Boxer output to an OWL/RDF ontology. We can distinguish two types of rules: (i) translation rules, which define global transformations from DRS constructs to OWL constructs, and (ii) heuristic rules, which define local transformations that deal with adapting the results of Boxer heuristics to the needs of a Semantic Web ontology.

Translation Rules. DRT is basically FOL (although Boxer uses a subset of it), hence the first step is to define a set of global translation rules that allow us to transform DRS constructs into OWL/RDF constructs (except when overruled by local rules, see below). Table 3[10] indicate DRT constructs, their syntax in Boxer, their corresponding FOL construct, and their corresponding OWL/RDF construct. Additionally, Boxer has a set of built-in predicates. Those most frequently used are listed in Table 4,[11] each associated with a Semantic Web entity, to which we align it by default. We have represented all Boxer built-in types and relations in a publicly available ontology[12]. Finally, the semantic roles that are applied to the frames detected in sentences can belong to three different

[10] NotE and E are generated classes for the event E contained in a (simple) DRS. The prefixes map to the following namespaces: rdf: `http://www.w3.org/1999/02/22-rdf-syntax-ns`, owl: `http://www.w3.org/2002/07/owl`

[11] With prefixes: foaf: `http://xmlns.com/foaf/0.1/`; dbpedia: `http://www.dbpedia.org/ontology/`, dul: `http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#`, to: `http://www.w3.org/2006/time#`

[12] Boxer types and relation ontology, `http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl`

vocabularies: the default one is VerbNet, the second is FrameNet, the third is a domain relation that cannot be resolved to any of VerbNet or FrameNet roles.[13]

For example, the sentence "Paul Newman hit the window with an open hand" is transformed by Boxer in the "boxed" form shown in Example 1. Each box represents a DRS: at the top it shows its variables, at the bottom its formal sentences. Boxer recognizes the roles `agent`, `patient`, and `instrument` and uses them to link entities to the event `x2` denoted by the verb *hit*, which expresses the situation type `hit` occurring in the sentence. It recognizes that the agent `x0` is a named entity `paul_newman` and that he is a person (`per`); that the patient `x1` is a `window` and the instrument `x3` is a `hand` that is also `open` (by co-reference of the variable). All discourse referents (variables) here are existentially quantified: this interpretation always holds in DRT, except when a variable is in the antecedent DRS of an implication, so bearing a universal quantification.

```
Example 1
    %%%   -------------------------   -----------------
    %%%  |x0 x1                    | |x2 x3            |
    %%%  |.........................| |.................|
    %%% (|named(x0,paul_newman,per)|A|hit(x2)          |)
    %%%  |window(x1)               | |Agent(x2,x0)     |
    %%%  |_________________________| |Patient(x2,x1)   |
    %%%                             |open(x3)         |
    %%%                             |hand(x3)         |
    %%%                             |Instrument(x2,x3)|
    %%%                             |_________________|
```

Based on the mappings reported in Table 3 and Table 4, our tool FRED (that implements our method and is described in section 5) generates RDF code that produces e.g. the graph depicted in Figure 2. The resulting ontology defines the class `Hit`, which is a situation type i.e. a frame. Such situation type is instantiated by a named individual `hit_1`. The situation (frame occurrence) `hit_1` involves: (i) `PaulNewman`, an instance of `foaf:Person` having the role of `vn:agent` in the situation; (ii) an instance of `OpenHand` having the role of `vn:instrument` and an attribute `Open`; and (iii) an instance of `Window` having the role of `vn:patient`.

Most of the generated RDF works as from the translation rules shown in Table 3, but although this example is fairly simple, it makes it emerge more structure than those rules can generate. In fact, in order to design a proper OWL ontology, we need more design-oriented rules. The reason for additional rules lies in the way Boxer applies a "flat" FOL modeling style to natural language in order to produce a DRT-based logical form. FOL modeling style is not always compatible or appropriate to Semantic Web and Linked Data design. We cannot exemplify all additional heuristic rules implemented by FRED, however we provide some sample cases that demonstrate our approach.

Heuristic Rules. In the simple example of Figure 2, we notice some non-trivial names and axioms, which cannot be derived from translation rules. For example, `open(x3)` and `hand(x3)` are only co-referential in Boxer, but do not form a unique term. A heuristical rule fires here based on the co-reference, and

[13] With prefixes: vn: `http://www.ontologydesignpatterns.org/ont/vn/abox/role/`; fn: `http://www.ontologydesignpatterns.org/ont/framenet/abox/fe/`; domain: <namespace chosen by the user>.

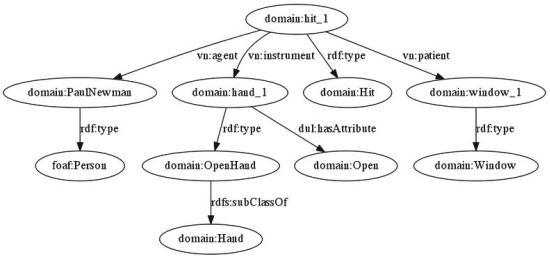


Fig. 2. FRED RDF graph for the sentence "Paul Newman hit the window with an open hand"

generates a new term for the class `OpenHand`. Moreover, an additional class `Hand` is created by FRED, and defined as super-class of `OpenHand`: this heuristic rule defines a default behavior when a branching structure is met in the text, in English this is a *left branching* (or "head-final") construction. Finally, the predicate `open` denoted by the adjective *open* is used by FRED as an attribute of `x3`.

Another heuristic rule has to do with naming. FRED uses the CamelCase convention as it is pretty popular on the Semantic Web. For example, properties start with lower case, while classes and individuals start with capital case.

An important issue is constituted by blank nodes, which are not desirable in linked data, but should be produced out of Boxer's variables. FRED implements a heuristics that creates individuals with a generated name to existentially quantified variables that are not resolved as named entities. In the sample graph from Figure 2, `hit_1` and `window_1` are such individuals.

Other design heuristics have to do with the generation of terminology associated with the definition of appropriate classes or properties. Let's consider the more complex sentence "At the meeting of European Union leaders, Germany leader Angela Merkel was facing Mario Monti, who forced the Iron Chancelor to blink.", in this case Boxer produces the result shown in Example 2.

```
Example 2
%%   ------------------------------   --------------------------
%%  |x0 x1 x2 x3 x4                | |x5 x6 x7 x8               |
%%  |------------------------------| |--------------------------|
%% (|named(x0,angela_merkel,loc)   |A|force(x5)                 |)
%%  |named(x0,germany,loc)         | |face(x7)                  |
%%  |leader(x0)                    | |agent(x5,x1)              |
%%  |named(x1,mario_monti,per)     | |theme(x5,x6)              |
%%  |tenacious(x1)                 | |     -----------------    |
%%  |opponent(x1)                  | |    |x9               |   |
%%  |named(x2,iron_chancellor,org)| | x6:|-----------------|   |
%%  |summit(x3)                    | |    |Body_movement(x9)|   |
%%  |meeting(x3)                   | |    |Agent(x9,x2)     |   |
%%  |named(x4,brussels,loc)        | |    |-----------------|   |
%%  |------------------------------| |finally(x5)               |
%%                                  |agent(x7,x0)              |
%%                                  |patient(x7,x1)            |
%%                                  |named(x8,european_union,org)|
%%                                  |leader(x8)                |
%%                                  |of(x3,x8)                 |
%%                                  |in(x3,x4)                 |
%%                                  |at(x7,x3)                 |
%%                                  |--------------------------|
```

On its turn, FRED produces the ontology depicted in Figure 3, in this case using FrameNet frames and roles. The power of the design heuristics is even more evident here. The preposition *of* is used for generating the property `domain:leaderOf` between instances of `Leader` (e.g. `AngelaMerkel`) and of `Organization` (e.g. `EuropeanUnion`). The situation `blink_1` with frame `fn:BodyMovement` is an argument to another situation `force_1` from frame `fn:ConfrontingProblem`. The definition of classes and properties guided by the occurrence of specific lexico-syntactic patterns is the subject of a number of FRED heuristics, all designed by respecting the requirement of preserving the frame-like structure of the designed ontology.

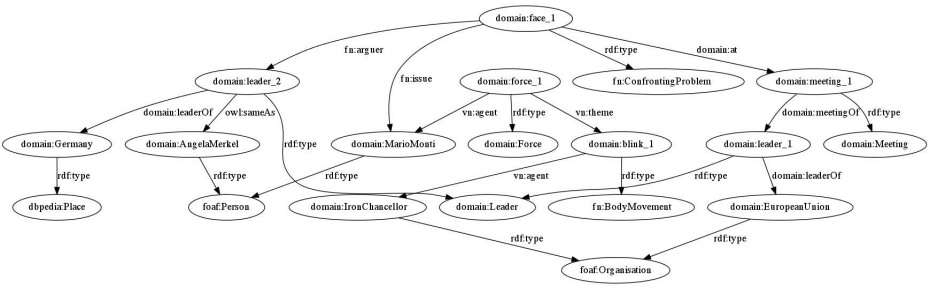


Fig. 3. FRED RDF graph for the sentence "At the meeting of European Union leaders, Germany leader Angela Merkel was facing Mario Monti, who forced the Iron Chancellor to blink"

5 Prototype

Our method is implemented in a tool named FRED, which is accessible online.[14] Figure 4 shows FRED's architecture. FRED is designed in order to be deployed as a web service, hence one important goal it has to address is minimizing computing time. In addition it implements a modular, highly interoperable and customizable architecture in order to ensure reusability by other applications, and extensibility.

The framework is constituted by four main components: (i) *Boxer* (implemented in Prolog) performs deep parsing of natural language text including frame-detection, and provides an output a DRS output; (ii) the *communication* component realizes a lightweight HTTP server based on RESTful architecture. This component is in charge of publicly exposing APIs for querying the system. It takes a language text and some optional parameters as input, and returns an OWL/RDF ontology; (iii) The *refactoring* component transforms Boxer output in a form to be passed to the re-engineering component, which is responsible of implementing the semantic transformation from DRT to OWL; (iv) the *re-engineering* component implements all translation and heuristic rules described in Section 4. The last three components are implemented in Python.

[14] `http://wit.istc.cnr.it/fred`

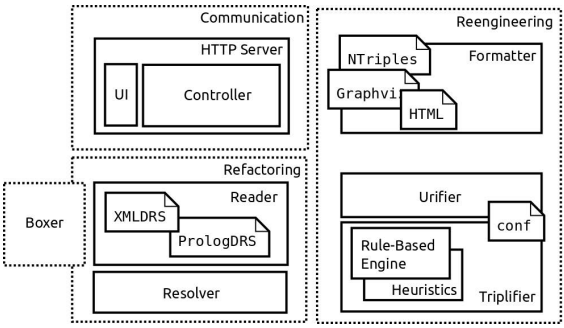


Fig. 4. FRED software architecture

In addition to such components, FRED exploits external services performing entity resolution on linked data, and word sense disambiguation. The NER component is named "Enhancer" and is part of the Apache incubating project "Stanbol"[15]. Stanbol allows to indicate any number of datasets to be used as sources for entity recognition and resolution, hence providing great flexibility and customizability with respect to the entities that are of interest for a FRED user. WSD is performed by means of the UKB tool[16], by aligning domain classes to WordNet synsets, and the synsets to DOLCE+DnS foundational ontology classes[17] and WordNet lexnames ("super-senses")[18]. Additional linking is provided for special purposes, for example the "Tìpalo" service[19] employs FRED and external services to automatically type the entities referred by Wikipedia pages.

6 Conclusion and Future Work

We have presented a novel approach and a tool, FRED, for ontology learning and population in the Semantic Web. It is based on DRT and frame-based ontology design. In order to demonstrate its potential, we have extended one of its core components, Boxer, to perform frame recognition, evaluating its performances against the state-of-art tool, Semafor. Results are promising: this new approach to frame detection shows better time computational performance than existing tools for frame detection, and the ontologies produced are internally well-connected and linked-data-ready. The real payoff of having an RDF and OWL representation of texts is in fact the ability to quickly and accurately process relevant texts for the population of the Semantic Web. FRED is able to semantically map DRT logical forms (DRS) to RDF and OWL by exploiting

[15] http://incubator.apache.org/stanbol/
[16] http://ixa2.si.ehu.es/ukb/
[17] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[18] http://wordnet.princeton.edu/man/lexnames.5WN.html
[19] http://wit.istc.cnr.it/tipalo

frame-based design, and by implementing a set of heuristics that address terminology and structure generation according to Semantic Web design practices.

Future work includes the improvement of the frame detection algorithm through e.g. extending Boxer coverage of FrameNet frames, and by addressing frame disambiguation. Additionally, we are performing an extensive evaluation of the overall workflow in terms of computing time and design quality of the produced ontologies. To this purpose, we have defined a user-based evaluation that involves expert ontology engineers in evaluating FRED on a set of preselected texts. The evaluation is conducted by adopting the quality measures and experimental settings that we have defined and executed in [4].

References

1. Augenstein, I., Padó, S., Rudolph, S.: LODifier: Generating Linked Data from Unstructured Text. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 210–224. Springer, Heidelberg (2012)
2. Baker, C.F., Ellsworth, M., Erk, K.: Semeval 2007 task 19: frame semantic structure extraction. In: Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval 2007, pp. 99–104. ACL (2007)
3. Blomqvist, E.: OntoCase-Automatic Ontology Enrichment Based on Ontology Design Patterns. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 65–80. Springer, Heidelberg (2009)
4. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with eXtreme Design. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 120–134. Springer, Heidelberg (2010)
5. Bos, J.: Wide-Coverage Semantic Analysis with Boxer. In: Bos, J., Delmonte, R. (eds.) Semantics in Text Processing, pp. 277–286. College Publications (2008)
6. Chaudhri, V.K., John, B., Mishra, S., Pacheco, J., Porter, B., Spaulding, A.: Enabling Experts to Build Knowledge Bases from Science Textbooks. In: Proceedings of KCAP 2007 (2007)
7. Chen, D., Schneider, N., Das, D., Smith, N.A.: Probabilist frame-semantic parsing. In: Proceedings of NAACL-HLT (2010)
8. Cimiano, P.: Ontology learning and population from text: Algorithms, evaluation and applications. Springer (2006)
9. Cimiano, P., Völker, J.: Text2onto - a framework for ontology learning and data-driven change discovery (2005)
10. Coppola, B., Gangemi, A., Gliozzo, A., Picca, D., Presutti, V.: Frame Detection over the Semantic Web. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 126–142. Springer, Heidelberg (2009)
11. Das, D., Smith, N.A.: Semi-supervised frame-semantic parsing for unknown predicates. In: Lin, D., Matsumoto, Y., Mihalcea, R. (eds.) ACL, pp. 1435–1444. The Association for Computer Linguistics (2011)
12. Draicchio, F.: Frame-driven Extraction of Linked Data and Ontologies from Text. Master's Thesis, University of Bologna Electronic Press (February 2012), http://amslaurea.unibo.it/3165/

13. Petasis, G., Karkaletsis, V., Paliouras, G., Krithara, A., Zavitsanos, E.: Ontology Population and Enrichment: State of the Art. In: Paliouras, G., Spyropoulos, C.D., Tsatsaronis, G. (eds.) Multimedia Information Extraction. LNCS (LNAI), vol. 6050, pp. 134–166. Springer, Heidelberg (2011)
14. Fillmore, C.J.: Frame semantics, pp. 111–137. Hanshin Publishing Co., Seoul (1982)
15. Heath, T., Bizer, C.: Linked data: Evolving the web into a global data space, 1st edn. Synthesis Lectures on the Semantic Web: Theory and Technology 1:1. Morgan & Claypool (2011)
16. Kamp, H.: A theory of truth and semantic representation. In: Groenendijk, J.A.G., Janssen, T.M.V., Stokhof, M.B.J. (eds.) Formal Methods in the Study of Language, vol. 1, pp. 277–322. Mathematisch Centrum (1981)
17. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems 16, 72–79 (2001)
18. Magnini, B., Pianta, E., Popescu, O., Speranza, M.: Ontology Population from Textual Mentions: Task Definition and Benchmark. In: Proc. of the 2nd Workshop on Ontology Learning and Population. ACL (2006)
19. Nuzzolese, A.G., Gangemi, A., Presutti, V.: Gathering Lexical Linked Data and Knowledge Patterns from FrameNet. In: Proc. of the 6th International Conference on Knowledge Capture (K-CAP), Banff, Alberta, Canada (2011)
20. Ovchinnikova, E.: Integration of World Knowledge for Natural Language Understanding. Atlantis Press, Springer (2012)
21. Ruppenhofer, J., Ellsworth, M., Petruck, M.R.L., Johnson, C.R., Sceffczyk, J.: Framenet ii: Extended theory and practice (2010)
22. Tanev, H., Magnini, B.: Weakly supervised approaches for ontology population. In: Proceedings of the 2008 Conference on Ontology Learning and Population, pp. 129–143. IOS Press (2008)
23. Witte, R., Khamis, N., Rilling, J.: Flexible Ontology Population from Text: The OwlExporter. In: Calzolari, N., et al. (eds.) LREC, European Language Resources Association (2010)
24. Zhang, Z., Ciravegna, F.: Named Entity Recognition for Ontology Population using Background Knowledge from Wikipedia. In: Wong, W., Liu, W., Bennamoun, M. (eds.) Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances. IGI Global (2011)

Investigating the Semantics of Frame Elements

Sara Tonelli, Volha Bryl, Claudio Giuliano, and Luciano Serafini

Fondazione Bruno Kessler, Italy
{satonelli,bryl,giuliano,serafini}@fbk.eu

Abstract. Compared to other existing semantic role repositories, FrameNet is characterized by an extremely high number of roles or *Frame Elements* (FEs), which amount to 8,884 in the last resource release. This represents an interesting issue to investigate both from a theoretical and a practical point of view. In this paper, we analyze the semantics of frame elements by automatically assigning them a set of synsets characterizing the typical FE fillers. We show that the synset repository created for each FE can adequately generalize over the fillers, while providing more informative sense labels than just one generic semantic type. We also evaluate the impact of the enriched FE information on a semantic role labeling task, showing that it can improve classification precision, though at the cost of lower recall.

1 Introduction

FrameNet [29] is one of the most important semantic resources encoding information about situations, the *frames*, and participants, the semantic roles, also called *frame elements* (FEs). Although Filllmore's original formulation of frames included only six generic roles [15], FrameNet is currently characterized by an extremely high number of roles, which amount to 8,884 in the last resource release (version 1.5). This represents an interesting issue to investigate both from a theoretical and a practical point of view, also in the light of its ontological consistency [25].

FrameNet developers assume that events and situations in the world, corresponding to frames, should be described through highly specific roles rather than few, generic ones. Despite this, around 54% of the roles in FrameNet 1.5. have been assigned a semantic type label (40 labels in total), in a partial attempt to identify some common traits among different FEs and provide semantic constraints on FE fillers. For instance, the semantic type *Sentient* was assigned to the *Agent* FE. However, this information does not cover the whole FE set (only 54% of the FEs have a semantic type) and is often very high-level (consider e.g. *Physical_entity* type coupled with *Instrument* FE). Therefore, it could hardly be used for large-scale semantic analysis.

The number of frame elements makes the semantic role labeling (SRL) very challenging, since for many FEs only few training examples are available. In order to improve the performance of SRL systems, it would be important to generalize over the role fillers, while preserving specific FE information.

In this work, we present a generalization strategy over FE lexical fillers that relies on extracting information from a combination of existing Semantic Web resources, namely Wikipedia, WordNet [14] and YAGO [31]. Specifically, we devise an approach that

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 130–143, 2012.

takes advantage of the strengths of each of these resources: Wikipedia is currently the most extensive sense repository and shows an excellent coverage of named entities. Furthermore, robust tools for Wikipedia-based word sense disambiguation (WSD) allow for the disambiguation of the role fillers. WordNet, instead, has a better coverage of nominal entities and its hierarchical structure can be exploited to compute similarity measures between the fillers. Finally, the YAGO ontology combines knowledge about Wikipedia individuals and WordNet taxonomy structure, thus providing a structured description of Wikipedia concepts, which includes the characterization of a concept in terms of WordNet synsets. We show that these three resources are interoperable and can be successfully connected to enrich the frame element repository with additional semantic information. Furthermore, we present an experiment in which we integrate this knowledge into a postprocessor that checks the consistency of FE labels, as annotated by a SRL system, and discards annotations that are unlikely, given our enriched FE repository.

The paper is structured as follows: in Section 2 we present related work on FE analysis and selectional preferences. In Section 3 the workflow for the creation of the sense repository is detailed. In Section 4 we analyze a subpart of the repository, providing a qualitative evaluation of the resource. In Section 5 a task-based evaluation is presented, in which the sense repository is exploited to improve the performance of a SRL system. We draw some conclusions and detail future work in Section 6.

2 Related Work

FrameNet structure has undergone several revisions and consistency studies. However, they have been mainly focused on frames and on frame-to-frame relations [25], rather than on its large repository of frame elements. Past attempts to link FrameNet and WordNet concerned mainly finding for each frame a set of corresponding synsets, disregarding FE information [11,19,33]. An exception is the work by [2], in which the authors assign a synset to FE fillers based on latent semantic analysis in order to build lexical patterns to be included in a domain ontology. Also [9] present a methodology to enrich frames with argument restrictions provided by a super-sense tagger and domain specialization. The authors employ an off-the-shelf parser, whose tagset comprises 26 labels for nominal FE fillers. Our approach is different from the one presented in [9] in that we devise our own strategy for sense assignment and apply a different generalization method.

Our analysis of the semantics of frame elements leans on past research on *Selectional Preferences* (SPs), which describe typical fillers of a predicate's argument. However, while the original notion of selectional preferences referred to predicates [17,34], we adopt here a slightly different definition, by considering SPs *on a frame basis*, similar to [12]. In other words, we assume that SPs are shared among all predicates belonging to the same frame.

SPs have been used in different semantically-based NLP tasks such as word sense disambiguation [21], pseudo-disambiguation [13] and semantic role labelling [16,36]. Starting from [28], most approaches proposed for the acquisition of selectional preferences rely on a corpus-based methodology based on two steps: first, all argument heads

are *extracted* from a reference corpus, and then such heads are used to model some selectional preferences over the arguments by *generalizing* to other similar words.

In the generalization step, most relevant models of selectional preferences can be grouped into two classes: *i)* distributional models [27,12,4], which rely on word co-occurrence, and *ii)* semantic hierarchy-based models [28,1,8], which generalize over seen headwords using an ontology or a semantic hierarchy, usually WordNet. While the former are particularly suited to domain-specific applications, because they can be easily adapted using domain-specific generalization corpora, semantic hierarchy-based models can make predictions also for infrequent words with good accuracy, given that they are included in the hierarchy. In this paper, we focus on the second model for acquiring selectional preferences for FrameNet FEs and building a repository of preferences on a frame basis. We encode these preferences as a list of synsets assigned to each FE.

A first study aimed at including selectional preferences in frame argument classification was presented by [16]. The authors generalize over seen heads of nominal arguments in three ways: with automatic clustering, by bootstrapping training data from unlabeled texts and by ascending the WordNet type hierarchy until reaching a synset for which training data are available. The three approaches show similar precision but a lower coverage for the WordNet-based model. [12] proposes a methodology to compute selectional preferences for semantic roles by using the BNC as a generalization corpus to compute corpus-based similarity metrics between a candidate role headword and the seen headwords. This distributional model achieves smaller error rates than Resnik's WordNet-based model and EM-based clustering models, although it shows lower coverage than EM-based models. The approach is then extended in [13].

[36] show that the integration of SPs in a state-of-the-art SRL system yields statistically significant improvement in classification accuracy. However, our results are not directly comparable because we adopt FrameNet paradigm for semantic role labelling, while [36]'s work is based on PropBank-style roles. Besides, they integrate SPs in the system in the form of features and individual class predictors, which we see as part of our future work.

Most works on selectional preferences that implement WordNet-based models currently follow Resnik's formulation [28], which estimates the argument preference strength without performing word sense disambiguation. Attempts have been made to tackle ambiguity in the acquisition of selectional preferences, for example using Bayesian networks [7], but it still remains an open problem, since existing WordNet-based WSD systems cannot be easily applied to new domains and do not handle named entities. We claim that disambiguation is a necessary step in order to perform an accurate generalization over FEs fillers, but we introduce a new methodology that first disambiguates these fillers by assigning a Wikipedia page and then relies on WordNet and YAGO for generalization.

3 Creation of a FE Repository with Selectional Preferences

The aim of this work is to create a repository in which, for each tuple $(frame, FE_f)$, where FE_f is a frame element of $frame$, a set of senses with a relevance score is listed.

We extract these senses from WordNet because it provides high-quality additional information (e.g. relations, synonyms, etc.) and its hierarchy can be further exploited for computing similarity between FE fillers (see Section 5).

The repository is built based on the following steps:

- Disambiguation of FE fillers from the FrameNet corpus using a Wikipedia-based WSD system
- Linking each disambiguated FE filler to a WordNet synset
- If a synset was not assigned to a FE filler, mapping it to YAGO
- Creation of a synset repository for each $(frame, FE_f)$ by assigning a relevance score to each synset

The four steps are detailed in the following subsections.

3.1 Step 1: Disambiguation of FE Fillers

In order to collect a sense repository for each FE filler, we first disambiguate each filler by assigning a Wikipedia page W. We employ *The Wiki Machine*, a kernel-based WSD system (details on the implementation are reported in [32]), which has been trained on the Wikipedia dump of March 2010[1]. Since FE fillers can be both common nouns and named entities, we needed a WSD system that performs satisfactorily on both nominal types. A comparison with state-of-the-art system *Wikipedia Miner* [23] on the ACE05-WIKI dataset [3] showed that *The Wiki Machine* achieves a good performance on both types (0.76 F1 on named entities and 0.63 on common nouns), while *Wikipedia Miner* has a poorer performance on the second noun type (0.76 and 0.40 F1, respectively). These results were confirmed also in a more recent evaluation [22], in which *The Wiki Machine* achieved the highest F1 compared to an ensemble of academic and commercial systems such as DBpedia Spotlight, Zemanta, Open Calais, Alchemy API, and Ontos.

Disambiguation is performed on each annotated sentence from the FrameNet database 1.5 [29], including also the documents provided with continuous annotation. The system applies an 'all word' disambiguation strategy, in that it tries to disambiguate each word (or multiword) in a given sentence. Then, we match each disambiguated term with the original frame annotation $(frame, FE_f)$ and, in case the term (partially) matches a string corresponding to a FE, we assume that one possible sense of $(frame, FE_f)$ is represented in Wikipedia through W. The WSD system also assigns a confidence score to each disambiguated term. This confidence is higher in case the words occurring in the same context of the disambiguated term show high similarity, because the system assumes that disambiguation is likely to be more accurate.

We show in Fig. 1 the Wikipedia pages (and confidence score) that the WSD system associates with the sentence 'Sardar Patel was assisting Gandhiji in the Salt Satyagraha with great wisdom', an example sentence for the ASSISTANCE frame originally annotated with four FEs, namely *Helper*, *Benefited_party*, *Goal* and *Manner*.

Since Wikipedia is a repository of concepts, which are usually expressed by nouns, we are able to disambiguate only nominal fillers. This is in line with past research on selectional preferences, that are usually limited to nominal arguments [12,16].

[1] http://download.wikimedia.org/enwiki/20100312

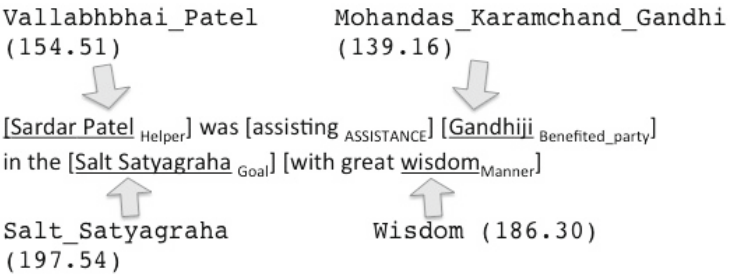


Fig. 1. Disambiguation example with confidence score

3.2 Step 2: Linking with WordNet

Although Wikipedia has an extensive coverage of senses, it is a semi-structured resource, whose concept organization is not rigorous because it is continuously growing and lacks a design a priori. Therefore, after the disambiguation step, we map each Wikipedia page W to a WordNet synset, so that we can better generalize over the lexical fillers using WordNet hierarchy.

For this purpose, we use BABELNET [24], a large semantic network where lexicographic and encyclopedic knowledge from WordNet and Wikipedia have been automatically integrated. The authors report a mapping between 105,797 synsets and 199,735 Wikipedia pages with a precision of 81.9%.

In our experiment, we apply such mapping in order to replace each Wikipedia page W assigned to $(frame, FE_f)$ with a synset. In this way, we devise a sort of Wikipedia-detour to WordNet-based WSD, because we obtain for each $(frame, FE_f)$ a list of related synsets (henceforth $Syns(FE_f)$). This approach has several advantages: since our WSD system is trained on the whole Wikipedia, it can potentially disambiguate (i.e. assign a Wikipage) to every term or multiword. This includes also many named entities that are not present in WordNet. For example, the expression *American_businessman* is not in WordNet but it was linked to the `Businessperson` page in Wikipedia and then to the synset {*businessperson#n#1, bourgeois#n#1*} through BabelNet. Also *mrs_thatcher* and *mr_gorbachev* cannot be matched with any lemma in WordNet, but they were respectively connected to the `Margaret_Thatcher` and the `Mikhail_Gorbachev` page, and then to *Thatcher#n#1* and *Gorbachev#n#1*.

3.3 Step 3: Enrichment through YAGO

Since Wikipedia contains much more concepts (i.e. senses) than WordNet, not all pages W associated with the FE fillers can be mapped to a WordNet synset. Furthermore, BABELNET covers only a subset of all existing Wikipedia pages. As reported in Table 1, 226,520 FE fillers were disambiguated by assigning a Wikipedia page with non-zero confidence score, but 29,924 of them were not linked to a synset because the pages were not found in BABELNET.

In order to cope with this problem, we exploited YAGO [31], an automatically created large-scale ontology, which combines WordNet taxonomy with knowledge about individuals extracted from Wikipedia. Specifically, most of WordNet synsets become

classes in YAGO and *SubClassOf* relation between classes is derived from hyponymy relation in WordNet. In turn, Wikipedia categories are mapped to WordNet synsets, that is, they are added to YAGO as subclasses of aforementioned WordNet-derived classes. Thus, given a Wikipedia page URL, one can query the YAGO ontology to obtain a list of WordNet synsets corresponding to its Wikipedia categories. Note that, differently from Step 2 where BABELNET provided us with at most one WordNet synset per Wikipedia page, multiple synsets corresponding to multiple page categories are obtained.

At the end of this step we were able to acquire WordNet synsets for 21,505 FE fillers, thus increasing the linking by 9%. To access the ontology, we used Java API for quering YAGO in SPARQL, which is provided by the YAGO team and works with YAGO dumps in TDB format[2].

3.4 Step 4: Creation of the Repository

In the final FE repository, we do not want to simply list all synsets acquired for each $(frame, FE_f)$. The key idea is rather to exploit WordNet taxonomy to generalize over the extracted synsets. In particular, for each $(frame, FE_f)$ pair a tree of senses was created based on WordNet hyponymy relation, where a node corresponds to a synset s and stores the number of examples linked to s or its hyponym. Then, the distribution of senses for $(frame, FE_f)$ is the lowest (starting from the most specific synsets) tree level, in which the nodes containing at least 10% of examples cover in total at least 60% of examples. In this way, the most representative among the most specific synsets are selected.

For each synset which is finally selected and included in the repository, we also compute the conditional probability $P(s|FE_f)$ as:

$$P(s|FE_f) = \frac{Count(s, FE_f)}{Count(FE_f)} \tag{1}$$

The selected synsets often provide a better characterization of a FE than its original description in FrameNet. For example, the *Undergoer* FE in the BEING_ROTTED frame was described as 'the organic matter that has decayed' by FrameNet lexicographers. However, the *Undergoer* instances in FrameNet annotated sentences include a variety of fillers such as 'house', 'mural', 'nest', 'lung', 'butter', 'reptile', etc. These are all well represented in the synset repository acquired for *Undergoer*:[3]

substance#n#1, 6, 0.27
artifact#n#1, 4, 0.18
body_part#n#1, 3, 0.14
natural_object#n#1, 3, 0.14
food#n#1, 2, 0.09
living_thing#n#1, 2, 0.09
substance#n#8, 1, 0.04
matter#n#3, 1, 0.04

[2] http://www.mpi-inf.mpg.de/yago-naga/
[3] Note that the synset entries are in the form *synset, occurrences,* $P(s|FE_f)$.

For more examples the reader is referred to Section 4, to [5] and to the online version of the repository: its text version is available online[4], while the creation of the RDF/OWL version is the work in progress.

We report in Table 1 some statistics about the data processed in the four steps described above. Note that we consider only the fillers for which the WSD system provides a non-zero confidence value. Taking zero confidence links into account would increase the coverage by 12.5%, but it would also increase the amount of noise and, consequently, decrease the quality of the acquired senses.

Table 1. Statistics about the extracted data

FE fillers linked to a Wikip. (Step1)	226,520
FE fillers linked to a synset (Step2)	196,596 (87%)
FE fillers linked to a synset through YAGO (Step3)	21,505 (9%)
$(frame, FE_f)$ pairs in the final repository (Step4)	3,847
Avg. synsets for $(frame, FE_f)$	6

4 Qualitative Evaluation

The acquired repository of senses can be divided into three parts. To a small portion of frame elements a semantic type explicitly mapped to a WordNet synset was assigned by FrameNet lexicographers. There are 242 such $(frame, FE_f)$ pairs, approximately 6% of the repository. For 1573 pairs (41%) a FrameNet semantic type was assigned but no mapping to WordNet was provided. For the remaining 53% of the pairs, no semantic description of a role filler is available in FrameNet. In the following, we present a qualitative evaluation of three repository parts, done manually for those pairs that most frequently appear in the FrameNet corpus. Some more details on the qualitative evaluation of the first version of the sense repository can be found in [5].

FrameNet semantic type is assigned and linked to WordNet: We have considered 88 $(frame, FE_f)$ pairs, each having greater or equal than 50 examples in the FrameNet corpus. For this small portion of the repository, a straightforward evaluation is possible, comparing the acquired synsets with those assigned to the FE semantic types in FrameNet. Specifically, we count as correct the synsets assigned to $(frame, FE_f)$ in the repository if they match or are hyponyms of the synset associated with the semantic type of $(frame, FE_f)$ in FrameNet.

18 out of the 88 pairs considered were originally assigned the *Human* semantic type (and *person#n#1* synset) in FrameNet. For 16 of them, the average matching score is around 96%. The other 2 examples are (COLLABORATION, *Partners*) and (COLLABORATION, *Partner_2*) (100 examples per pair). Let us consider the latter case, *Partner_2*:

[4] https://dkm.fbk.eu/index.php/FrameNet_extension_repository_of_senses

for this frame element, the acquired distribution of senses includes not only *person#n#1* synset, but also *social_group#n#1, organization#n#1, company#n#1* and some others, which not only justifies the low matching score but allows for a more complete description of possible FE fillers. In some other cases, one sense has been acquired for a FE, being more specific than the corresponding FrameNet semantic type. This would make a replacement possible between the original FE semantic type and the newly acquired synset. For instance, the pair (WEAPON, *Weapon*) (165 examples) was assigned the semantic type *artifact#n#1* in FrameNet, while we have acquired the *weapon#n#1* synset in 98% of the examples. High matching scores are obtained for the *artifact#n#1* and the *body_part#n#1* semantic types. For a number of types (e.g. *location#n#1*) the matching score is low, while the suggested distribution includes, in addition to *location#n#1*, the *structure#n#1* (hyponym of *room#n#1* and *house#n#1*), *organization#n#1* and *event#n#1* synsets.

FrameNet semantic type is assigned but not linked to WordNet: For this part of the repository, evaluation is more complex because we do not rely on gold synsets. Therefore, we perform a post-hoc analysis and count as correct the synsets assigned to $(frame, FE_f)$ in the repository if they comply with the semantic type associated with $(frame, FE_f)$ in FrameNet, according to human judgment. We have considered 69 $(frame, FE_f)$ pairs of this group, each having more than 400 examples in the FrameNet corpus. 64 of these pairs were assigned the *Sentient* semantic type in FrameNet, and for them the average matching score is 93%: more specifically, for 63 pairs the score is greater than 80%, and for 30 pairs greater or equal than 95%. One exception with 76% match is the (KILLING, *Victim*) pair (520 examples), where another 12% of the occurrences are learnt to be *animal#n#1* and *group#n#1* (e.g. ethnical groups). Other semantic types for these 69 pairs are *Goal* (3 pairs), *Path* and *State_of_affairs* (1 pair each). To give an example, for (SELF_MOTION, *Goal*) (FrameNet semantic type is *Goal*) the distribution of senses includes *structure#n#1* (hyponym of *room#n#1, area#n#1, building#n#1*), *location#n#1* (hyponym of e.g. *area#n#4*), *vehicle#n#1, event#n#1, social_group#n#1, instrumentality#n#1* (hyponym of e.g. *furniture#n#1*).

No semantic type associated with FE in FrameNet: This is the largest part of the repository and the most problematic from the evaluation point of view. Also in this case, we perform a post-hoc analysis, and accept as correct the synsets assigned to $(frame, FE_f)$ if they comply with the FE_f definition in FrameNet. We have considered 31 $(frame, FE_f)$ pairs, each having more than 400 examples in the FrameNet corpus. For 16 pairs, the *person#n#1* synset was suggested as the most frequent role filler type, with 13 pairs having the average matching score of 91%. For other pairs, like, for instance, (MAKE_NOISE, *Sound_source*), the distribution includes, in addition to *person#n#1*, *animal#n#1, instrument#n#1, atmospheric_phenomenon#n#1*, etc. Another example of a semantic role description is that acquired for *(*LEADERSHIP, *Governed*), whose distribution of senses includes *person#n#1, location#n#1, socail_group#n#1* and *structure#n#1* synsets.

Although the presented analysis is limited to a small set of $(frame, FE_f)$ pairs, we still can conclude that in many cases the acquired senses are not only correct, but also provide a much more accurate semantic description of possible FEs fillers than FrameNet semantic types.

5 Task-Based Evaluation

An issue we want to address with the present study is the possibility to exploit the information on FE fillers collected so far in a real NLP task. Specifically, we want to assess if, despite the possible mistakes introduced in our pipeline through the linking steps (from Wikipedia to WordNet through BABELNET and YAGO), the sense repository can be used as it is to improve the performance of existing NLP systems.

We test this hypothesis in a semantic role labelling (SRL) task. For each $(frame, FE_f)$ pair, we employ the assigned synset list as selectional preferences and apply them to the output of a SRL system, in order to accept or discard the FE labels based on such preferences.

The evaluation process comprises the following steps:

1. Annotate unseen text with a frame-based SRL system
2. For each $(frame, FE_f)$ assigned by the system, disambiguate the lexical filler by assigning a Wikipedia page
3. Map the Wikipedia page to a WordNet synset s_0 using BABELNET and YAGO (see Sections 3.2 and 3.3.)
4. Compute a similarity score between s_0 and the synsets $Syns(FE_f)$ previously associated with $(frame, FE_f)$
5. If the similarity score is above a certain threshold, the FE_f label assigned by the SRL system is accepted, otherwise it is discarded.

While the widely used WordNet-based model proposed by [28] estimates selectional preferences for an argument filler by splitting the filler frequency equally among all synsets containing it, we can now take advantage of the outcome of the disambiguation step, and apply a model for selectional preferences directly to synsets. Another advantage is that we generalize over fillers based on the synsets, therefore we can admit also unseen lexical fillers (i.e. terms that were not included in the FrameNet data used to create the synset repository).

5.1 Similarity Function

In order to compute the similarity score mentioned in Section 5 at (4), we implement a function that computes the similarity measure between s_0 and each $s \in Syns(FE_f)$ and multiplies it by the probability of s $P(s|FE_f)$ as described in Eq. 1. The highest product obtained during this comparison corresponds to the similarity score $S_{FE_f}(s_0)$ between s_0 and the most similar s in the sense repository. The probability value should boost the similarity between s_0 and s, if s was frequently observed for FE_f in the FrameNet corpus:

$$S_{FE_f}(s_0) = arg \max_{s \in Syns(FE_f)} sim(s_0, s) \times P(s|FE_f) \quad (2)$$

5.2 Similarity Measures

The similarity metrics instantiating the sim function of Equation 2 are based on existing implementations of WordNet-based metrics of semantic *similarity* implemented in the *WordNet::Similarity* library [26][5]. In particular, we test 4 different measures that are either based on information content or on path length.

The information-based measures of *similarity* rely on the *information content* of the least common subsumer (LCS) of two synsets, given the sense-tagged corpus SemCor[6] as reference source for information. The measures we use are *jcn* [18] and *lin* [20]. *jcn* and *lin* are based on the sum of the information content of the two synsets being considered. The *jcn* measure corresponds to the difference between the sum and the information content of LCS, while the *lin* measure is equal to twice the information content of LCS divided by the sum of the information content of each input synset.

As for path-length based measures of semantic similarity, they rely on the assumption that the length of the path between a pair of synsets in the WordNet taxonomy can be used to compute their similarity. In particular, *path* corresponds to the inverse of the shortest path between two synsets and *wup* [35] divides the depth of the synsets' LCS by the sum of the depth of the single synsets.

5.3 Evaluation Results

Our task-based evaluation relies on a postprocessor that, given the output of a FrameNet-based SRL system, assesses for each annotated FE if it is correct or not. Since our post-processing algorithm requires a boolean true/false judgment, whereas each of the similarity measures we applied returns a numerical value of similarity, we try different *acceptance thresholds* for each measure in order to estimate the best cutoff value for accuracy optimization.

We use the test set released for SemEval 2010, Task 10 [30], which we first annotate using the SEMAFOR frame semantic parser [10] in the standard setting. Then, we disambiguate each nominal filler of an annotated FE by assigning a Wikipedia link and then a WordNet synset. Finally, we compute a similarity score between the disambiguated filler and the sense repository previously associated with the given FE by applying the sim function. We test the similarity function described in Section 5.1 combined with the four WordNet metrics. Further, we experiment with different acceptance thresholds.

We report in Table 2 the results of this evaluation. SEMAFOR performance[7] is compared with the *Exact match* score, which is obtained by retaining only the FE labels assigned by SEMAFOR whose associated synset appears in the sense repository for the given FE. For instance, if the token *house* was annotated by SEMAFOR as belonging to the LOCATING frame and having the *Location* FE, we first disambiguate it by assigning

[5] For an overview see [6]

[6] http://www.cse.unt.edu/~rada/downloads.html

[7] SEMAFOR performance is lower than the one reported in the SemEval task [30], because the system release used for this evaluation was not trained on SemEval training data. Also our sense repository does not include these training data.

the *house#n#1* synset and consider it correct only if we find *house#n#1* in the sense repository for *Location*, LOCATING. The *Exact match* implements a basic version of our strategy for FE selection, in that no similarity function is applied. In Table 2 we report also the performance achieved by applying to SEMAFOR output our strategy for FE validation based on selectional preferences. The acceptance thresholds of the four WordNet similarity metrics are set in order to maximize precision.

Table 2. Comparative evaluation of SEMAFOR performance and FE selection strategy

	Precision	**Recall**	**F_1**
SEMAFOR	0.446	0.492	0.468
Exact match	0.467	0.371	0.414
jcn	0.469	0.412	**0.439**
lin	0.474	0.375	0.419
wup	0.476	0.399	0.434
path	**0.478**	0.399	0.435

5.4 Results

Table 2 shows that the information collected on the semantics of FEs can be used to improve SRL precision. The best strategy for FE selection is based on path distance between the current synset and previously observed synsets. In general, however, our approach does not seem to be effective in improving SRL performance, because none of the proposed strategies yields an improvement over SEMAFOR F_1. The information collected on the semantics of FEs should probably be integrated into the system in the form of features in order to be fully exploited and interact with the existing syntactic and lexical features. Although the system is available as an open source project,[8] however, it is extremely difficult to manipulate its complex architecture and extend it in order to include new features. This extension is left to future work.

6 Conclusions

In this paper, we have investigated how to cast the semantics of frame elements without relying on generic semantic type labels. Specifically, we have created a repository in which each $(frame, FE_f)$ pair has been associated with a list of synsets that are representative of the fillers' senses. The repository has been obtained by mapping the FE fillers to WordNet synsets through Wikipedia, BABELNET and YAGO.

The repository has been manually inspected, showing that, in some cases, the acquired synsets provide a more accurate and multifaceted semantic description of possible FEs fillers than FrameNet semantic types. When using the sense repository to improve the performance of a SRL system, however, recall is a main issue, while precision achieves some improvement.

[8] http://code.google.com/p/semafor-semantic-parser/

In the future, we plan to further improve the consistency of the sense repository by performing a semi-automatic check of the acquired synsets. In addition, we will investigate the possibility to merge the synsets of different $(frame, FE_f)$ pairs, if the FEs share the same label, such as (ABUSING, *Victim*), (ATTACK, *Victim*), (OFFENSES, *Victim*).

As for SRL, we will model the acquired knowledge on typical FE fillers as features, so that it can be integrated directly into a supervised SRL system. This will allow for a more thorough task-based evaluation.

References

1. Abe, N., Li, H.: Learning word association norms using tree cut pair models. In: Proceedings of the 10th International Conference on Machine Learning, Bari, Italy, pp. 3–11 (1996)
2. Basili, R., Croce, D., De Cao, D., Giannone, C.: Learning Semantic Roles for Ontology Patterns. In: Proceedings of the Workshop on Natural Language Processing and Ontology Engineering (NLPOE 2009), Held in Conjunction with the Conference on Web Intelligence (WI/IAT 2009), Milan, Italy (2009)
3. Bentivogli, L., Forner, P., Giuliano, C., Marchetti, A., Pianta, E., Tymoshenko, K.: Extending English ACE 2005 Corpus Annotation with Ground-truth Links to Wikipedia. In: 23rd International Conference on Computational Linguistics, pp. 19–26 (2010)
4. Bergsma, S., Lin, D., Goebel, R.: Discriminative learning of selectional preferences from unlabeled text. In: Proceedings of the 13th Conference on Empirical Methods in Natural Language Processing, Honolulu, HI, US, pp. 59–68 (2008)
5. Bryl, V., Tonelli, S., Giuliano, C., Serafini, L.: A Novel FrameNet-based Resource for the Semantic Web. In: Proceedings of ACM Symposium on Appliced Computing, Technical Track on The Semantic Web and Applications (2012)
6. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Computational Linguistics 31(1), 13–48 (2006)
7. Ciaramita, M., Johnson, M.: Explaining away ambiguity: Learning verb selectional preference with Bayesian networks. In: Proceedings of the 18th International Conference on Computational Linguistics (COLING), Saarbrücken, Germany, pp. 187–193 (2000)
8. Clark, S., Weir, D.: Class-based probability estimation using a semantic hierarchy. In: Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA, US, pp. 95–102 (2001)
9. Coppola, B., Gangemi, A., Gliozzo, A., Picca, D., Presutti, V.: Frame Detection over the Semantic Web. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 126–142. Springer, Heidelberg (2009)
10. Das, D., Schneider, N., Chen, D., Smith, N.A.: Probabilistic frame-semantic parsing. In: Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies Conference, Los Angeles, CA, USA (2010)
11. De Cao, D., Croce, D., Pennacchiotti, M., Basili, R.: Combining Word Sense and Usage for modeling Frame Semantics. In: Proceedings of STEP 2008, Venice, Italy (2008)
12. Erk, K.: A simple, similarity-based model for selectional preferences. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007), Prague, Czech Republic, pp. 216–223 (2007)
13. Erk, K., Padó, S., Padó, U.: A flexible, corpus-driven model of regular and inverse selectional preferences. Computational Linguistics 36(4), 723–763 (2010)

14. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press (1998)
15. Fillmore, C.J.: The Case for Case. In: Bach, E., Harms, R.T. (eds.) Universals in Linguistic Theory, pp. 1–88. Holt, Reinhart and Winston, New York (1968)
16. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. Computational Linguistics 28(3), 245–288 (2002)
17. Hirst, G.: Semantic Interpretation and the Resolution of Ambiguity. Cambridge University Press (1987)
18. Jiang, J.J., Conrath, D.W.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X), Taiwan, pp. 19–33 (1997)
19. Laparra, E., Rigau, G.: Integrating wordnet and framenet using a knowledge-based word sense disambiguation algorithm. In: Proceedings of RANLP 2009, pp. 208–213 (2009), http://www.aclweb.org/anthology/R09-1039
20. Lin, D.: Automatic retrieval and clustering of similar words. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics, Montreal, Quebec, Canada, pp. 768–774 (1998)
21. McCarthy, D., Carroll, J.: Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. Computational Linguistics 29(4), 639–654 (2003)
22. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In: Proceedings of I-SEMANTICS, the 7th Conference on Semantic Systems (2011)
23. Milne, D., Witten, I.H.: Learning to link with Wikipedia. In: CIKM 2008: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 509–518. ACM, NY (2008)
24. Navigli, R., Ponzetto, S.P.: Babelnet: Building a very large multilingual semantic network. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, pp. 216–225 (2010)
25. Ovchinnikova, E., Vieu, L., Oltramari, A., Borgo, S., Alexandrov, T.: Data-Driven and Ontological Analysis of FrameNet for Natural Language Reasoning. In: Proceedings of LREC 2010, Valletta, Malta (2010)
26. Pedersen, T., Patwardhan, S., Michelizzi, J.: WordNet:Similarity - Measuring the Relatedness of Concepts. In: Proceedings of the 5th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2004), Boston, MA, USA, pp. 38–41 (2004)
27. Pereira, F., Tishby, N., Lee, L.: Distributional clustering of English words. In: Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, Columbus, OH, US, pp. 183–190 (1993)
28. Resnik, P.: Semantic classes and syntactic ambiguity. In: Proceedings of the Workshop on Human Language Technology, Morristown, NJ, USA, pp. 278–283 (1996)
29. Ruppenhofer, J., Ellsworth, M., Petruck, M.R., Johnson, C.R., Scheffczyk, J.: FrameNet II: Extended Theory and Practice (2010), http://framenet.icsi.berkeley.edu/book/book.html
30. Ruppenhofer, J., Sporleder, C., Morante, R., Baker, C.F., Palmer, M.: SemEval-2010 Task 10: Linking Events and Their Participants in Discourse. In: Proceedings of SemEval 2010: 5th International Workshop on Semantic Evaluations (2010)
31. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706. ACM Press (2007)
32. Tonelli, S., Giuliano, C., Tymoshenko, K.: Wikipedia-based WSD for multilingual frame annotation. Artificial Intelligence (2012)

33. Tonelli, S., Pighin, D.: New Features for FrameNet - WordNet Mapping. In: Proceedings of CoNLL 2009, Boulder, Colorado, pp. 219–227 (2009), http://www.aclweb.org/anthology/W09-1127
34. Wilks, Y.: An intelligent analyzer and understander of English. Communications of the ACM 18(5), 264–274 (1975)
35. Wu, Z., Palmer, M.: Verb Semantics and Lexical Selection. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, New Mexico (1994)
36. Zapirain, B., Agirre, E., Màrquez, L., Surdeanu, M.: Improving semantic role classification with selectional preferences. In: Proceedings of the 2010 Annual Conference of the North American Chapter of the ACL, Los Angeles, CA, USA, pp. 373–376 (2010)

Keys and Pseudo-Keys Detection for Web Datasets Cleansing and Interlinking

Manuel Atencia[1,2], Jérôme David[1,3], and François Scharffe[4]

[1] INRIA & LIG, France
{manuel.atencia,jerome.david}@inria.fr
[2] Université de Grenoble 1, France
[3] Université de Grenoble 2, France
[4] Université de Montpellier 2 & LIRMM, France
francois.scharffe@lirmm.fr

Abstract. This paper introduces a method for analyzing web datasets based on key dependencies. The classical notion of a key in relational databases is adapted to RDF datasets. In order to better deal with web data of variable quality, the definition of a pseudo-key is presented. An RDF vocabulary for representing keys is also provided. An algorithm to discover keys and pseudo-keys is described. Experimental results show that even for a big dataset such as DBpedia, the runtime of the algorithm is still reasonable. Two applications are further discussed: (i) detection of errors in RDF datasets, and (ii) datasets interlinking.

1 Introduction

The notion of a key is essential for relational databases. Keys allow to uniquely identify each tuple in a relation. Further, the unicity property of keys is exploited in order to optimize data access through the construction of indexes. Keys are usually identified and chosen by the relational schema engineer, as part of the schema normalization process. However, there also exist algorithms that detect keys and functional dependencies inside a given database [1,2].

In the context of the Semantic Web, it is only since the release of OWL2 that modelling keys is possible. A key in OWL2 for a given class consists of a set of properties allowing to uniquely identify an instance of the class. According to the semantics of OWL2, two instances having the same values for the properties of a key are considered identical. Using keys thus requires to know in advance the data that will be represented according to the ontology. This is not compatible with the decentralized publication of datasets in the Web. However, the discovery of keys in RDF datasets allows to perform integrity checking such as duplicate detection. More interestingly, keys can be used to select sets of properties with which to compare data issued from different datasets.

In this paper we propose to discover potential keys in RDF datasets. Given the variable quality of web data, our approach allows to tolerate a few instances to have the same values for the properties of a key. In that case, we will use the

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 144–153, 2012.

term pseudo-key. We also put forward to associate discovered keys to the dataset as metadata by extending the VoID vocabulary [3].

The remainder of the paper is organized as follows. Section 2 includes formal definitions of key and pseudo-key dependencies in RDF datasets, and presents a brief description of an algorithm to discover keys and pseudo-keys. Section 3 shows experimental results. An RDF vocabulary for representing keys is given in Section 4. Two distinct applications are explained in Section 5. Related work is summarized in Section 6 and Section 7 concludes the paper.

2 Key and Pseudo-Key Dependencies in RDF Datasets

In this section we introduce the definitions of a key and a pseudo-key in an RDF dataset (Section 2.1). Then we briefly describe an algorithm for detecting keys and pseudo-keys in RDF datasets (Section 2.2). This algorithm is based on TANE [2], an algorithm for discovering functional approximate dependencies in relational databases.

2.1 Definitions

Data representation on the Semantic Web is realized using the RDF language. In this paper, we denote the sets of all URIs, blank nodes and literals by $\mathbf{U}$, $\mathbf{B}$ and $\mathbf{L}$, respectively. An RDF *triple* is a tuple $t = \langle s, p, o\rangle$ where $s \in \mathbf{U} \cup \mathbf{B}$ is the *subject* or instance of t, $p \in \mathbf{U}$ is the *predicate* or property, and $o \in \mathbf{U} \cup \mathbf{B} \cup \mathbf{L}$ is the *object* of t. An RDF *graph* is a set of RDF triples. Given an RDF graph G, the sets of subjects, predicates and objects appearing in G are denoted by $sub(G)$, $pred(G)$ and $obj(G)$, respectively.

Let G be an RDF graph. A predicate $p \in pred(G)$ can be seen as a relation between the subject and object sets of G, i.e., $p \subseteq sub(G) \times obj(G)$. It can also be seen as a partial function between the subject set and the powerset of the object set, i.e., $p : sub(G) \to 2^{obj(G)}$. This is the formalization that we will follow in this paper. To be more precise,

$$p(s) = \{o \in obj(G) : \langle s, p, o\rangle \in G\}$$

Then, the domain of the predicate p is the set

$$dom(p) = \{s \in sub(G) : \text{there exists } o \in obj(G) \text{ with } \langle s, p, o\rangle \in G\}$$

In the following definition we introduce our notions of key and minimal key in an RDF graph.

Definition 1. *Let G be an* RDF *graph and $P \subseteq pred(G)$. The set of predicates P is a* key *in G if for all $s_1, s_2 \in sub(G)$ we have that, if $p(s_1) = p(s_2)$ for all $p \in P$ then $s_1 = s_2$. The set P is a* minimal key *if it is a key and there exists no set $P' \subseteq pred(G)$ such that P' is a key and $P' \subset P$.*

The above definition is analogous to that one of the relational model in databases. The first main difference is that, unlike attributes, predicates can take multiple values: if p is a predicate and $s \in dom(p)$, then $p(s)$ is, in general, a non-singleton value set. The second one is that properties are not necessary defined on the whole set of individuals.

In line with TANE algorithm, given an RDF graph G, our approach lies in building the partition of $sub(G)$ induced by a set of predicates P. If this partition is made up of singletons, then P is a key.

Definition 2. *Let G be an* RDF *graph and $p \in pred(G)$. The* partition induced by the predicate p *is defined by*

$$\pi_p = \{p^{-1}(p(s))\}_{s \in dom(p)}$$

Let $P = \{p_1, \ldots, p_n\} \subseteq pred(G)$. The partition induced by the predicate set P *is defined by*

$$\pi_P = \{S_1 \cap \ldots \cap S_n\}_{(S_1,\ldots,S_n) \in \pi_{p_1} \times \ldots \times \pi_{p_n}}$$

Lemma 1. *Let G be an* RDF *graph and $P \subseteq pred(G)$. The predicate set P is a key in G if and only if π_P is made up of singletons, i.e., $|S| = 1$ for all $S \in \pi_P$.*

The complexity of finding keys in an RDF graph is polynomial in the number of subjects, but exponential in the number of predicates. For this, we introduce two criteria to reduce the search space. First, we discard sets of predicates which share few subjects compared to the total number of subjects in the graph, as they are not interesting for the applications we have in mind (Section 5). Second, we restrict ourselves to compute "approximate" keys, what we call pseudo-keys.

Definition 3. *Let G be an* RDF *graph and $P \subseteq pred(G)$. The* support *of P in G is defined by*

$$support_G(P) = \frac{1}{|sub(G)|} \left| \bigcap_{p \in P} dom(p) \right|$$

The predicate set P fulfills the minimum support criterion *if $support(P) \geq \lambda_s$ where $\lambda_s \in [0, 1]$ is a given* support threshold.

Definition 4. *Let G be an* RDF *graph and $P \subseteq pred(G)$. The predicate set P is a* pseudo-key *in the graph G with* discriminability threshold λ_d *if*

$$\frac{|\{S \subseteq sub(G) | S \in \pi_P \text{ and } |S| = 1\}|}{|\pi_P|} \geq \lambda_d$$

2.2 Algorithm

The algorithm to compute keys and pseudo-keys in RDF datasets uses the same partition representation and breadth-first search strategy as TANE [2]. In order to prune the search space we discard any set of predicates including

- a key or a pseudo-key (for a given discriminability threshold λ_d),
- a subset the support of which is lower than a given threshold λ_s,
- a subset in which a functional dependency holds.

Unlike TANE, we discard properties by looking at their support. Indeed, this is useful because in RDF datasets properties may not be instantiated for each individual. In contrast, we cannot apply the optimization used in TANE based on stripping partitions (discarding singleton sets). Finally, since our goal is to find keys only, there is no need to test exhaustively all the functional dependencies.

3 Experimental Results

The key and pseudo-key discovery algorithm is implemented in Java. In order to improve time performance, datasets are locally stored on disk and indexed using Jena TDB[1]. We ran the algorithm on a quad-core Intel(R) Xeon(R) E5430 @ 2.66GHz computer with 8GB of memory. Table 1 shows the amount of time needed (computation+disk access) for computing all the keys and pseudo-keys for every class in DBpedia, DrugBank, DailyMed and Sider. The algorithm was parametrized with support and discriminability thresholds $\lambda_s = 0.1$, $\lambda_d = 0.99$.

Table 1. Datasets size, number of keys and pseudo-keys found, and runtime

	#triples	#classes	#properties	#instances	#keys	#pseudo-keys	runtime
DBpedia	13,8 M	250	1,100	1,668,503	2,945	6,422	179'48"
DrugBank	0,77 M	8	119	19,693	285	2,755	6'58"
DailyMed	0,16M	6	28	10,015	3	1,168	1'46"
Sider	0,19M	4	11	2,674	11	3	5"

The runtime results of Table 1 agree with the fact that the number of minimal keys can be exponential in the number of properties. However, even for a dataset of the size of DBpedia with 13,8M of triples, the runtime is still reasonable.

4 An RDF Vocabulary for Representation Keys

Once computed, keys and pseudo-keys constitute a new body of knowledge that can be linked to the dataset as part of its metadata. We present in this section a small vocabulary that allows to represent keys and pseudo-keys in RDF. This vocabulary gives an alternative to the `owl:hasKey` property. As mentioned in Section 1, OWL2 keys for a class expressed in an ontology imposes that every dataset using the class must respect the key. When a key is not general enough to be applied to every dataset, it might be more convenient to attach it at the dataset level instead. Keys can be computed by analysing the dataset using an algorithm like the one introduced in this paper.

An adequate vocabulary to attach metadata to RDF datasets is the so-called Vocabulary of Interlinked Datasets (VoID) [3]. In particular, VoiD defines the class `void:Dataset` to represent datasets. Keys can thus be attached to datasets using this class as a hook. The "Key Vocabulary" contains 1 class (**Key**) and the following 8 properties (see also Figure 1):

[1] http://incubator.apache.org/jena/documentation/tdb/

isKeyFor link a key to a VoID dataset
hasKey indicates a key for a given class
property a property belonging to a key
nbProp the number of properties in a key
support support for a key as defined in Section 2.1
discriminability discriminability threshold for a key as defined in Section 2.1
instanceCount the number of instances for a class in the dataset
hasException subjects violating the key (in case of a pseudo-key)

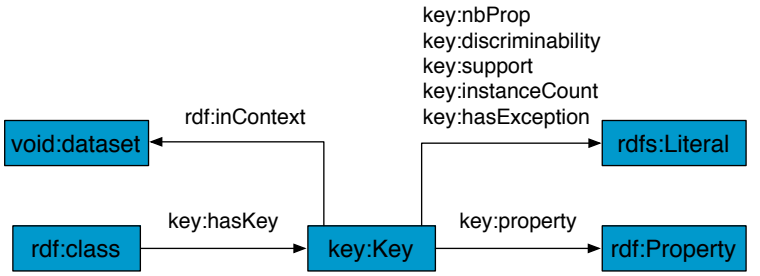


Fig. 1. Key vocabulary

Keys computed for diverse datasets, including DBPedia, are all published according to this vocabulary and available as linked-data on our server.[2].

5 Applications

Below we describe two applications: error detection (Section 5.1) and datasets interlinking (Section 5.2).

5.1 Error Detection

The discovery of pseudo-keys in a dataset may reveal the existence of duplicates or errors in the dataset. In order to find errors, each pseudo-key is transformed into a SPARQL query to retrieve the instances that have the same values for the properties of the pseudo-key.[3] The query results can be used as a basis for error correction. This workflow is illustrated in Figure 2.

We have applied this method over the 250 classes of DBPedia. Table 2 shows the pseudo-keys for the class `dbpedia:Person` computed with a minimal support $\lambda_s = 0.2$ and a discriminability threshold $\lambda_d = 0.999$.

The first row of Table 2 tells us that there exist persons who were born and dead in the same days, which is possible but unlikely to happen. The following query finds which resources of the class `DBPedia:Person` have the same values for `dbpedia:birthDate` and `dbpedia:deathDate`:

[2] `http://data.lirmm.fr/keys/`
[3] The transformation is performed by the program DuplicateFinder available at `https://gforge.inria.fr/projects/melinda/`

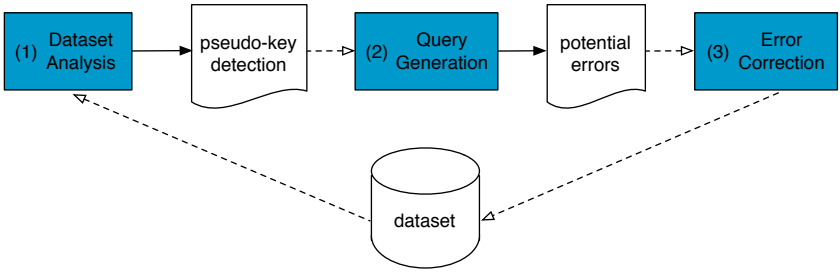


Fig. 2. Workflow for error detection using pseudo-keys

Table 2. Pseudo-key detection for the class `DBPedia:Person`

Pseudo-keys	Support
`http://dbpedia.org/ontology/deathDate` `http://dbpedia.org/ontology/birthDate`	0.203
`http://dbpedia.org/ontology/deathDate` `http://dbpedia.org/ontology/deathPlace`	0.216
`http://xmlns.com/foaf/0.1/name` `http://dbpedia.org/ontology/birthPlace`	0.442
`http://xmlns.com/foaf/0.1/surname` `http://purl.org/dc/elements/1.1/description`	0.459
`http://dbpedia.org/ontology/deathPlace` `http://dbpedia.org/ontology/birthDate`	0.480

```
SELECT DISTINCT ?x ?y
WHERE {
  ?x dbpedia-owl:deathDate ?dp1;
     dbpedia-owl:birthDate ?dp2;
     rdf:type dbpedia-owl:Person.
  ?y dbpedia-owl:deathDate ?dp1;
     dbpedia-owl:birthDate ?dp2;
     rdf:type dbpedia-owl:Person.
      MINUS {
         ?x dbpedia-owl:deathDate ?dpx1;
            dbpedia-owl:birthDate ?dpx2.
         ?y dbpedia-owl:deathDate ?dpy1;
            dbpedia-owl:birthDate ?dpy2.
         FILTER (?dpx1!=?dpy1)
         FILTER (?dpx2!=?dpy2)
      }
   FILTER (?x!=?y) }
```

In this particular example, the MINUS query pattern is not required because `dbpedia-owl:birthDate` and `dbpedia-owl:deathDate` are both single valued properties, but it would be necessary in the case of multivalued properties.

The above query returned 122 pairs of instances.[4] Manual analysis confirmed several kinds of errors. A first type of error is due to the existence of resources describing the same object. For example, `dbpedia:Louis_IX_of_France` and `dbpedia:Louis_IX_of_France__Saint_Louis___1`. Now, a second type of error seems to be due to the process of infobox extraction when generating DBpedia. These errors usually lead to resource misclassification problems. For example, `dbpedia:Timeline_of_the_presidency_of_John_F._Kennedy` is classified as a person, even though it is actually a timeline. Finally, a third type of error is caused by Wikipedia inconsistencies between the infobox and the article,[5] or from documents from which these articles were created.[6] Table 3 shows error repartition for the class `dbpedia:Person`.

Table 3. Repartition of errors in the DBPedia class Person

Class	duplicate	misclassification	others
`dbpedia:Person`	31	75	16

An exhaustive analysis of the experiment results on every DBpedia class is out of the scope of this article. These results are available online in RDF.[7] This method can be reproduced on any dataset without any prior knowledge of the data.

5.2 Datasets Interlinking

The data interlinking problem can be formulated as follows: given two distinct datasets, which resources represent the same real-world objects? This problem is fully described in [4].

The discovery of keys (and pseudo-keys) in datasets can be helpful for the task of interlinking when combined with ontology matching techniques [5]. More specifically, we propose the following approach:

1. use the algorithm to detect keys in two datasets,
2. use an ontology matcher to find equivalent properties in the two datasets,
3. find instances that violate keys made up of equivalent properties,
4. relate these instances by means of `owl:sameAs`.

This is in line with the framework presented in [6], in the context of the Datalift project.[8] Below we illustrate the above process on the basis of the two datasets Drugbank and Sider.

[4] Query executed on the DBPedia SPARQL endpoint `http://dbpedia.org/sparql`

[5] See for example `http://dbpedia.org/resource/Phromyothi_Mangkorn` and `http://dbpedia.org/resource/Kraichingrith_Phudvinichaikul`

[6] See for example `http://dbpedia.org/resource/Merton_B._Myers` and `http://dbpedia.org/resource/William_J._Pattison` and the footnote at the end of these articles.

[7] `http://data.lirmm.fr/keys`

[8] `http://datalift.org`

Drugbank[9] and Sider[10] are two datasets on drugs. We would like to interlink drugs described by the classes `drugbank:drugs` and `sider:drugs`. The datasets contain, respectively, 4772 and 924 drugs in their RDF versions[11] described by 108 and 10 properties, respectively. Execution of our algorithm returned the keys shown in Table 4(a) and Table 4(b) and ordered by decreasing support.

Table 4. Key discovery for `drugbank:drugs` and `sider:drugs`

(a) Keys of `drugbank:drugs`

Properties of the key	Support
`foaf:page`	1
`db:genericName`	1
`db:primaryAccessionNo`	1
`db:updateDate`	1
`rdfs:label`	1
`db:limsDrugId`	1

`db:smilesStringCanonical`
`db:drugType`
`db:pubchemCompoundId`
`db:creationDate`
	0.928

`db:pubchemCompoundId`
`db:drugType`
`db:creationDate`
`db:smilesStringIsomeric`
	0.928

`db:pubchemSubstanceId`	0.922

(b) Keys of `sider:drugs`

Properties of the key	Support
`si:siderDrugId`	1
`si:drugName`	1
`foaf:page`	1
`rdfs:label`	1
`si:stitchId`	1
`si:sideEffect`	0.965
`rdfs:seeAlso`	0.848

Notice that the properties `drugbank:genericName` and `sider:drugName` are keys for the classes `drugbank:drugs` and `sider:drugs`, respectively. Note also that these properties are equivalent (in practice, ontology matchers can help to automatically discover equivalences between different properties). Our proposal is to identify instances that have the same values for `drugbank:genericName` and `sider:drugName`. This identification can be materialized by means of the property `owl:sameAs`. Furthermore, the property `rdfs:label` is also a key for the two datasets. Thus, `rdfs:label` is a potential candidate for interlinking the datasets. The property `foaf:page` is a key in the two datasets too. This means that each drug has a web page in each dataset. This property, however, is not useful for interlinking as the URL of these pages are hard to compare.

6 Related Work

The use of keys and functional dependencies for quality analysis and reference reconciliation of RDF data on the Web is attracting a lot of attention in the Semantic Web community.

[9] `http://www.drugbank.ca/`
[10] `http://sideeffects.embl.de/`
[11] See `http://www4.wiwiss.fu-berlin.de/drugbank` and `http://www4.wiwiss.fu-berlin.de/sider/`

The extraction of key constraints for reference reconciliation has been tackled by Symeonidou et al. [7]. In this work the authors introduce KD2R as a method for automatic discovery of keys in RDF datasets. KD2R is based on the Gordian technique which allows to discover composite keys in relational databases with a depth-first search strategy. However, no experimental results concerning the run-time efficiency and scalability of the proposed algorithm are provided. The biggest dataset tested with KD2R contains only 3200 instances, whereas our algorithm has been tested with DBpedia with more than 1.5 million of instances.

Song and Heflin also rely on key discovery for data interlinking [8]. Their definition of a key is different from the one proposed in this paper. It is based on the notions of coverage and discriminability of a property; the coverage of a property is defined as the ratio of the number of instances of a class having that property to the total number of instances of that class; the discriminability of a property is the ratio of the number of distinct values for the property to the total number of instances having that property. Song and Heflin do not consider conjunction of properties, but single properties. A property is a key if it has coverage and discriminability equal to 1.

Instead of key constraints, Yu and Heflin rely on functional dependencies for quality analysis in RDF datasets [9]. In order to adapt the classical notion of functional dependencies in relational databases to the singularities of RDF datasets, the authors introduce the notion of value-clustered graph functional dependency. Nonetheless, keys are not considered for quality analysis as they are pruned by their algorithm.

7 Conclusions

In this paper we have introduced a method for analyzing web datasets based on key dependencies. We have adapted the definition of a key in relational databases to RDF datasets and introduced the notion of the support of a key. The practical interest of computing "approximate" keys led us to define pseudo-keys on the basis of a discriminability threshold.

We have implemented an algorithm for the discovery of keys and pseudo-keys in RDF datasets. Even for a big dataset such as DBpedia, the runtime of this algorithm is reasonable. This is thanks to pruning techniques based on minimal support criterion and redundancy elimination.

Two applications are described: datasets interlinking, and duplicate and error detection in datasets. We have shown these on computed keys and pseudo-keys of DBpedia. Although a lot of work remains to be done, these applications show the potential of the proposed method.

As future work, we plan to optimize the algorithm by reasoning over class and property hierarchies. The choice of support and discriminability thresholds is not a trivial task, and we would like to look into it. For instance, a too high discriminability may lead to missing interesting pseudo-keys, while, on the other hand, a too low discriminability may lead to discovering very generic and meaningless pseudo-keys. The task of data interlinking is specially interesting

and we also plan to fully develop the approach based on key discovery and property matching described in this paper.

Acknowledgements. This work is supported under the grant TIN2011-28084 from the Ministry of Science and Innovation of Spain, co-funded by the European Regional Development Fund, and under the Datalift (ANR-10-CORD-009) and Qualinca (ANR-2012-CORD-012) projects, sponsored by the French National Research Agency.

The authors would like to thank Daniel Vila-Suero for his helpful comments during the preparation of this paper.

References

1. Mannila, H., Raiha, K.-J.: Algorithms for inferring functional dependencies from relations. Data & Knowledge Engineering 12, 83–99 (1994)
2. Huhtala, Y., Kärkkäinen, J., Porkka, P., Toivonen, H.: Tane: An efficient algorithm for discovering functional and approximate dependencies. Comput. J. 42(2), 100–111 (1999)
3. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. In: Proceedings of the WWW 2009 Workshop on Linked Data on the Web. CEUR Workshop Proceedings, vol. 538. CEUR-WS.org (2009)
4. Ferrara, A., Nikolov, A., Scharffe, F.: Data linking for the Semantic Web. Int. J. Semantic Web Inf. Syst. 7(3), 46–76 (2011)
5. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
6. Scharffe, F., Euzenat, J.: MeLinDa: an interlinking framework for the web of data. CoRR abs/1107.4502 (2011)
7. Symeonidou, D., Pernelle, N., Saïs, F.: KD2R: A Key Discovery Method for Semantic Reference Reconciliation. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2011 Workshop. LNCS, vol. 7046, pp. 392–401. Springer, Heidelberg (2011)
8. Song, D., Heflin, J.: Automatically Generating Data Linkages Using a Domain-Independent Candidate Selection Approach. 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. 649–664. Springer, Heidelberg (2011)
9. Yu, Y., Li, Y., Heflin, J.: Detecting abnormal semantic web data using semantic dependency. In: Proceedings of the 5th IEEE International Conference on Semantic Computing (ICSC 2011), Palo Alto, CA, USA, September 18-21, pp. 154–157. IEEE (2011)

Ranking RDF with Provenance via Preference Aggregation

Renata Dividino, Gerd Gröner, Stefan Scheglmann, and Matthias Thimm

Institute for Web Science and Technologies (WeST),
University of Koblenz-Landau, Germany
{dividino,groener,schegi,thimm}@uni-koblenz.de

Abstract. Information retrieval on RDF data benefits greatly from additional provenance information attached to the individual pieces of information. Provenance information such as origin of data, certainty, and temporal information on RDF statements can be used to rank search results according to one of those dimensions. In this paper, we consider the problem of aggregating provenance information from different dimensions in order to obtain a joint ranking over all dimensions. We relate this to the problem of preference aggregation in social choice theory and translate different solutions for preference aggregation to the problem of aggregating provenance rankings. By exploiting the ranking orderings on the provenance dimensions, we characterize three different approaches for aggregating preferences, namely the lexicographical rule, the Borda rule and the plurality rule, in our framework of provenance aggregation.

1 Introduction

When querying the Web we are faced with highly varying quality of information. Techniques to find the relevant information out of the Web data should include ways to investigate the value of information. Provenance provides knowledge that can be used to quantify this value, and may come in different forms, e. g., trustworthiness of a source, time of validity, certainty and vagueness. Determining the top-k answers of a query that includes provenance information is an emerging problem.

Recent works [7,22,8,19] have proposed frameworks for representing provenance structures. Provenance is associated to RDF statements in form of *annotations* and the different forms of provenance, denoted as *annotation dimensions*, are captured by an algebraic structure [11], which enables annotation propagation through query evaluation. Given a set of query results where each tuple is associated with annotations, we consider the problem of ranking the tuples according to the annotation dimensions. Usually, determining the top-k results when considering only one annotation dimension, for instance the temporal dimension, is done by sorting the results according to *all* the time values in order of importance (increasing or decreasing order). Ranking query results when an aggregation of many annotation dimensions is necessary, e. g., temporal and fuzzy, poses a variety of further challenges.

In this paper, we relate the problem of ranking stemming from an aggregation of different annotation dimensions to the problem of *preference aggregation* (or *judgement*

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 154–163, 2012.

Table 1. The set of annotated RDF statements in Ex. 1

ID	Statement	Time
#1	[Cinemark Palace plays The Grey]	05.05.12
#2	[Cinemark Palace plays Man on a Ledge]	05.05.12
#3	[Prado Cinema Cafe plays Underworld: Awakening]	06.06.12
#4	[Deerfield Cinema plays Man on a Ledge]	02.03.12
#5	[The Grey hasGenre Action]	05.05.12
#6	[Man on a Ledge hasGenre Action]	02.03.12
#7	[Underworld: Awakening hasGenre Action]	02.03.12

aggregation) in social choice theory [15]. By adopting methods from preference aggregation, we formulate a general framework for annotation dimension aggregation that is based on solid formal grounds. We translate different solutions for preferences aggregation to the problem of aggregating provenance rankings. By exploiting the ranking orderings on the provenance dimensions, we characterize three different approaches for aggregating preferences, namely the lexicographical rule, the Borda rule, and the plurality rule, in our framework of provenance aggregation.

The combination of provenance has been considered before in e. g. [17] where the focus is on the representation of the combined annotation dimensions based on their possible dependency interactions. In [24], the aggregation of results from two specific dimensions is presented, but, to our knowledge, the problem of a general aggregation approach of multiple dimensions for top-k ranking has not been considered yet.

This paper is organized as follows. Section 2 describes foundations of representing and querying RDF with provenance. Section 3 presents the formal framework for ranking taking into consideration all annotation dimensions. In Section 4, we discuss how the aggregation is applied in an offline setting and for streaming data. We compare our work with related ones in Section 5 and conclude with a summary in Section 6.

2 Foundations

We follow the representation for annotated RDF used in [8]. Let p be some annotation dimension (such as *time*) and Ω_p its set of possible values, e. g., the set of all valid dates. Then, an annotated RDF statement is a quadruple $\alpha : \theta$ where α is an RDF statement and $\theta \in \Omega_p$ the attached annotation.

Example 1. Table 1 shows some RDF statements about movie theaters and films. Each statement is annotated with an element from the temporal annotation dimension. In the first column, we represent the statements in a simplified textual syntax. The second column shows the annotation value $\theta \in \Omega_p$ assigned to the statement. For example, the statement #1 states that the movie theater *Cinemark Palace* plays the film *The Grey*. The annotation value assigned to this statement is the timestamp 05.05.12. saying that this information has been published on this date.

We assume that annotation dimensions are represented in form of commutative *semirings* [11]. Let $K_p = (\Omega_p, \otimes_p, \oplus_p, \top_p, \bot_p)$ be a commutative and idempotent semiring of an annotation dimension p where Ω_p is the value domain of p, $\otimes_p$, $\oplus_p$ are custom user-defined binary functions on Ω_p, and $\top_p$ and $\bot_p$ are top and bottom elements. On K_p we can define a *partial order* $\preceq_p$ on the values in Ω_p. For all $\theta_1, \theta_2 \in \Omega_p$, $\theta_1 \preceq_p \theta_2$ (θ_1 precedes θ_2) if and only if there is $\theta_3 \in \Omega_p$ such that $\theta_1 \oplus \theta_3 = \theta_2$. We write $\theta_1 \equiv \theta_2$ if both $\theta_1 \preceq \theta_2$ and $\theta_2 \preceq \theta_1$. For technical convenience, we only consider annotation dimensions p such that the order $\preceq_p$ is a *total preorder*.

Example 2. Let Ω_p be the set of all possible dates, $\otimes_p$ be the minimum function and $\oplus_p$ the maximum function, then we obtain the natural linear order $\preceq_p$ for dates, i. e., for $\theta_1, \theta_2 \in \Omega_p$ it holds that $\theta_1 \preceq_p \theta_2$ if and only if the date represented by θ_1 is not later than the date represented by θ_2.

There are various query languages [8,24] for annotated RDF graphs that have been developed on top of SPARQL [21]. Those languages extend the algebra operators of SPARQL to compute annotation values of results, i. e., enable annotation propagation through queries using the algebraic operators of the underlying annotation structure. Basically, annotation values for some annotation dimension p can be propagated through query evaluation along *how-provenance* [11], i. e., annotation values derived from the individual derivation trees used to assign the variables. For a query q and an RDF graph G, we define $\Phi(r) \in \Omega_p$ to be the annotation value for a query result r in the result set $res(G, q)$. We refer to [8] for a complete formalization of how $\Phi(r)$ is determined.

Example 3. Let q be a query that asks for all movies theaters that play some action movie[1]. Querying the (annotated) RDF graph G from Ex. 1 with query q yields the result set $res(G, q) = \{r_1, r_2, r_3\}$ where $r_1 = \langle$Cinemark Palace$\rangle$, $r_2 = \langle$Prado Cinema Cafe$\rangle$, and $r_3 = \langle$Deerfield Cinema$\rangle$. Given the semantics of the semirings operators $\oplus$ and $\otimes$ in Ex. 2 and since r_1 is derived by joining the statements #1 and #5 or #1 and #6, its the provenance value is be obtained by evaluating the respective annotation values $(05.05.12 \otimes 05.05.12) \oplus (05.05.12 \otimes 02.03.12) = 05.05.12$, which corresponds to the most cautious but valid timestamp r_1 bases upon.

Finally, the order $\preceq_p$ on the annotation values of the annotation dimension p can be exploited to obtain a (partial) ranking on the results $res(G, q)$ for some query q. To do so, we extend the order $\preceq_p$ on $res(G, q)$ by defining

$$r \preceq_p r' \qquad \text{if and only if} \qquad \Phi(r) \preceq_p \Phi(r')$$

for all $r, r' \in res(G, q)$.

Example 4. In Ex. 3 $\{r_1, r_2, r_3\}$ are returned in the order $\langle r_3, r_1, r_2\rangle$, which reflects the recency of the statements those results are based upon.

3 Multiple Annotation Dimensions and the Aggregation Problem

Now we extend the concept of annotated RDF to allow for more than a single annotation dimension. As ranking according to various annotation dimensions may lead to different ranking orderings of answers, we formulate the problem of ranking aggregation.

[1] In SPARQL syntax:"SELECT ?*x* WHERE {?*x plays* ?*y* . ?*y hasGenre* '*Action*'.}"

Table 2. The set of annotated RDF statements in Ex. 5

ID	Statement	Time	Source	Certainty
#1	[Cinemark Palace plays The Grey]	05.05.12	City Guide	0.8
#2	[Cinemark Palace plays Man on a Ledge]	05.05.12	City Guide	0.8
#3	[Prado Cinema Cafe plays Underworld: Awakening]	06.06.12	Movie Today	0.6
#4	[Deerfield Cinema plays Man on a Ledge]	02.03.12	Cinema On	0.7
#5	[The Grey hasGenre Action]	05.05.12	City Guide	0.7
#6	[Man on a Ledge hasGenre Action]	02.03.12	Cinema On	0.7
#7	[Underworld: Awakening hasGenre Action]	02.03.12	Movie Today	0.7

3.1 Annotated RDF with Multiple Dimensions

Until now, we only consider a single annotation for each RDF statement. However, statements may be annotated along multiple dimensions such as source, time, and certainty. Assume $\Gamma = \{p_1, \ldots, p_\gamma\}$ is a fixed set of *independent annotation dimensions* with $|\Gamma| = \gamma$ and let $K_{p_i} = (\Omega_{p_i}, \otimes_{p_i}, \oplus_{p_i}, \top_{p_i}, \bot_{p_i})$, for $i = 1, \ldots, \gamma$, be the corresponding annotation structures. We extend the definition of an annotated RDF statement accordingly. An annotated RDF statement is a tuple $S = \alpha : \theta_1, \ldots, \theta_\gamma$ with α being an RDF statement and $\theta_i \in \Omega_{p_i}$ an annotation. Similarly, the function $res(G, q)$ is extended to return tuples annotated with multiple annotations, where the annotation value in each dimension is determined separately using the approach from the previous section.

Example 5. We extend the annotations of the set of RDF statements from Table 1 along the independent dimensions $\Gamma = \{source, time, certainty\}$, cf. Table 2. The first row states that the movie theater *Cinemark Palace* plays the film *The Grey*, and that this statement was received from 'City Guide', has been published on '05.05.12' and with certainty degree of 0.8.

Each single annotation dimension of an annotated RDF graph can be exploited for ranking results as discussed above. In general, those rankings do not coincide and depend on the chosen dimension. We are interested in a joint ranking taking *all* annotation dimensions into account.

Example 6. We assume that the RDF statements presented in Table 2 represent all information available on the Web. The query and query results presented in Ex. 3 correspond to the search request and its results respectively. We can exploit the orders $\preceq_p$ to obtain (partial) ranking ordering on the results. Suppose $\preceq_{\text{time}}$ corresponds to the natural linear order on dates, $\preceq_{\text{certainty}}$ corresponds to the order on (fuzzy) certainty values, and for the source dimension, *Movie Today*, *Cinema On*, and *City Guide* represents the set of all the possible values and we assume that *Movie Today* $\preceq_{source}$ *City Guide*, *Cinema On* $\preceq_{source}$ *City Guide* and both *Movie Today*, *Cinema On* are equally reliable. The ranking ordering of the results considering the temporal dimension is $\langle r_3, r_1, r_2 \rangle$, which reflects the recency of the statements those results are based upon. For the certainty dimension we have $\langle r_3, r_2, r_1 \rangle$, which reflects the vagueness of the statements, and for the source dimension we have $\langle (r_2, r_3), r_1 \rangle$, which reflects the reliability of the sources.

3.2 Aggregating Annotation Rankings

In the following, we consider the problem of how results of a query should be ranked according to an aggregation of independent annotation dimensions. Therefore, we wish to aggregate the annotation rankings into a single ranking ordering. This is a well-known problem in the field of *judgement aggregation* (or *preference aggregation*) [2,15]. There, the problem under consideration is to aggregate interests or votes in order to come up with a decision that is favorable in the light of the individual interests. A well-known application for social choice theory is the problem of *voting*, i. e., of constructing a voting mechanism that is, in some sense, fair. In order to relate our approach to those works, we borrow some properties that were originally stated for social choice theory and investigate them in our framework of aggregating annotation rankings.

Let G be an annotated RDF graph, $\Gamma = \{p_1, \ldots, p_\gamma\}$ a set of annotation dimensions, and q some query for G. As discussed above, querying G with q yields a set of tuples $res(G, q)$, and each tuple is annotated with some annotation value for every annotation dimension in Γ. Using the rankings $\preceq_{p_1}, \ldots, \preceq_{p_\gamma}$ the set $res(G, q)$ can be ordered in different ways. We call $P = \{\preceq_{p_1}, \ldots, \preceq_{p_\gamma}\}$ the *annotation profile* of $res(G, q)$.

Definition 1 (Profile aggregator). *Let $P = \{\preceq_{p_1}, \ldots, \preceq_{p_\gamma}\}$ be an annotation profile of $res(G, q)$. A mapping $\preceq_{p_1}, \ldots, \preceq_{p_\gamma} \mapsto \preceq$, such that $\preceq$ is a partial order on $res(G, q)$, is called an* annotation aggregator.

In the field of judgement aggregation, desirable properties and different mechanisms for defining the aggregation of multiple orderings have been investigated for more than 50 years. One of the most important technical results is *Arrow's impossibility theorem* [2], which states that there is no such thing as a fair voting mechanism or "every voting mechanism is flawed". More precisely, there is no mapping $\mapsto$ satisfying the following three properties (let $\preceq$ be defined via the mapping $\preceq_{p_1}, \ldots, \preceq_{p_\gamma} \mapsto \preceq$):

Pareto-efficiency For every $r, r' \in res(G, q)$, if $r \preceq_{p_i} r'$ for all $i = 1, \ldots, \gamma$ then $r \preceq r'$.
Non-dictatorship There is no $i \in \{1, \ldots, \gamma\}$ such that $\preceq_{p_i} = \preceq$ for *every* profile.
Independence of irrelevant alternatives If for two sets $\{\preceq_{p_1}, \ldots, \preceq_{p_\gamma}\}$ and $\{\preceq'_{p_1}, \ldots, \preceq'_{p_\gamma}\}$ and every $i = 1, \ldots, \gamma$ it holds $r \preceq_{p_i} r'$ whenever $r \preceq'_{p_i} r'$ then $r \preceq r'$ whenever $r \preceq' r'$ (with $\preceq_{p_1}, \ldots, \preceq_{p_\gamma} \mapsto \preceq$ and $\preceq'_{p_1}, \ldots, \preceq'_{p_\gamma} \mapsto \preceq'$).

In the following, we consider three examples of profile aggregators, which are inspired by approaches to solve the problem of judgement aggregation in social choice theory. A simple profile aggregator (that fails to satisfy *non-dictatorship*) is the *lexicographical aggregator*. The lexicographical aggregator assumes some given total order on the annotation dimensions, e. g., the temporal information may be more important than the source information. Given this, a result item r' is preferred to r if r' is preferred to r wrt. the given timestamps ($r \preceq_{time} r'$) or r and r' have equal timestamps ($r \equiv_{time} r'$) but r' is preferred to r wrt. the source information ($r \preceq_{source} r'$). Note, that this kind of annotation aggregator order is heavily biased by the given ordering of dimensions.

Definition 2 (Lexicographical aggregator). *Let $\{\preceq_{p_1}, \ldots, \preceq_{p_\gamma}\}$ be an annotation profile on $res(G, q)$, and assume w.l.o.g. that $\langle \preceq_{p_1}, \ldots, \preceq_{p_\gamma} \rangle$ is the order of importance of*

the annotation profile ($\preceq_{p_1}$ being the most important ranking). Then the lexicographical aggregation $\preceq_l$ *of* $\preceq_{p_1}, \ldots, \preceq_{p_\gamma}$ *is defined via*

$$r \preceq_l r' \;\; \textit{iff} \;\; \neg\exists k \in \{1, \ldots, \gamma\} : \Big(r \not\preceq_{p_k} r' \wedge \forall i \in \{1, \ldots, \gamma\} : i < k \Rightarrow r \preceq_{p_i} r' \wedge r' \preceq_{p_i} r\Big)$$

for all $r, r' \in res$.

Example 7. Given the annotation profiles presented in Ex. 6, and we assume that $\langle \preceq_{certainty}, \preceq_{time}, \preceq_{source} \rangle$ is the order of importance of the annotation profiles. Then the lexicographic ordering of r_1, r_2, and r_3 corresponds to $r_3 \preceq_l r_2 \preceq_l r_1$. This represents exactly the ranking ordering when considering just the certainty dimension. Now, we assume that $\langle \preceq_{source}, \preceq_{time}, \preceq_{certainty} \rangle$ is the order of importance of the annotation profiles, then r_1, r_2, and r_3 are ordered in $r_2 \preceq_l r_3 \preceq_l r_1$.

Another popular aggregation method from voting theory is the *Borda rule*. The Borda rule defines preferences by assigning a value to each tuple according to its positions in the domain order. The sum of all the values of a tuple represents its score.

Definition 3 (Borda aggregator). *Let* $\{\preceq_{p_1}, \ldots, \preceq_{p_\gamma}\}$ *be an annotation profile on* $res(G, q)$. *For each* $r \in res(G, q)$ *define* $sc(r)$ *via*[2]

$$sc(r) = \sum_j \sum_{r' \in res(G,q) \setminus \{r\}} [r' \preceq_{p_j} r]$$

for all $r \in res(G, q)$. *Then the* Borda aggregation $\preceq_b$ *of* $\preceq_{p_1}, \ldots, \preceq_{p_\gamma}$ *is defined via*

$$r \preceq_b r' \quad \textit{iff} \quad sc(r) \leq sc(r')$$

for all $r, r' \in res$.

Example 8. Given the annotation profiles presented in Ex. 6, then the Borda ordering corresponds to $r_2 \preceq_b r_3 \preceq_b r_1$. The Borda score of r_1 is 5 (at the temporal dimension score 1, at the certainty dimension score 2, and at the source dimension 2, since r_1 is derived from the most reliable source). The Borda score of r_3 is 3 (2, 0, and 1 at temporal, certainty, and source dimension respectively) and of r_2 is 2 (0, 1, and 1 at temporal, certainty, and source dimension).

At last, we consider the *plurality* aggregation method, which defines the preferred element to be the tuple in the result set with the most votes (plurality), i. e., the tuple r which is the most preferred considering all annotation profiles.

Definition 4 (Plurality aggregator). *Let* $\preceq_{p_1}, \ldots, \preceq_{p_\gamma}$ *be an annotation profile on* $res(G, q)$. *Then the* plurality aggregation $\preceq_m$ *of* $\preceq_{p_1}, \ldots, \preceq_{p_\gamma}$ *is defined via*

$$r \preceq_m r' \quad \textit{iff} \quad |\{i \mid r \preceq_{p_i} r'\}| \geq |\{i \mid r' \preceq_{p_i} r\}|$$

for all $r, r' \in res$.

[2] Note that $[A]$ is the Iverson bracket defined via $[A] = 1$ if A is true and $[A] = 0$ otherwise.

Note that the plurality aggregator defined above suffers from the *Condorcet paradox* if more than two rankings are to be aggregated [10]. That is, even if $\preceq_1, \ldots, \preceq_\gamma$ are partial orders, the relation $\preceq_m$ may contain cycles.

Example 9. Let $res = \{r_1, r_2, r_3\}$ and consider $\preceq_1$, $\preceq_2$, and $\preceq_3$ defined via

$$r_1 \preceq_1 r_2 \preceq_1 r_3 \qquad r_2 \preceq_2 r_3 \preceq_2 r_1 \qquad r_3 \preceq_3 r_1 \preceq_3 r_2$$

and observe that $r_1 \preceq_m r_2$, $r_2 \preceq_m r_3$, and $r_3 \preceq_m r_1$.

Example 10. Given the annotation profiles of the certainty and source dimensions presented in Ex. 6, then the plurality ordering for r_1, r_2, and r_3 corresponds to $r_2 \preceq_b r_3 \preceq_b r_1$ since $r_2 \preceq_p r_1$ and $r_3 \preceq_p r_1$ for all the dimensions and $r_2 \preceq_{certainty} r_3$, and $r_2 \equiv_{source} r_3$.

4 Ranking of Stream Data

The proposed profile aggregators can be applied in offline and online settings. Offline setting means that the whole set of results is completely available, while online setting refers to the aggregation of streaming data. In the following, both settings are extensively discussed.

4.1 Offline Setting

First, we perform top-k querying on locally stored annotated RDF graphs rather than on-the-fly, i. e., the whole result set $res(G, q)$ of a query q is known a priori. As one or multiple annotation dimensions can be used for ranking, in the following, we shortly describe how ranking on annotated RDF can be done:

One-dimensional Approach. Providing top-k results when considering only one annotation dimension, for instance the temporal dimension, is done straightforward by sorting the results in order of importance (increasing or decreasing order).

Multi-dimensional Approach. A top-k ranking over multiple annotation dimensions requires an aggregation of the top-k rankings from each dimension. According to Sec. 3.2, the profile aggregators offer a variety of aggregation means with respect to individual preferences among the different annotation dimensions. Therefore, to obtain a joint ranking taking *all* annotation domains into account, we use one of the described annotation aggregators (e. g., the lexicographical rule, the Borda rule, or the plurality rule) to aggregate results from multiple annotation dimensions into a single ordering.

4.2 Online Setting

In contrast to offline ranking, in stream ranking $res(G, q)$ is not available at once, we rather receive the result tuples $r \in res(G, q)$ continuously whereas r_t is the tuple received at time t. We want to start presenting results up the first available tuple. Such a stream ranking mechanism could not decide upon receipt if a result tuple r_t is part of final the top-k results. It could only decide if a tuple r_t is part of the top-k results at time t because there could be better results not received at time t.

One-dimensional Approach. A simple approach for ranking over streams starts with an empty top-k result set res_k. New result tuples r_t are added to res_t until the result set is full (k tuples in res_k). We suppose that the k tuples are sorted wrt. a given criteria, for instance, in decreasing order. Let r_b, denoted as the border-tuple, be the lowest/smallest (the k-th) tuple. All tuples r_t received from this point have to be compared with the border-tuple r_b. If $r_t \preceq r_b$, the tuple r_t could be ignored otherwise r_b is removed and r_t is added to res_k. In this case, a new border-tuple has to be computed.

Multi-dimensional Approach. For the aggregation over multiple dimensions, the streaming algorithm remains the same if the ordering of the elements in res_k does not change when additional elements are received, i.e., if the *independence of irrelevant alternatives* holds. This property holds for several aggregators, e.g., for the lexicographic aggregator. The Borda aggregator, however, is an example where is is not possible to determine an aggregation without considering the whole set of results.

Example 11. In Ex. 8 $\langle r_1, r_3, r_2 \rangle$ corresponds to the top-3 result with respect to the Borda aggregator. As already mentioned, pairwise element comparison is not applicable to the Borda aggregator. To illustrate that, let us assume that we are looking for the top-2 results and that the elements are received in the following order, r_1, r_2, r_3. The top-2 results should be $r_3 \preceq_b r_1$. Like in the one-dimensional case, we fill our result set with the first two elements, i.e., $\{r_1, r_2\}$. Then we compare these elements to determine the border element to be r_2. Next, we compare this border element to our last element r_3. Since, it ranks r_2 and r_3 as equal, we are not able to determine if we have to replace the border element r_2 with r_3 or keep it. It is not possible to determine our top-2 result precisely, and only the top-2 result sets $\{r_1, r_2\}$ and $\{r_1, r_3\}$.

5 Related Work

Ranking of query results has been considered from different perspectives in the database research. In Agrawal et al. [1] the ranking criteria is based on a similarity measure between terms in the query condition and terms of tuple attributes in tables. A similar principle is applied by Fuhr [9], where ranking techniques from information retrieval are used to rank query results on databases. This approach relies on a relevance feedback from the user. Ilyas et al. [14,13] give an overview of rank-aware join operations. While in the top-k query result ranking the results are ranked according to attribute values of result tuples, e.g,. by using order-by construct on attributes, the top-k join ranking specifies ranking scores based on multiple relations. Natsev et al. [20]. propose a a heuristic algorithm over multiple ranked data sources to efficient combining ranked results from single dimensions using. None of these approaches rank over RDF data and they do not consider the particular problem of aggregation of different independent ranking dimensions.

From the ranking perspective, several approaches consider also the problem of combining several dimensions of ranking criteria [6,16,23,12]. Preferences are specified in terms of partial orders on attributes and they can be accumulated and aggregated to build complex preference orders. In [6], the general idea is to rank query results when there is no exact match, but some results are similar to the query. They compute the

distance of attribute values of the relation with respect to the query attributes. In [12], linear sums of attributes are used to rank preferences (assigned to attributes). Likewise, Li et al. [18] presents top-k ranking for different dimensions for relational databases. Compared to our work, none of them considers the ranking of semi-structured data like RDF and their focus is not on ranking w.r.t. annotations of data.

In the realm of the Semantic Web, we compare our work to annotated RDF data in general, and to aggregation principles in particular. Based on semirings [11], Buneman and Kostylev [7] and Straccia et al. [22] present an algebra for RDF annotations. Their approaches are for annotations in general, but they do not consider multiple dimensions simultaneously. Zimmermann et al. [24] present a combination of multiple annotation dimensions. They combine two dimensions by a composition into one dimension, modeled as a compounded annotation dimension. An aggregation function maps annotation dimensions into a set of pairs of annotations. Kostylev et al. [17] also consider the problem of combining various forms of provenance that, analogous to the previous, map annotation dimensions into a set of vectors. They introduce restrictions to semirings in order to define containment and equivalence of combined annotation relations. The latter ones are different from our work since we aggregate annotation dimensions considering aggregation functions, which does not rely on the structure of the annotation dimensions and can be generalized to the aggregation of every ordered set.

Ranking for streaming data is an emerging problem when querying large data collections as on the Web, and it is therefore considered for RDF querying and reasoning. For instance, SPARQL extensions allows ranking query results on streams [4,5], as well as general reasoning frameworks incorporate the management of streaming data [3]. However, ranking with aggregation of multiple annotation dimensions is not studied on streaming RDF data so far.

6 Summary

We have presented a novel approach for top-k querying of RDF data with multiple provenance dimensions. RDF data is annotated with values providing knowledge on the origin, truthworthiness or validity of data and this knowledge should be taken into account when answering queries. Top-k ranking becomes complicated if data have multiple provenance dimensions, and the ranking has to incorporate a holistic ordering over all these dimensions. We have presented an aggregation approach over multiple provenance dimensions, which is based on preference aggregation in social choice theory. We have formalized three different aggregation methods, namely the lexicographical, the Borda and the plurality rule.

Additionally, we consider these aggregations in offline and online settings. For online settings, we first elaborate how the aggregators deal with a continuously enlargement of the available result tuples that are considered in the aggregation. Secondly, we explain which aggregators can be applied in online settings and which not. Further investigations and implementations for efficient ranking on streaming data are planned.

Acknowledgments. The research reported here was partially supported by the SocialSensor FP7 project (EC under contract number 287975).

References

1. Agrawal, S., Chaudhuri, S., Das, G., Gionis, A.: Automated Ranking of Database Query Results. In: CIDR (2003)
2. Arrow, K.J.: A Difficulty in the Concept of Social Welfare. Journal of Political Economy 58(4), 328–346 (1950)
3. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental Reasoning on Streams and Rich Background Knowledge. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 1–15. Springer, Heidelberg (2010)
4. Barbieri, D.F., Braga, D., Ceri, S., Grossniklaus, M.: An Execution Environment for C-SPARQL Queries. In: EDBT, pp. 441–452. ACM (2010)
5. 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)
6. Bruno, N., Chaudhuri, S., Gravano, L.: Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation. ACM Trans. Database Syst. 27(2), 153–187 (2002)
7. Buneman, P., Kostylev, E.: Annotation algebras for RDFS. In: SWPM, CEUR Workshop Proceedings (2010)
8. Dividino, R., Sizov, S., Staab, S., Schueler, B.: Querying for provenance, trust, uncertainty and other meta knowledge in rdf. JWS 7(3), 204–219 (2009)
9. Fuhr, N.: A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In: VLDB, pp. 696–707 (1990)
10. Gehrlein, W.V.: Condorcet's Paradox. Theory and Decision Library C, vol. 40. Springer, Heidelberg (2006)
11. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS, pp. 31–40. ACM, New York (2007)
12. Hristidis, V., Koudas, N., Papakonstantinou, Y.: PREFER: A System for the Efficient Execution of Multi-parametric Ranked Queries. In: SIGMOD, pp. 259–270 (2001)
13. Ilyas, I.F., Beskales, G., Soliman, M.A.: A Survey of Top-k Query Processing Techniques in Relational Database Systems. ACM Comput. Surv. 40(4), 11:1–11:58 (2008)
14. Ilyas, I.F., Shah, R., Aref, W.G., Scott Vitter, J., Elmagarmid, A.K.: Rank-aware Query Optimization. In: SIGMOD, pp. 203–214 (2004)
15. Kelly, J.: Social Choice Theory: An Introduction. Springer (1988)
16. Kießling, W.: Foundations of Preferences in Database Systems. In: VLDB, pp. 311–322 (2002)
17. Kostylev, E.V., Buneman, P.: Combining dependent annotations for relational algebra. In: ICDT, pp. 196–207. ACM, New York (2012)
18. Li, C., Chen-Chuan Chang, K., Ilyas, I.F., Song, S.: RankSQL: Query Algebra and Optimization for Relational Top-k Queries. In: SIGMOD, pp. 131–142 (2005)
19. Lopes, N., Polleres, A., Straccia, U., Zimmermann, A.: AnQL: SPARQLing Up Annotated RDFS. 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. 518–533. Springer, Heidelberg (2010)
20. Apostol, N., Chang, Y.-C., Smith, J.R., Li, C.-S., Vitter, J.S.: Supporting Incremental Join Queries on Ranked Inputs. In: VLDB, pp. 281–290 (2001)
21. Prud'hommeaux, E., Seaborne, A.: Sparql query language for rdf. W3c recommendation, W3C (January 2008)
22. Straccia, U., Lopes, N., Lukácsy, G., Polleres, A.: A General Framework for Representing and Reasoning with Annotated Semantic Web Data. In: AAAI, pp. 1437–1442 (2010)
23. Xin, D., Han, J., Cheng, H., Li, X.: Answering Top-k Queries with Multi-Dimensional Selections: The Ranking Cube Approach. In: VLDB, pp. 463–475 (2006)
24. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated semantic web data. JWS 11(0), 72–95 (2012)

Freshening up while Staying Fast: Towards Hybrid SPARQL Queries

Jürgen Umbrich, Marcel Karnstedt, Aidan Hogan, and Josiane Xavier Parreira

Digital Enterprise Research Institute, National University of Ireland, Galway
firstname.lastname@deri.org

Abstract. Querying over cached indexes of Linked Data often suffers from stale or missing results due to infrequent updates and partial coverage of sources. Conversely, live decentralised approaches offer fresh results directly from the Web, but exhibit slow response times due to accessing numerous remote sources at runtime. We thus propose a *hybrid query* approach that improves upon both paradigms, offering fresher results from a broader range of sources than Linked Data caches while offering faster results than live querying. Our hybrid query engine takes a cached and live query engine as black boxes, where a hybrid query planner splits an input query and delegates the appropriate sub-queries to each interface. In this paper, we discuss query planning alternatives and their main strengths and weaknesses. We also present coherence measures to quantify the coverage and freshness for cached indexes of Linked Data, and show how these measures can be used for hybrid query planning to optimise the trade-off between fresh results and fast runtimes.

1 Introduction

Current query approaches over Linked Data offer either (i) fast query times by materialising local optimised indexes [2,10,3] that cache Web data, but at the cost of potentially incomplete or outdated results; or (ii) fresh results by accessing query relevant data from the Web at runtime, but at the cost of slower query times [4,16,8]. This trade-off between fresh and fast results becomes even more crucial as Linked Data expands and becomes more dynamic.

In previous work [18], we identified and started to address this tradeoff problem, proposing the hybrid query execution architecture as shown in Figure 1. Our envisioned query engine has two SPARQL interfaces to combine (i) the performance of a centralised SPARQL store (henceforth: *cache*) with (ii) the up-to-date results of a live SPARQL engine. We do not build yet another local or live SPARQL engine, nor do we design techniques for either: instead, we propose to take off-the-shelf engines (one live and one cache) as black boxes and investigate the techniques by which they can be combined in a complementary way. Our proposed engine can be seen as adding a live "query wrapper" for existing SPARQL stores, or as adding a SPARQL-enabled cache for live approaches. The core components of our architecture are (i) a *coherence monitor* that computes and stores statistics about the dynamicity and coverage of cache

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 164–174, 2012.

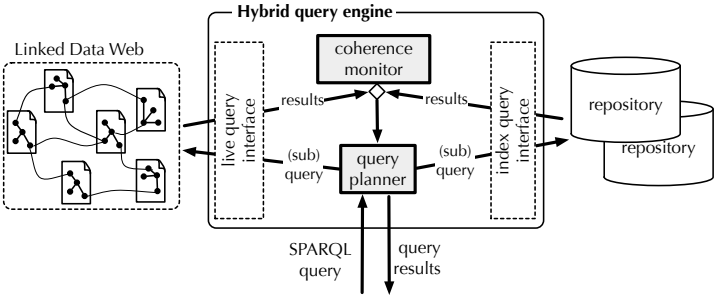


Fig. 1. Architecture of a hybrid query-engine

data; and (ii) a *query planner* that uses these statistics to (amongst other tasks) decide which parts of the query should be run against the cache and which live. By getting the cache to quickly service query patterns for which it has good up-to-date coverage, and by running the remaining patterns of the query live, our hybrid query planner aims to optimise the aforementioned trade-off of fresh vs. fast results; the coherence monitor provides the statistics that make this feasible.

As an abstract motivating example, say that a user asks a Linked Data cache: What are the current temperatures in European capital cities? Also, say that the cache endpoint has up-to-date information on which cities are capitals. However, say that this endpoint does not have knowledge (or perhaps only has partial knowledge) of which continent the cities are on; this information will have to be retrieved live. Furthermore, information about temperatures indexed by the cache endpoint are likely to be old; hence, to avoid stale results, this information should *only* be fetched live. Ideally, our hybrid query approach would use the cache endpoint to quickly return a list of capital cities, go live over the Web to check which ones are European, and then (also live) retrieve the current temperatures of these capitals directly from source. To enable this hybrid execution, we need knowledge of the coverage and freshness of the cache.

Finding an effective trade-off between fresh and fast results by combining live and centralised caches introduces a wide range of challenges. In this work, we contribute towards the realisation of our envisioned hybrid query engine [18] by further investigating two core aspects. First, we mention various alternatives for query planning and review their strengths and weaknesses for a black box scenario in which the participating query engines are not aware of their role. Second, we propose various coherence-estimate formulae needed for generating effective hybrid query plans, comparing them against each other and discussing different approaches to obtain these statistics from a black box centralised cache.

We continue this paper with some background on Linked Data query answering (Section 2). We discuss various alternatives for creating hybrid query plans in Section 3. Next, we propose methods to probe centralised caches so as to collect coherence estimates, and present results for two public Linked Data SPARQL engines (Section 4). We then conclude and discuss future work (Section 5).

2 Background

Centralised approaches for SPARQL query execution integrate data from different sources and are designed to provide (i) fast execution times by building optimised indexes and (ii) scalability through vertical partitioning or data distribution. However, constantly maintaining a *broad and fresh* coverage of remote data from millions of sources is unfeasible in such approaches. Current centralised SPARQL endpoints cover selected subsets of Linked Data, e.g., FactForge [2]; or aim to achieve a broad coverage of Linked Data, e.g., OpenLink's LOD cache[1] and the Sindice [10] SPARQL endpoint[2] (both powered by Virtuoso [3]).

Recently, various authors have proposed methods for executing SPARQL queries "live", by accessing remote data *in situ* and at runtime [8,10,14,16,4]. This guarantees up-to-date and complete results wrt. to the accessed data. Ladwig and Tran [8] categorise three variations of such approaches, which are (i) top-down, (ii) bottom-up, and (iii) mixed strategies. Top-down evaluation determines remote, query-relevant sources using a *source-selection index*, such as inverted-index structures [10], query-routing indexes [14], or lightweight hash-based structures [16]. Bottom-up query evaluation strategies discover relevant sources on-the-fly during the evaluation of queries, starting from a "seed set" of URIs taken from the query. The seminal proposal in this area is called *link-traversal based query execution* (LTBQE) [4]. In the third strategy [8], an initial seed list potentially containing more sources than actually relevant is generated in a top-down fashion. Additional relevant sources are discovered by using a bottom-up approach. In a sense, our hybrid-query proposals follow this strategy.

While the above approaches access raw data sources directly, an orthogonal approach to live querying is that of federated SPARQL, where queries are executed over a group of possibly remote endpoints [12,13,1]. Like in our proposal, federation involves splitting and delegating sub-queries to different engines. Federated SPARQL approaches either use service descriptions that are indexed locally and used to route queries [12,13], or rely on user-specified service URIs [1].

Recent works look to combine local (i.e., centralised) and remote (i.e., live) querying on a theoretical, engineering and social level (e.g., [5,18]). However, to the best of our knowledge, no one has looked at deciding which *parts* of a query are suitable for local/remote execution. Our work also relates to research on guaranteeing (Web) cache coherence [11] and semantic caching [7]. However, such systems typically rely entirely on cached data or completely discard it. Various authors have recently discussed invalidation of *internal* SPARQL caches [9,19] but rather focus on local index coherence, not on remote (Web) coherence.

3 Query Planner

Traditional query planning within closed systems focuses on optimising for performance by ordering the execution of query operators to minimise intermediate

[1] http://lod.openlinksw.com/sparql
[2] http://sparql.sindice.com/

results. Such query planning often relies on *selectivity estimates*, which indicate the amount of results a given operation will generate. For hybrid query planning, we wish to optimise for both speed *and* freshness. Thus, analogous to (and in combination with) selectivity estimates that optimise for speed, we need other metrics to optimise for freshness. Recalling that (sub-)queries will often be answered faster by materialised indexes than live engines, in the interest of speed, we wish to push as much of the query processing to the cache endpoint. However, we only want to send requests for which the cache has fresh data available. Hence, along with selectivity estimates, we also need *coherence estimates* to measure how well synchronised the cache is wrt. the Web. We will propose and test concrete measures and techniques for computing coherence estimates for a cache endpoint in Section 4. For now, we introduce high-level approaches for creating hybrid query plans that optimise for both speed and freshness.

Split types The core aim of the query planner is to decide which parts of the input query should go to the cache, which parts should go live, and how results can be combined. In terms of splitting the query, there are two high-level options:

Multi-split. In this approach, there is no restriction placed on how many splits are made in the query or on which parts go where. This is the most general case. However, if the query is split into many parts, processing the query will involve a lot of coordination of interim results and synchronisation between the cache and the live engines.[3] Also, if the cache is accessed through a public interface, it may not allow a sufficient query rate for this approach to work.

Single-split. Another simpler (but more restricted) option is to split the query into two: a cache and a live sub-query. Much less coordination is then required between engines. Assuming nested evaluation, an open question is whether the cache or live request should be run first.[4] We argue that going to the cache first makes more sense: (i) this would only require a single query to be run against the cache's materialised index, (ii) the cache can quickly return initial results, (iii) results from the cache give more information (bindings) to the live engine for finding query-relevant sources from the Web.

Reordering strategies. As per traditional selectivity optimisation, before the query can be split, the query (join) tree needs to be reordered to decide which query primitives are executed first.[5] Here we also wish to optimise for freshness; hence, the ordering can include selectivity for speed and/or coherence for freshness.

Selectivity-based ordering. Query patterns in the join tree are first ordered by selectivity. Selectivities for each pattern can be computed by rule-based

[3] Where supported, SPARQL 1.1 `VALUES` could be helpful to ship bindings, but would still require a synchronisation point for each split.

[4] If both parts of the query are deemed to have low selectivity, they could be run in parallel and hash-joined.

[5] We focus on optimising simple BGPs. Aside from features like `OPTIONAL`, `MINUS` and (`NOT`) `EXISTS`, other query features and optimisations can be layered above.

estimates or variable-counting techniques [13]), from analysis of prevalence of patterns in Web data, by sampling data from the cache using probe queries, or (where supported) by posing SPARQL 1.1 `COUNT` queries or queries for statistical summaries against the cache indexes. Patterns that match and return fewer data are executed earlier to minimise interim results.

Coherence-based ordering. Query patterns are ordered by the coherence of cache data available for them; coherence gauges the coverage and freshness of cache indexes for answering a pattern (more coherence implies broader and fresher cache coverage). Coherence measures can be computed for patterns based on analysis of the dynamicity of Web data, using probe queries against the cache which can be compared with live results, or (assuming cooperation of the cache) by listening for updates to the cache indexes [19]. A single-split strategy is appropriate for this ordering, where patterns that can be best answered by the cache will be executed first and the rest then answered live.

Both orderings have inherent advantages and disadvantages. The selectivity-based ordering should minimise interim results but makes no guarantees about freshness. In particular, (in)coherent patterns will be mixed throughout the query tree. This raises another crucial question for the query planner: *where to split a query*. One option is to apply selectivity ordering but use coherence estimates to decide split positions, where either (i) a multi-split is used to fully leverage the performance of the cache while guaranteeing freshness or (ii) a single-split is used at the lowest incoherent pattern, which will minimise coordination but end up running coherent patterns live that the cache could answer. For single-splits, one could also use a pre-defined threshold of coherence to support user-defined expectations of freshness.

Conversely, the coherence-based ordering maximises freshness but makes no guarantees about the size of interim results. This ordering lends itself well to a single-split strategy, where all of the highly-coherent patterns can be first sent directly to the cache; for deciding where to split, a threshold can be used as mentioned before, or a simple fixed-position split strategy can be employed (e.g., always send only the least coherent pattern live). However, in this approach, the cache sub-query may have a low selectivity and require the cache to materialise a lot of results. In the best case, high selectivity and high coherence correlate with each other such that there is no conflict in the fresh vs. fast trade-off. However, as we will see for experiments in the next section, this is not necessarily the case.

Finally, we highlight that the hybrid query planner is responsible for ordering, splitting, delegating and executing sub-queries. The sub-queries sent to the cache or to the live engines are then subject to internal optimisations. This is particularly relevant for single-split coherence-ordering, where the sub-query delegated to the cache endpoint will be reordered according to internal selectivities.

In summary, hybrid query planning is a challenging problem and presents many alternative strategies to explore. We will leave further investigation of these alternatives for future work. For now, we focus on the extraction of coherence measures from public cache endpoints, as required for the above methods.

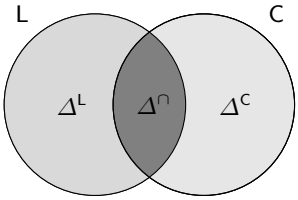


Fig. 2. Result set (Venn) diagram

```
SELECT ?sIn ?pIn ?oOut ?pOut
WHERE {
 ?sIn ?pIn <entityURI> .
 <entityURI> ?pOut ?oOut .
}
```

Fig. 3. Entity-query template

4 Coherence Estimation

In the general case, the degree of optimisation that can be achieved for a query depends on the quality of the statistics available to the query planner. As previously discussed, we first and foremost need coherence estimates to gauge the extent to which, for a given query pattern, the cached indexes are coherent with respect to current information available on the Web. We identify two main approaches to obtain knowledge about coherence for different query patterns.

Cache-independent estimates involve monitoring a large range of Linked Data sources to build a comprehensive, global picture of the dynamicity of the Web of Data. Previous empirical studies [17,15] have shown varying levels of dynamicity across Linked Data sources; furthermore, we speculate that dynamicity varies by the schema of data [17]. In term of benefits, cache-independent estimates can be applied generically to any store (and indeed to other use-cases) [6]; however, they give no indication as to the specific coverage or update rates, etc., of the cache engine at hand.

Cache-specific estimates involve periodically comparing the results of a centralised store against the current version of Linked Data on the Web. Versus a broad empirical study, cache-specific analysis is more sensitive to the particular update patterns and coverage of that store. As shown later in our experiments, analogous coherence estimates can vary for different caches.

For the above mentioned reason, we focus on *cache-specific estimates* where Figure 2 illustrates how the results from the live engine (L) and results from the cached centralised engine (C) may diverge. The cache may return results that live querying does not ($\Delta^{\mathsf{C}} := \mathsf{C} \setminus \mathsf{L}$), some of which may be stale, others of which may be accurate but involve a source that the live engine did not access. Conversely, the live approach may find answers that the cache could not ($\Delta^{\mathsf{L}} := \mathsf{L} \setminus \mathsf{C}$), some of which may be caused by remote data changing, others of which may be from sources that the store did not cache. Some of the results—which we deem to be coherent—are the same for the cache and the Web ($\Delta^{\cap} := \mathsf{L} \cap \mathsf{C}$).

In order to test coherence, we propose to probe both the cache's endpoint and the Web with a broad range of simple queries and compare the results, characterising parts of the cache's index that are likely to be stale or missing.

We view results as consisting of variable–binding pairs (i.e., C, $\mathsf{L} \subset \mathbf{V} \times \mathbf{UL}$ reusing common notation for the set of all query variables, URIs and literals resp.) and we exclude answers involving blank nodes to avoid issues of scoping and inconsistent labelling. We identified two possible methods by which queries can be used to test coherence.

Document-based estimates. Assuming the SPARQL cache uses *Named Graphs* to track the original source of information on the Web, we can compare the data for a Web document against the data cached in the corresponding graph using `GRAPH` queries. However, (and as is the case for the two caches we test later) many stores do not have consistent naming of graphs: sometimes the graph may indeed refer to a particular Web document, but oftentimes the graph will be a high-level URI (e.g., `http://dbpedia.org`), informally indicating a dump from which the data were loaded but which cannot be directly retrieved.

Triple pattern estimates. Thus, we instead focus on triple-based estimates. To ensure lightweight statistics with broad applicability, our notion of coherence for triple patterns centres around predicates. This restricts our approach to triple patterns with a constant as predicate; other patterns are assigned a default estimate. To derive a comparable set of results for generating triple-pattern estimates, we probe the cache and the live engine with a broad set of entity queries (see Figure 3). To quantify the coherence of predicates based on the results, we study two measures. To present these, we apply notation from Figure 2 to the results of the probe queries, adding subscripts to indicate results for a certain query, e.g., Δ^{L}_q. We denote results involving a predicate p as, e.g., $\Delta^{\mathsf{L}}_q(p) := \{r \in \Delta^{\mathsf{L}}_q : (\texttt{?pIn}, p) \in r \vee (\texttt{?pOut}, p) \in r\}$, and say $p \in \Delta^{\mathsf{L}}_q$ iff $\Delta^{\mathsf{L}}_q(p) \neq \emptyset$.

Query-based coherence: The coherence of a predicate p is measured as the ratio of queries for which p appeared in Δ^{L}. For the full set of queries $\mathcal{Q}$, let $M_q(p)$ denote for how many queries a live result involving the predicate was missing at least once in the cache results ($M_q(p) = |\{q \in \mathcal{Q} : p \in \Delta^{\mathsf{L}}_q\}|$). In addition, we count $L_q(p) = |\{q \in \mathcal{Q} : p \in \mathsf{L}_q\}|$ as the queries for which the live engine returned at least one result containing p. The *query-based coherence* of the predicate p is then computed as:

$$\mathsf{coh}_q(p) = 1 - \frac{M_q(p)}{L_q(p)} \quad .$$

Result-based coherence: For this measure, we inspect the ratio of missing results for a predicate p, rather than the fraction of stale queries. Let $M_r(p)$ denote the count of all live results involving the predicate p that were missed by the cache, summated across all queries ($M_r(p) = \sum_{q \in \mathcal{Q}} |\Delta^{\mathsf{L}}_q(p)|$). Let $L_r(p)$ denote the count of all results involving p retrieved by the live engine ($L_r(p) = \sum_{q \in \mathcal{Q}} |\mathsf{L}_q(p)|$). The *result-based coherence* is then:

$$\mathsf{coh}_r(p) = 1 - \frac{M_r(p)}{L_r(p)} \quad .$$

Experimental setup. To test the different coherence-estimate proposals, we conducted experiments with two SPARQL stores that cache a broad range of Linked Data: "the Semantic Web Index" hosted by Sindice, and the "LOD Cache" hosted by OpenLink (see Section 2). We randomly sampled 12 thousand URIs from the 2011 Billion Triple Challenge dataset[6], which contains over 2.1 billion quadruples taken from over 7.4 million RDF/XML documents on 791 pay-level domains.[7] For each URI, we ran the entity query listed in Figure 3 against both caches and live with our LTBQE implementation, giving us a large set of results to compare. Experiments were run in early March 2012; we extracted coherence estimates for 2,550 predicates in OpenLink and 1,627 predicates in Sindice.

Results. We first explored the difference between our two measures. We measured the correlation between both measures across all predicates using Kendall's τ which measures the agreement in ordering for two measures in a range of $[-1, 1]$, where -1 indicates perfectly inverted ordering and 1 indicates the exact same ordering. We found a τ value of 0.98 for OpenLink and 0.95 for Sindice, with a negligible p-value, indicating very strong agreement between both coherence measures. Henceforth, we use the result-based formula $\mathsf{coh}_r(.)$ as it returns more granular results—e.g., it had 8% fewer scores of precisely 0 than $\mathsf{coh}_q(.)$—and thus results in fewer ties and fewer trivial decisions when ordering patterns.

Using the $\mathsf{coh}_r(.)$ measure, we observed that 67% of the tested predicates in the OpenLink index are entirely up-to-date ($\mathsf{coh}_r(p) = 1$), versus 30% of the predicates for the Sindice endpoint. In contrast, information for 14% of the tested predicates for OpenLink are entirely missing or out-of-date ($\mathsf{coh}_r(p) = 0$), versus 40% for Sindice; these high percentages are due to partial coverage of Web sources, outdated data-dumps in the index, and predicates with dynamic values. Hence we see that these caches are not up-to-date for many predicates, which motivates our current investigation of hybrid-query techniques.

Furthermore, we analysed the correlation for coherence estimates of the same predicates *across* both endpoints. The τ-score was 0.16, again with a negligible p-value. The low correlation highlights the endpoint-specific nature of these measures: the hybrid query approach tackles both the endpoint-independent problem of dynamicity and the endpoint-specific problem of index coverage and updates.

Finally, we looked at the correlation between the selectivity of predicates (i.e., how often they occur) and their coherence, which may have potential consequences for query planning. Specifically, for each endpoint, we compared the number of (live) results generated for each predicate across all queries and their $\mathsf{coh}_r(p)$ value. The τ-value for OpenLink was 0.1, indicating that less selective patterns tend to have slightly lower coherence; the analogous τ-value for Sindice was -0.03, indicating a negligible correlation in the opposite direction. Though limited, we take this as anecdotal evidence to suggest that correlation between

[6] http://challenge.semanticweb.org/

[7] We considered using the SPARQL 1.1 `SAMPLE` keyword, but (i) Virtuoso does not support SPARQL 1.1; (ii) `SAMPLE` does not guarantee randomness of results. A pay-level-domain is the level that one pays to register (e.g., `bbc.co.uk`, `dbpedia.org`).

the selectivity and coherence of predicates is weak, if any. This observation makes the choice between selectivity- and coherence-based ordering even more crucial.

Predicate–domain estimates. So far, we naïvely assume a single coherence value for predicates in all cases, ignoring subject or object URIs: keeping information for each subject/object would have a high overhead. However, the coherence of predicates may vary depending on the site from which the subject/object originated. We can thus generalise subject/object values into pay-level-domains (PLD) and then track coherence for predicate–domain pairs. We mapped the entity URIs of the queries to their PLDs (581 PLDs with a maximum of 74 queries per domain) and resolved the coherence of predicates for individual PLDs. We found that the difference between coherence values across all PLD pairs was ≤ 0.1 for $\sim$40% of the OpenLink and $\sim$15% of the Sindice predicates. All remaining predicates exhibited a higher variance for coherence values across different PLDs.

Maintenance. Assuming the cooperation of the endpoint, various methods can be used to learn about content changes or updates to the centralised index (similar in principle to internal SPARQL caching proposals [19]). Data providers may push change notifications to the endpoints and/or the endpoints can learn about changes by actively monitoring remote sources [15]; this information can then be pushed to the coherence monitor. In a strict black box scenario, where only the public SPARQL interface is available for the cache, one has to periodically rerun or update the gathered statistics, where queries that are observed to return static results could be probed less frequently in an adaptive monitoring setup [6]. In addition, the system could perform a demand-based or "lazy" maintenance of statistics that is based on, e.g., (i) keeping only frequent query patterns up-to-date or (ii) actively updating coherence estimates as hybrid queries are processed.

5 Conclusion

In this paper, we proposed a hybrid query architecture that aims to combine the strengths of centralised cache endpoints, which provide fast results, and the strengths of novel live querying approaches, which provide fresh results from a broad range of Linked Data sources. First, we proposed different techniques by which coherence and selectivity estimates can be combined with reordering and hybrid-split strategies to design an effective hybrid query plan that (potentially) speeds up live results while freshening up cache results. Second, we discussed different coherence estimate formulae and various methods of extracting the necessary information from endpoints to compute the values, e.g., based on probing queries. We conducted an empirical study based on two public endpoints and showed that data are often stale or missing, and that the proposed coherence measures are indeed well-suited to identify this. Furthermore, we showed that the coherence estimates differ for the same predicates across different endpoints and also across different data providers for the same endpoints.

Given the potential scope and dynamicity of Linked Data, query engines will need to employ a wide range of techniques to efficiently offer fresh results with

broad coverage. We believe that our hybrid query proposals make a significant step in this direction by combining both centralised and decentralised query paradigms to exploit both cached and live results. Towards making our proposals a concrete reality, our next steps involve implementing, evaluating and further exploring the combinations of hybrid query techniques presented herein.

Acknowledgements. *This research has been supported by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-II).*

References

1. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 1–15. Springer, Heidelberg (2011)
2. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: FactForge: A fast track to the web of data. In: SWJ (2011)
3. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In: Networked Knowledge – Networked Media. Springer (2009)
4. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
5. Hartig, O., Langegger, A.: A database perspective on consuming Linked Data on the web. In: Datenbank-Spektrum (2010)
6. Käfer, T., Umbrich, J., Hogan, A., Polleres, A.: Towards a Dynamic Linked Data Observatory. In: LDOW at WWW (2012)
7. Karnstedt, M., Sattler, K., Geist, I., Höpfner, H.: Semantic Caching in Ontology-based Mediator Systems. In: Web und Datenbanken, Berliner XML-Tage (2003)
8. Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
9. Martin, M., Unbehauen, J., Auer, S.: Improving the Performance of Semantic Web Applications with SPARQL Query Caching. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 304–318. Springer, Heidelberg (2010)
10. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. IJMSO (2008)
11. Podlipnig, S., Böszörményi, L.: A survey of web cache replacement strategies. ACM Comput. Surv. (2003)
12. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
13. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on 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. 601–616. Springer, Heidelberg (2011)

14. Tran, T., Zhang, L., Studer, R.: Summary Models for Routing Keywords to Linked Data Sources. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 781–797. Springer, Heidelberg (2010)
15. Umbrich, J., Hausenblas, M., Hogan, A., Polleres, A., Decker, S.: Towards dataset dynamics: Change frequency of linked open data sources. In: LDOW (2010)
16. Umbrich, J., Hose, K., Karnstedt, M., Harth, A., Polleres, A.: Comparing data summaries for processing live queries over Linked Data. WWWJ (2011)
17. Umbrich, J., Karnstedt, M., Land, S.: Towards understanding the changing web: Mining the dynamics of linked-data sources and entities. In: KDML (2010)
18. Umbrich, J., Karnstedt, M., Parreira, J.X., Polleres, A., Hauswirth, M.: Linked Data and Live Querying for Enabling Support Platforms for Web Dataspaces. In: DESWEB at ICDE (2012)
19. 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)

Linked-Data Aware URI Schemes for Referencing Text Fragments

Sebastian Hellmann[1], Jens Lehmann[1], and Sören Auer[2]

[1] Universität Leipzig, IFI/BIS/AKSW, D-04109 Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org
[2] Technische Universität Chemnitz, Informatik/ISST, D-09107 Chemnitz, Germany
soeren.auer@informatik.tu-chemnitz.de

Abstract. The NLP Interchange Format (NIF) is an RDF/OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources and annotations. The motivation behind NIF is to allow NLP tools to exchange annotations about text documents in RDF. Hence, the main prerequisite is that parts of the documents (i.e. strings) are referenceable by URIs, so that they can be used as subjects in RDF statements. In this paper, we present two NIF URI schemes for different use cases and evaluate them experimentally by benchmarking the stability of both NIF URI schemes in a Web annotation scenario. Additionally, the schemes are compared with other available schemes used to address text with URIs. The String Ontology, which is the basis for NIF, fixes the referent (i.e. a string in a given text) of the URIs unambiguously for machines and thus enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform.

1 Introduction

We are currently observing a plethora of *Natural Language Processing* (NLP) tools and services being available and new ones appearing almost on a weekly basis. Some examples of web services providing just *Named Entity Recognition* (NER) services are *Zemanta*, *OpenCalais*, *Ontos*, *Evri*, *Extractiv*, *Alchemy*. Similarly, there are tools and services for language detection, Part-Of-Speech (POS) tagging, text classification, morphological analysis, relationship extraction, sentiment analysis and many other NLP tasks. Each of the tools and services has its particular strengths and weaknesses, but exploiting the strengths and synergistically combining different tools is currently an extremely cumbersome and time consuming task. The programming interfaces and result formats of the tools have to be analyzed and differ often to a great extent. Also, once a particular set of tools is integrated this integration is *not reusable* by others.

Additionally, the use of LOD background knowledge in NLP applications poses some particular challenges. These include: *identification* – uniquely identifying and reusing identifiers for (parts of) text, entities, relationships, NLP

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 175–184, 2012.

concepts and annotations etc.; *provenance* – tracking the lineage of text and annotations across tools, domains and applications; *semantic alignment* – tackle the semantic heterogeneity of background knowledge as well as concepts used by different NLP tools and tasks.

In order to simplify the combination of tools, improve their interoperability and facilitate the use of Linked Data we developed the NLP Interchange Format (NIF). NIF is an RDF/OWL-based format that aims to achieve interoperability between *Natural Language Processing* (NLP) tools, language resources and annotations. The NIF specification has been released in an initial version 1.0 in November 2011[1] and implementations for 8 different NLP tools (e.g. UIMA, Gate ANNIE and DBpedia Spotlight) exist and a public web demo[2] is available. NIF addresses the interoperability problem on three layers: the *structural*, *conceptual* and *access* layer. NIF is based on a Linked Data enabled URI scheme for identifying elements in (hyper-)texts that are described by a String Ontology (structural layer) and a selection of ontologies for describing common NLP terms and concepts (conceptual layer). NIF-aware applications will produce output (and possibly also consume input) adhering to the NIF URI Scheme and the String Ontology as REST services (access layer). Other than more centralized solutions such as UIMA[3] and GATE[4], NIF enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. Another benefit is that a NIF wrapper has to be only created once for a particular tool, but enables the tool to interoperate with a potentially large number of other tools without additional adaptations. Ultimately, we envision an ecosystem of NLP tools and services to emerge using NIF for exchanging and integrating rich annotations.

We present the NIF URI Schemes including an experimental evaluation in Section 2. The usage of identifiers in the String Ontology is discussed in Section 3. We review related work in Section 4 and conclude with an outlook on future work in Section 5.

2 NIF URI Schemes

The motivation behind NIF is to allow NLP tools to exchange annotations about documents in RDF. Hence, the main prerequisite is that parts of the documents (i.e. strings) are referenceable by URIs, so that they can be used as subjects in RDF statements. We call an algorithm to create such identifiers *URI Scheme*: For a given text t (a sequence of characters) of length $|t|$ (number of characters), we are looking for a *URI Scheme* to create a URI, that can serve as a *unique* identifier for a substring s of t (i.e. $|s| \leq |t|$). Such a substring can (1) consist of adjacent characters only and it is therefore a unique character sequence within the text, if we account for parameters such as context and position or (2) derived by a function which points to several substrings as defined in (1).

[1] http://nlp2rdf.org/nif-1-0/
[2] http://nlp2rdf.lod2.eu/demo.php
[3] http://uima.apache.org/
[4] http://gate.ac.uk/

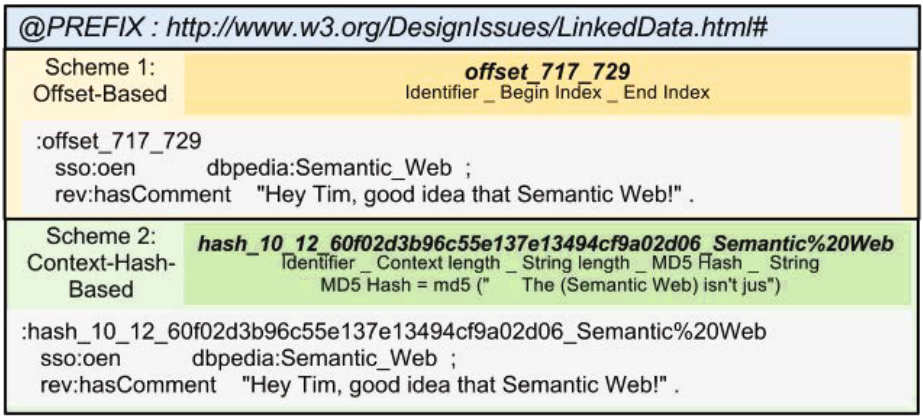


Fig. 1. NIF URI schemes: Offset (top) and context-hashes (bottom) are used to create identifiers for strings, see Section 3 for `sso:oen`

NIF provides two URI schemes, which can be used to represent strings as RDF resources. In this section, we focus on the first scheme using offsets. In the top part of Figure 1, two triples are given that use the following URI as subject: `http://www.w3.org/DesignIssues/LinkedData.html#offset_717_729`

According to the above definition, the URI points to a substring of a given text t, which starts at index 717 until the index 729.

For the URI creation scheme, there are three basic requirements – *uniqueness*, *ease of implementation* and *URI stability* during document changes. Since these three conflicting requirements can not be easily addressed by a single URI creation scheme, NIF defines two URI schemes, which can be chosen depending on which requirement is more important in a certain usage scenario. Naturally further schemes for more specific use cases can be developed easily. After discussing some guidelines on the selection of URI namespaces, we explain in this section how stable URIs can be minted for parts of documents by using *offset-based* and *context-hash* based schemes (see Figure 1 for examples).

Namespace Prefixes. A NIF URI is constructed from a namespace prefix and the actual *identifier* (e.g. "offset_717_729"). Depending on the selected context, different prefixes can be chosen. For practical reasons, it is recommended that the following guidelines should be met for NIF URIs: If we want to annotate a (web) resources, the whole content of the document is considered as `str:Context`, as explained in the next section, and it is straightforward to use the existing document URL as the basis for the prefix. The prefix should then either end with slash ('/') or hash ('#')[5].

[5] Note that with '/' the identifier is sent to the server during a request (e.g. Linked Data), while everything after '#' can only be processed by the client.

Recommended prefixes for `http://www.w3.org/DesignIssues/LinkedData.html` are:
– `http://www.w3.org/DesignIssues/LinkedData.html/`
– `http://www.w3.org/DesignIssues/LinkedData.html#`

Offset-Based URIs. The offset-based URI scheme focuses on ease of implementation and is compatible with the position and range definition of RFC 5147 [6] (esp. Section 2.1.1) and builds upon it in terms of encoding and counting character positions (See Section 4 for a discussion). Offset-based URIs are constructed of three parts separated by an underscore '_': (1) a *scheme identifier*, in this case the string 'offset', (2) *start index*, (3) the *end index*. The indexes are counting the gaps between the characters starting from 0 as specified in RFC 5147 with the exception that the encoding is defined to be Unicode Normal Form C (NFC)[6] and counting is fixed on Unicode Code Units[7]. This scheme is easy and efficient to implement and the addressed string can be referenced unambiguously. Due to its dependency on start and end indexes, however, a substantial disadvantage of offset-based URIs is the *instability* with regard to changes in the document. In case of a document change (i.e. insertion or deletion of characters), all offset-based URIs after the position the change occurred become invalid.

Context-Hash-Based URIs. As an alternative to the offset-based scheme, context-hash-based URIs are designed to remain more robust regarding document changes. Context-hash-based URIs are constructed from five parts separated by an underscore '_':

1. a *scheme identifier*, in this case the string 'hash',
2. the *context length* (number of characters to the left and right used in the message for the hash-digest),
3. the *overall length* of the addressed string,
4. the *message digest*, a 32-character hexadecimal MD5 hash created from the string and the context. The message M consists of a certain number C of characters (see 2. context length above) to the left of the string, a bracket '(', the string itself, another bracket ')' and C characters to the right of the string: 'leftContext(String)rightContext'. If there are not enough characters to left or right, C is adjusted and decreased on the corresponding side (see the 'Hurra!' example below).
5. the *string itself*, the first 20 (or less, if the string is shorter) characters of the addressed string, urlencoded.

The additional brackets '(' and ')' around the string were introduced to make the identifier more uniquely distinguishable. If there is a sentence 'Hurra! Hurra!' and the context size is too large, e.g. 10, then the first and the second 'Hurra!' would have the same hash. By adding brackets, however, the hash is easily distinguishable: `md5("(Hurra! Hurra!)")` != `md5("(Hurra!) Hurra!")` != `md5("Hurra! (Hurra!)")`.

[6] `http://www.unicode.org/reports/tr15/#Norm_Forms`
[7] `http://unicode.org/faq/char_combmark.html#7`

Note that context-hash-based URIs are unique identifiers of a specific string only if the context size is chosen sufficiently large. If, for example, a complete sentence is repeated in the document, parts of the preceding and/or subsequent sentences are to be included to make the reference to the string unique. However, in many NLP applications, a unique reference to a specific string is not necessary, but rather, all word forms within the same minimal context (e.g., one preceding and one following word) are required to be analysed in the same way. Then, a context-hash-based URI refers uniquely to words in the same context, not one specific string. Using a small context, one can refer to a whole class of words rather than just an individual one. For example, by using the string ' the ' (with one preceding and following white space as context) we obtain the digest: md5(' (the) '). The resulting URI is `http://www.w3.org/DesignIssues/LinkedData.html#hash_1_5_8dc0d6c8afa469c52ac4981011b3f582_%20the%20` and would denote all occurrences of 'the' in the given reference context, surrounded by a single white space on both sides.

Trivially, every string is uniquely addressable if the context-length is large enough. The algorithm for finding the addressed strings in a given text is simple: 1. URL decode the fifth part (the string itself) and search for all occurrences and get the start indices. 2. From all found start indices generate the hash by calculating the end index (start index + overall length), adding brackets and including the context (if start index − context length < 0, then left context starts at index 0, right context starts at end index and does not go beyond end of text). The following algorithm computes the minimal context-length (MinCl) on a fixed document with a given set of annotations, so that each URI only denotes one substring.

```
1: procedure MinCl(annotations, cl)
2:     uris ← {}
3:     for all annotations a do
4:         uri ← makeUri(a)
5:         if uris contains uri then
6:             return MinCl (annotations, cl +1 )
7:         else
8:             uris ← uris ∪ uri
9:         end if
10:        return cl
11:    end for
12: end procedure
```

URI Stability Evaluation. As the context-hash-based URI scheme differs significantly in terms of uniqueness and stability from the offset-based scheme, we evaluate both schemes with real revision histories from Wikipedia articles. Although Wikipedia pages are edited quite frequently (≈ 202,000 edits per day[8]), the senses of each page tend to remain relatively stable after a certain number of revisions [2].

[8] `http://www.wikistatistics.net/wiki/en/edits/365`

Table 1. Evaluation of URI stability with different context length versus the offset scheme. The second last column measures how many annotations remain valid over 100 edits on Wikipedia.

tok ≈ 7410.7	Unique	URatio	Stability	1...100	Stab 1...100
context 1	2830.2	0.3988	0.3946	2647.3	0.3680
context 5	7060.0	0.9548	0.9454	6417.7	0.8551
context 10	7311.4	0.9871	0.9771	6548.8	0.8712
context 20	7380.6	0.9963	0.9854	6429.1	0.8553
context 40	7402.2	0.9990	0.9866	6146.8	0.8183
context 80	7408.8	0.9998	0.9847	5678.6	0.7568
offset	7410.7	1.00	0.5425	104.4	0.0164

We downloaded a Wikipedia dump with the full edit revision history[9]. From this dump, we randomly selected 100 articles which had more than 500 edits total. We retrieved the last 100 revisions of these 100 articles and removed the wiki markup[10]. Then we split the resulting plain text into tokens at word level. We used a deterministic algorithm (mostly based on regular expressions) for the markup removal and the tokenisation to avoid any side effects. The text for each revision contained 57062.4 characters on average, which we split into 7410.7 tokens on average (around 7.7 chars/token). About 47.64 characters were added between each revision. For each token and each revision, we generated one URI for the offset scheme and six URIs for the context-based scheme with context length 1, 5, 10, 20, 40 and 80. Cf. Section 4 for details why other approaches were not included in this evaluation. For every same URI that was generated within one revision i for two different tokens (a violation of the uniqueness property), the uniqueness ratio (*URatio*) decreases: $\frac{|UniqueURIs_i|}{|Tokens_i|}$. The stability was calculated by the intersection of UniqueURIs of two revisions (i and $i+1$) over the number of tokens of the second revision: $\frac{|UniqueURIs_i \cap UniqueURIs_{i+1}|}{|Tokens_i+1|}$. Thus non-unique URIs were penalized for the calculation of stability (without this penalty the percentage was always about 99%). We did the same measurement between the first and the last revision (columns *1...100* and *Stab 1...100*) of each article. The results are summarized in Table 1.

While a high context length (cl=80) provides more stability between revisions (99.98%), cl = 10 yields 87.12% of the URIs valid over 100 edits. The offset-based URIs have a probability of 54% to become invalid between revisions. This corresponds roughly to the theoretically probability for a random insertion to break a URI: $\frac{a-1}{n+1} + \frac{n-a+2}{2n+2} = \frac{a+n}{2n+2}$ (n = text length, a = annotation length). For context-hash URIs: $\frac{a+2cl-1}{n+1}$.

3 Usage of Identifiers in the String Ontology

We are able to fix the referent of NIF URIs in the following manner: To avoid ambiguity, NIF requires that the whole string of the document has to be included

[9] http://dumps.wikimedia.org/enwiki/20111007/

[10] Code from http://www.mediawiki.org/wiki/Extension:ActiveAbstract

in the RDF output as an `rdf:Literal` to serve as the reference point, which we will call *inside context* formalized using an OWL class called `str:Context`[11]. By typing NIF URIs as `str:Context` we are refering to the content only, i.e. an arbitrary grouping of characters forming a unit. The term *document* would be inappropriate to capture the real intention of this concept as `str:Context` could also be applied to a *paragraph* or a *sentence* and is **absolutely independent** upon the *wider context* in which the string is actually used such as a Web document reachable via HTTP.

We will distinguish between the notion of outside and inside context of a piece of text. The *inside context* is easy to explain and formalise, as it is the text itself and therefore it provides a *reference context* for each substring contained in the text (i.e. the characters before or after the substring). The *outside context* is more vague and is given by an outside observer, who might arbitrarily interpret the text as a "book chapter" or a "book section".

The class `str:Context` now provides a clear reference point for all other relative URIs used in this context and blocks the addition of information from a larger (outside) context. `str:Context` is therefore disjoint with `foaf:Document`, because labeling a context resource as a document is an information, which is not contained within the context (i.e. the text) itself. It is legal, however, to say that the string of the context occurs in (`str:occursIn`) a `foaf:Document`. Additionally, `str:Context` is a subclass of `str:String` and therefore its instances denote textual strings as well.

```
@prefix : <http://www.w3.org/DesignIssues/LinkedData.html#> .
@prefix str: <http://nlp2rdf.lod2.eu/schema/string/> .
:offset_0_26546  a  str:Context ;
#the exact retrieval method is left underspecified
  str:occursIn <http://www.w3.org/DesignIssues/LinkedData.html> ;
# [...] are all 26547 characters as rdf:Literal
    str:isString    "[...]" .
:offset_717_729  a  str:String ;
    str:referenceContext :offset_0_26546 .
```

As mentioned in Section 2, NIF URIs are grounded on Unicode Characters using Unicode Normalization Form C counted in Code Units. For all resources of type `str:String`, the universe of discourse will then be the **words over the alphabet of Unicode characters** (sometimes called Σ^*). According to the "*RDF Semantics W3C Recommendation*", such an interpretation is considered a "semantic extension"[12] of RDF, because "extra semantic conditions" are "imposed on the meanings of terms"[13]. This "semantic extension" allows – per definitionem – for an unambiguous interpretation of NIF by machines. In particular, the `str:isString` term points to the string that fixes the referent of the context. The meaning of a `str:Context` NIF URI is then exactly the string contained in the object of `str:isString`. Note that Notation 3 even permits literals as subjects of statements, a feature, which might even be adopted to RDF[14].

[11] For the resolution of prefixes, we refer the reader to `http://prefix.cc`

[12] `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#urisandlit`

[13] `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#intro`

[14] `http://lists.w3.org/Archives/Public/www-rdf-comments/2002JanMar/0127.html`

Table 2. Comparison of URI schemes (first two are used in NIF)

	Uniq	Val	XML	Trans	Addr	Self	Impl	Exp	Example
Context-Hash(NIF)	+	+	+	+	+	+	o	o	#hash_10_12_60f0...
Offset(NIF)	++	++	+	- - -	++	+	++	o	#offset_717-729
Offset plain	++	++	-	- - -	++	-	++	o	#717-729
Yee (Context)	+	- -	+	+	- -	- -	- -	o	#:words:The-(Semantic We...
RFC 5147 [6]	++	++	+	- - -	++	++	+	+	#char=717-12
LiveURL (Content)	- -	+	-	+	+	-	++	o	#8Semantic12+0x206A73ED
LiveURL (Position)	+	+	-	- -	+	-	-	o	not available for text
Wilde et al. (Regex)	o	++	+	+	+	+	- -	++	#matching=`Semantic\sWeb`

Conceptual Interoperability via Ontologies. The *Structured Sentence Ontology* (SSO)[15] is built upon the String Ontology and provides additional classes for three basic units: *sentences*, *phrases* and *words*. Conceptual interoperability is ensured in NIF by providing ontologies and vocabularies for representing the actual annotations in RDF. For each NLP domain a pre-existing vocabulary was chosen that serves the most common use cases and facilitates interoperability. Details are described elsewhere: *Part-Of-Speech tags and Syntax* uses the Ontologies of Linguistic Annotation (OLiA, [1]); *Entity Linking* is realized using NERD [4], note that the property `sso:oen` – meaning 'one entity per name' – is explained and formalized there.

4 Related Work

As the suitability of the string identifiers highly depends on the specific task, we present in the following a list of criteria, which allow to evaluate and design suitable identifiers:

Uniqueness. The URIs must uniquely identify the substring. **Validity.** The URI scheme must produce valid URIs for arbitrary substrings. Valid URIs must not contain invalid characters and must be limited in length, since most browsers limit the size of the URIs, they can handle[16]. **XML Compatibility.** The identifier part for the generated URIs should be usable as an XML tag name (for RDF/XML serialisation). For example, XML tag elements can not begin with a number, thus prohibiting tags such as `<717-729>`. **Stability.** The URI should only become invalid if the referenced string is changed significantly, thus rightfully rendering the annotations void. It should not become invalid through unrelated changes. **Addressability.** The URIs can efficiently find the annotated substring within the text, i.e. calculate the start and end index (ideally rule based). **Self-Description.** Some URI schemes require certain parameters to find the appropriate substring in the document. The URIs should contain encoded information that can be used to identify the scheme itself and that can be used to reproduce the *configuration* of the scheme. As correct implementations are necessary to allow the creation of tool chains, it is beneficial, if the

[15] `http://nlp2rdf.lod2.eu/schema/sso/`

[16] MS Internet Explorer has a maximum URL length of 2,083 characters. `http://support.microsoft.com/kb/q208427/`

scheme has a low complexity to avoid **implementation** errors. **Expressivity.** This criteria measure how expressive the function is that references the strings (e.g. regex is more expressive than just start/end index).

Table 2 shows a comparison of various URI schemes. **LiveURLs** [3][17] is realized as a Firefox plugin and offers two different ways to produce string identifiers: a context-based and a position based. The user can select a text in the browser and then the plugin creates the URL pointing to the corresponding fragment. This URL can be shared and the referenced string is highlighted. As the identifier starts with a number, it can create a conflict with XML serialisation. Furthermore, the identifier does not contain enough information to uniquely distinguish duplicates, i.e. it would match several occurrences. The position based method uses a combination of the parent node's id and index in the DOM tree alongside an identifier for the child position. The position based method is content-specific and works only on XHTML. Analogous to all position based methods, the scheme is highly susceptible to change. **Wilde and Dürst** [6] filed an RFC in April 2008[18] proposing a parameter-like syntax using fragments that refer to statistics about the characters in the string (e.g. offsets, line, length), e.g. `ftp://example.com/text.txt#line=10,20;length=9876,UTF-8`. The basic properties of this scheme are a super set to the offset-based NIF scheme and the `owl:sameAs` relation holds: `:offset_717_729 owl:sameAs :char=717,729.` The *line* parameter will be considered for further benchmarks, but lacks the necessary granularity. The spec of the RFC restricts this scheme to the "plain text" media type, which excludes XML and HTML. Furthermore the scheme contains many optional parameters for integrity checking. When used as RDF subjects, it is tedious to resolve such optional parts, as `#line=10,20` is neither syntactically the same URI as `#line=10,20;length=9876`, nor can we automatically infer an `owl:sameAs` relation. **Yee** [7] proposed *Text-Search Fragment Identifiers*, which pinpoint the wanted substring with a fragment that includes the string and its context. Before the creation of the fragment identifier, however, the original HTML source is manipulated and all HTML tags are removed and special characters are normalized. The resulting URL for our example is: `#:words:The-(Semantic Web)-isnt-just-about-putting`. The problem is that the proposed normalization (i.e. remove HTML and tokenise context) can not be standardized easily as it relies on difficult to normalize NLP methods. Therefore, there is no guarantee to reproduce the manipulation bi-directionally (e.g. to find the annotated substring). Longer selected substrings lead to longer, invalid URIs. **Wilde and Baschnagel** [5] propose to use regular expression patterns following the parameter "matching" as fragment identifiers, i.e. `matching=Semantic\sWeb` would match all nine occurrences of "Semantic Web" at once. Although being powerful, it is not straight-forward to implement an algorithm that produces regular expressions addressing the correct strings in a text and thus results in high implementation complexity and unpredictability

[17] `http://liveurls.mozdev.org`
[18] `http://tools.ietf.org/html/rfc5147`

regarding uniqueness. Considering the possibility to include the context in an URI, this scheme is a superset of the previous approach by Yee.

5 Conclusions and Future Work

In this paper, we presented the URI schemes and relevant parts of the String Ontology, which underlie the NLP Interchange Format for integrating NLP applications. NIF addresses weaknesses of centralized integration approaches by defining an ontology-based and linked-data aware text annotation scheme. We argued that the URI schemes used in NIF have advantageous properties when compared with other approaches. This comparison is based on an extensive qualitative comparison as well as an experimental evaluation benchmark, which can be easily reproduced and extended for other scenarios. Especially, the context-hash based URIs look promising to provide a solution for web-scale annotation exchange. Future work comprises the creation of a new version NIF 2.0 with community feedback[19] as well as interoperability with (1) XML-based schemes, e.g. XPointer, (2) Media Fragments and the almost completed Provenance AQ .

Acknowledgments. We would like to thank our colleagues from AKSW research group and the LOD2 project for their helpful comments during the development of NIF. Especially, we would like to thank Christian Chiarcos for his support while using OLiA. This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. Chiarcos, C.: Ontologies of linguistic annotation: Survey and perspectives. In: LREC. European Language Resources Association (2012)
2. Hepp, M., Siorpaes, K., Bachlechner, D.: Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. IEEE Internet Computing 11(5), 54–65 (2007)
3. Kannan, N., Hussain, T.: Live urls: breathing life into urls. In: 15th Int. Conf. on World Wide Web, WWW 2006, pp. 879–880. ACM, New York (2006)
4. Rizzo, G., Troncy, R., Hellmann, S., Bruemmer, M.: NERD meets NIF: Lifting NLP extraction results to the linked data cloud. In: LDOW (2012)
5. Wilde, E., Baschnagel, M.: Fragment identifiers for plain text files. In: ACM HYPERTEXT 2005, pp. 211–213. ACM, New York (2005)
6. Wilde, E., Duerst, M.: URI Fragment Identifiers for the text/plain Media Type (2008), http://tools.ietf.org/html/rfc5147 (Online; accessed April 13, 2011)
7. Yee, K.: Text-Search Fragment Identifiers (1998), http://zesty.ca/crit/draft-yee-url-textsearch-00.txt (Online; accessed April 13, 2011)

[19] http://nlp2rdf.org/get-involved

An Interactive Guidance Process Supporting Consistent Updates of RDFS Graphs

Alice Hermann[1], Sébastien Ferré[2], and Mireille Ducassé[1]

[1] IRISA, INSA Rennes, Campus de Beaulieu, 35042 Rennes cedex, France
[2] IRISA, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes cedex, France

Abstract. With existing tools, when creating a new object in the Semantic Web, users benefit neither from existing objects and their properties, nor from the already known properties of the new object. We propose UTILIS, an interactive process to help users add new objects. While creating a new object, relaxation rules are applied to its current description to find similar objects, whose properties serve as suggestions to expand the description. A user study conducted on a group of master students shows that students, even the ones disconcerted by the unconventional interface, used UTILIS suggestions. In most cases, they could find the searched element in the first three sets of properties of similar objects. Moreover, with UTILIS users did not create any duplicate whereas with the other tool used in the study more than half of them did.

1 Introduction

Updating existing Semantic Web (SW) data is crucial to take into account information regularly discovered. This is, however, tedious and in practice data from the SW are rarely updated by users. In the Web 2.0, users, nevertheless, significantly contribute to the production of data, thus, motivation is not a problem. Models exist to bridge the gap between the SW and the Web 2.0, for example by linking tags created by users with SW vocabulary [13,11]. There are, however, still difficulties to integrate the Web 2.0 data in SW, for example to automatically align users tags and resources of the SW. Moreover, SW data are richer than user tags. SW indeed allows a more complex representation of data, as well as more elaborate queries. It is therefore important that users can directly create data in a SW format.

This paper presents UTILIS (Updating Through Interaction in Logical Information Systems), a method that uses existing objects and the current partial description of a new object, to help the user create that new object. UTILIS searches for objects similar to the object being created, namely objects having properties and values in common with it. These objects and their properties are used as suggestions to extend the description of the new object. In the following, examples and experiments are related to the extension of an annotation base of comics panels. An excerpt of the base is shown in Figure 1. Panel A is taken

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 185–199, 2012.

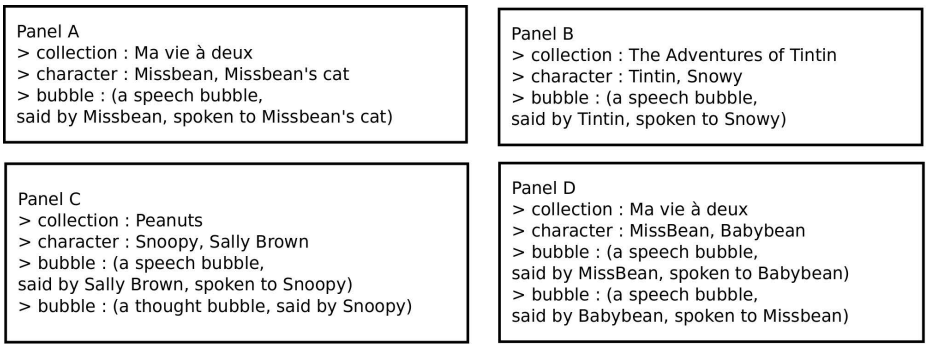


Fig. 1. Excerpt of the comics annotation base

from collection *Ma vie à deux*, with *Missbean* and her cat as characters, it has a speech bubble said by Missbean to her cat.

The main contribution of this paper is an interaction process that helps users constitently create new objects by suggesting properties and values finely adapted to the very object being created. The already known properties of a new object are used to suggest others. Let us assume that a user annotates a new panel and specifies its collection, it may help to suggest characters, it is likely that this panel and those of the same collection have characters in common. The process uses a set of relaxation rules, inspired by the work of Hurtado et al. [9], and an efficient algorithm for computing suggestions. An advantage of our approach is that the definition of an ontology is not necessary to calculate the suggestions, although UTILIS may use an ontology to improve its suggestions when one is available.

A user study conducted with students shows that they have used the suggestions of UTILIS. They found them relevant. In most cases, they could find the searched item in the first three sets of suggestions. In addition, they have appreciated the suggestion mechanism, indeed 14 students out of the 18 wish to have it in a SW data editor. Even if some users were disconcerted, the base resulting from the use of UTILIS was more consistent and contained less errors than when using Protégé.

Section 2 gives definitions related to the SW languages. It introduces logical information systems which are used to interact with users. Section 3 specifies our approach, UTILIS. Section 4 presents the creation of a panel description . Section 5 presents the user study. Section 6 compares our approach to related work.

2 Preliminaries

RDF, RDFS and SPARQL. RDF and RDFS are Semantics Web languages that enable tools to be interoperable. The basic elements of these languages are resources and triples. A resource can be either a URI (absolute name of a resource),

English	Which are the panels with at least one bubble said by Missbean?
SPARQL query	SELECT ?x WHERE {?x a :panel . ?x :bubble ?y . ?y :saidBy <MissBean>)}

Fig. 2. A question and its translation in SPARQL

a literal or an anonymous resource. A triple consists of 3 resources. A triple *(s, p, o)* can be read as a sentence where s is the subject, p is the verb, called predicate, and o is the object.

RDF allows data to be represented. For example, in the annotation base, triple (*<PanelK>, :character, <Missbean>*) can be read as "Panel K has character Missbean". RDF has a predefined vocabulary to represent membership in a resource class (*rdf:type*), the hierarchy between classes (*rdfs:subClassOf*) and between the properties (*rdfs:subPropertyOf*). For example, triple (*:Bubble1, rdf:type, :SpeechBubble)* tells that "Bubble1 has type SpeechBubble" or simply " Bubble1 is a speech bubble". Resource *:SpeechBubble* is a class. Triple (*:SpeechBubble, rdfs:subClassOf, :Bubble)* tells that "the SpeechBubble class is a subclass of Bubble", or "Each speech bubble is a bubble." In addition, RDFS is a language of knowledge representation with inference power [8]. By inference, the latter two triples can be used to deduce the previous triple (*<Bubble1>, rdf:type, :Bubble)*. In the following, for readability reasons, descriptions of the objects are written in Turtle notation [1]. For example, the following triples describing a new panel ((*<PanelK>, rdf:type, Panel*) (*<PanelK>, :character, <Missbean>*)) are written in Turtle as (*<PanelK> a panel; :character <Missbean>*).

SPARQL is a query language for RDF based on *graph pattern matching* [14]. A question and its translation into SPARQL are shown in Figure 2.

Logical Information Systems (LIS). LIS [6] are a paradigm of information retrieval and exploration, combining querying and navigation. They are close to the paradigm of faceted search [15]. Their query language has an expressiveness similar to that of SPARQL, and a syntax similar to Turtle [5]. A prototype, Sewelis[1], has been implemented. The user navigates from query to query. Navigation links are automatically computed from the dataset, and suggested to users, in a way that ensures that guided navigation is safe (no dead-end), and complete (every query that is not a dead-end can be reached). UTILIS is implemented in Sewelis.

3 UTILIS : An Interactive Guidance

This section describes UTILIS, our interactive guidance method to help users create objects in an RDFS graph. UTILIS searches for objects similar to the description of a new object, i.e., objects having common properties and values. Figure 3 illustrates the interactive process to refine the description of a new

[1] http://www.irisa.fr/LIS/softwares/sewelis

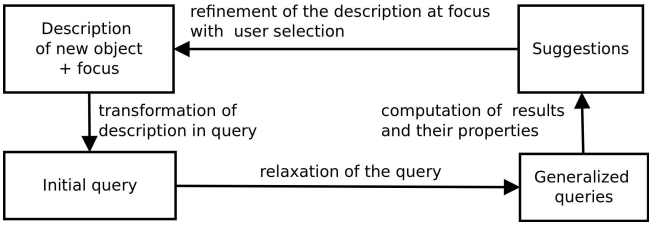


Fig. 3. Interactive refinement of the description of a new object

Description	<PanelK> a :panel; :collection <Ma vie à deux>; :character [ ]
Initial query	SELECT ?z WHERE {?x a :panel . ?x :collection ?y . ?x :character ?z . FILTER (?y = <Ma vie à deux>)}

Fig. 4. A description and its transformation into the initial query

object. The initial query is obtained from the current description of a new object and from a focus which is an element of the description that the user wants to complete. This description is transformed into a query (Section 3.1). That query is used as a starting point to obtain generalized queries using relaxation rules (Section 3.2). The initial and generalized queries allow similar objects to be retrieved. Those objects and their properties are used as suggestions to complete the new description (section 3.3). After refinement, the new description will be used to define a new query and so on. An efficient algorithm to compute the suggestions from the RDFS graph and the current description of the new object has been implemented with dynamic programming techniques (section 3.4).

3.1 Transformation of the Description into the Initial Query

The description consists of the set of elements that have been entered by the user on the new object. Figure 4 shows a description and the initial query obtained after transformation. The description means that panelK is a panel from the *Ma vie à deux* Collection, it has characters which have not yet been specified. The focus is represented by the underlined part. The description is transformed into a query in three steps. Firstly, the identity of the new object is replaced by a variable. For example, <PanelK> is replaced by $?x$. Secondly, the description is transformed into a SPARQL *graph pattern* in order to have only one modifiable component per triple. Each triple is composed of a single individual. The other elements of the triple are variables or predefined property *rdf:type*. For example, (*:collection <Ma vie à deux>*) is transformed into (*?x :collection ?y FILTER (?y = <Ma vie à deux>)*). Finally, the description is transformed into a query by setting the variable at the focus as the variable to look for, and the body of the query matches the graph pattern of step 2. This initial query provides all the individuals that have the already known properties and values of the new object.

Rule	Initial triple or constraint	Relaxed triple	Condition
SuperProperty	$?x\ p_1\ ?y$	$?x\ p_2\ ?y$	p_1 `subp` p_2
SuperClass	$?x$ `a` c_1	$?x$ `a` c_2	c_1 `subc` c_2
Resource	$?x = r_2$	`nil`	
Property	$?x\ p_1\ ?y$	`nil`	$\nexists p \neq p_1.(p_1$ `subp` $p)$
Class	$?x$ `type` c_1	`nil`	$\nexists c \neq c_1.(c_1$ `subc` $c)$

Fig. 5. Relaxation rules. Variables have a leading question mark, r_i are resources, p_j are properties, and c_k are classes; a relaxed triple set to `nil` means that the initial triple is actually removed. `type`, `subc`, `subp` are abbreviations for `rdf:type`, `rdfs:subClassOf` and `rdfs:subPropertyOf`

3.2 Query Relaxation

After adding some property-value pairs, the description of a new object, however, becomes unique in the database, because the database objects do not all have the same property-value pairs. The initial query, obtained from the description of the new object, then leads to no results. In order to continue to offer suggestions to the user, UTILIS seeks objects similar to the new object by generalizing the query. To generalize the initial query, we have defined relaxation rules, inspired by the query approximation rules of Hurtado et al. [9]. Figure 5 shows the set of relaxation rules, that apply to triples. The first column shows the rule name, the second column shows the triple before relaxation, the third column shows the relaxed triple and the fourth column shows potential conditions for the application of the rule. Except the rules *SuperProperty* and *SuperClass*, the rules do not depend on an ontology. Rule *SuperProperty* applies to a triple with a variable as subject, another one as object and a resource p_1 as predicate, under the condition that p_1 is a subproperty of another property p_2. After applying that rule, the subject and object remain the same but property p_1 is replaced by property p_2. For example, (`?x, :spokenTo, ?y`) can be relaxed in (`?x, :inConversationWith, ?y`) by the application of that rule if there exists triple (`:spokenTo, rdfs:subPropertyOf, :inConversationWith`). Relaxed triple `nil` corresponds to the suppression of the initial triple. The distance between the original query and a generalized query is the number of rules applied to switch from one to the other.

At distance 0, are the results of the initial query, namely the ones obtained directly from the description without generalization. For example, let us suppose that the initial query is (`SELECT` $?x$ `WHERE {`$?x$ `a :panel .` $?x$ `:collection` $?y$ `.` $?x$ `:character` $?z$`.` $?x$ `:character` $?a$ `. FILTER (`$?y$ `= <Ma vie à deux> && ?`z `= <MissBean> && ?`a `= <Fatbean>)})`[2]. A generalized query can be generated without the constraint (`?`z `= <Missbean>`) by applying rule *Resource* on this constraint. That generalized query is at distance one. At the same distance, rule *Resource* can also be used on another constraint,

[2] What are the panels of *Ma vie à deux* with characters Missbean and Fatbean ?

for example on constraint (?a = <Fatbean>). The union of all the results of generalized queries produced by a single relaxation step are the proposed results at distance one. Rules can be combined. The order of rule applications has no impact on the generalized queries. This confluence property allows an efficient algorithm using dynamic programming to be implemented (Section 3.4).

3.3 Refinement of the New Object Description

Similar objects are the results of queries, generalized or not. Those objects and their properties are used as suggestions to complete the new description. The suggestions are resources, classes or properties. If the focus is on a resource, the suggestions are classes and properties. A powerful feature of UTILIS is that users can add features to a description at any time and also at any "point" of the description. For example, a user can decide to add an information to the description of any panel already annotated, attaching that information to any existing individual. The place to be extended has to be identified. This is the aim of the focus. In order to avoid that users have to explicitely specify the focus at every step, the default strategy is as follows. When adding a property to the description of the object, the focus is moved to the variable representing the value of the property because it is assumed that the user will most probably want to select or enter a value there after. When the value of the property has been given, the focus returns to the description of the object. When adding a class to the description of the object, the focus remains on the description of the object.

To present suggestions to the user and allow him to complete the description of the new object, the interaction mechanisms of Sewelis, initially dedicated to navigation, have been reused to support creation. By default, only the suggestions at the smallest distance are proposed, the list can be enlarged to bigger distances upon user's request, until all objects compatible with the searched element are proposed. At any time, users can manually enter what they want to add. That is necessary for the new values. Auto-completion mechanisms help find existing values. Once a suggestion is selected by the user, the new description is used to define the initial query that will be generalized to make new suggestions to the user.

3.4 Relaxation Algorithm

A naive algorithm to compute the results at distance d of query q could consist in generating all generalized queries at distance d, and computing the union of their results. The generalized queries at distance d would be obtained by applying d relaxations on the n triples of the query, which amount to $\binom{n}{d}$ generalized queries. Due to the nature of queries whose graph patterns are trees because they are derived from Turtle expressions, the results of each generalized query can be computed in $O(n)$ set operations (intersections and relation crossings). The cost of the naive algorithm is too expensive, especially as the generalized queries are only intermediate steps in our approach.

In UTILIS, we use a more efficient algorithm implemented in dynamic programming. Tabled function $E(d, D)$ directly computes results at distance d of the generalizations of query *SELECT ?x WHERE* $\{$*?x D*$\}$, where D is a Turtle description without subject (list of predicate-object pairs). To make this query correspond to the initial query, it is sufficient to define D as the rewriting of the new object description from the focus. For example, for the initial description *<PanelK> a :panel; :collection []*, D is equal to *is :collection of [a :panel]*. Function E is defined by recursively calling itself with smaller distances and/or sub-descriptions. The base cases are when $d = 0$ and when D is an atomic description (i.e, of the form *a c*, *p r* or *p* []). The algorithm is defined by a set of equations, covering all combinations of a distance and of a description. For example, the equation that defines E for the conjunctions of descriptions is

$$E(d, D_1; D_2) = \bigcup_{i=0}^{d} (E(i, D_1) \cap E(d - i, D_2)).$$

This equation says that a result for a description of the form $D_1; D_2$ is both a result at distance d_1 from D_1 and a result at distance d_2 from D_2, such that $d_1 + d_2 = d$. Distance d is distributed between the two sub-descriptions ($d_1 = i$ and $d_2 = d - i$) in every possible way for $i = 0..d$. The equations for the description of the form *p* $[D_1]$ and *is p of* $[D_1]$ are similar, using the traversal of a relationship instead of an intersection. The results at distance d for classes (resp. properties) are based on the superclasses (resp. superproperties) at distance d in the hierarchy of classes (resp. properties).

The intermediate results of $E(d, D)$ are stored in a table with a line for each distance from 0 to d ($O(d)$ lines), and a column for D and each sub-description of D ($O(n)$ columns). The complexity for computing a cell of this table is $O(d)$ set operations. Therefore, the complexity of our algorithm is $O(nd^2)$ set operations, namely, polynomial instead of combinatorial for the naive algorithm.

4 Example

To illustrate UTILIS, this section describes some details of the creation steps of the description of a new panel in the annotation database. The database contains the panels shown in Figure 1 and six other panels. Let us assume that a user wants to add the panel of Figure 6, named PanelK. It is part of the *Ma vie à deux* collection. It has two characters, Missbean and Fatbean, and a speech bubble said by Missbean to Fatbean.

Figure 7 shows steps 2-7 of the creation. At each step, the top box contains the current description of the object, the focus is underlined, the bottom box contains suggestions with, in front of each of them, the minimum distance between the initial query and the generalized query which led to this suggestion. For space reasons, at each step we show only a limited number of suggestions. The element in boldface corresponds to the choice of the user.

At step 2, the current description is *<PanelK>* *a :panel*. Suggestions adapt to that description: they are the properties of at least one panel. The user chooses

Fig. 6. Panel of *Ma vie à deux : Pour le meilleur et pour le pire !*, by *Missbean, City Editions*

:collection []. The upper part of Figure 8 shows the user interface for step 3, annotations in black have been made by hand, the current description is on the left side. On the right side, suggested resources for the focus are listed, here the collections. Above that area, the number of suggestions is indicated and a *More* button allows them to be expanded to larger distances containing results. At step 3, the user chooses <*Ma vie à deux*> among all the collections. Let us go back to Figure 7, at step 4 the description is <u><*PanelK*></u> *a :panel; :collection* <*Ma vie à deux*>. The suggestions are the properties of at least one panel of *Ma vie à deux*. The property selected by the user is *:character []* to specify characters of the new panel. At step 5, the suggestions are all the characters of the *Ma vie à deux* panels already annotated. The user selects <*Missbean*> and <*Fatbean*>. At step 6, the initial query, corresponding to the description, has no results. Indeed, those two characters do not appear together in any panel of the base yet. By relaxation, UTILIS makes suggestions to the user at distances 1 and 7. The lower part of Figure 8 shows the user interface of step 7, suggested classes and properties are in the middle part. From there on, the user can continue the description of the panel and when he decides that it is complete, he adds it to the base with the *Assert* button.

5 Usability Study

In order to assess the suggestions made by UTILIS during the creation of new objects, an experiment was carried out. Users have tested and evaluated the usability of UTILIS and Protégé [12].[3]

The interface and guidance of Protégé are representative of the current editors. Indeed, two thirds of the Semantic Web users use it as editor [3]. Protégé requires the domain and range of properties to be defined in order to provide any suggestions. When a user chooses the class of an individual, a form with the properties having this class as domain is created. For the values, the suggestions are individuals of the range of each property.

[3] http://protege.stanford.edu

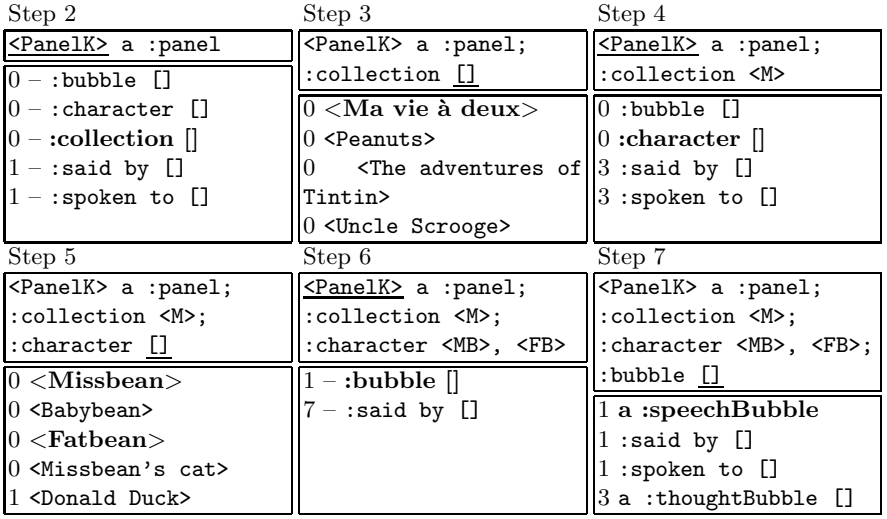


Fig. 7. Creation steps for new panel PanelK : It is related to the *Ma vie à deux* collection. It has characters Missbean and Fatbean. It has a speech bubble said by Missbean spoken to her husband : Fatbean.

5.1 Methodology

The subjects consisted of 18 master students in computer science. They had prior knowledge of relational databases, but knew neither Sewelis, nor Protégé, nor the Semantic Web. For each editor, they had to perform the same tasks. The experiment procedure for each subject was the following: 1) read an introductory note on the overall experiment ; 2) learn how to use an editor with a tutorial of 30 minutes with an example of annotation creation ; 3) develop and create annotation with the editor for 30 minutes ; 4) complete a questionnaire about the editor ; 5) proceed by repeating steps (2) to (4) with the other editor, and 6) complete a comparison questionnaire.

The subjects were asked to update an existing database, describing comics panels. The base was identical at each start of a test session. It consisted of 362 individuals, divided into 16 classes and connected by 20 properties, including 89 panels. Under Sewelis and Protégé, the class and property hierarchies were the same. The subjects were to update the database with two sets of panels. They were divided into four groups, each group conducted two sessions of 1:30. Each group tested one of the two editors on one of the two panel sets (set1 or set2) then it tested the other editor with the other panel set. Each set consisted of 11 panels including the panel used for the tutorial and 10 panels to be annotated, with at least 2 panels from the same collection. The panels were presented on paper in the same order for all subjects. They already existed in the database, but had no description. Subjects were instructed to enter a maximum of information on each panel, taking the description of the existing panels as model. They were

Fig. 8. UTILIS screen shots of steps 3 and 7 of PanelK creation

Table 1. Average number of annotated panels and pourcentage of average number of errors, under Protégé and UTILIS, according to the tools and set orders

	Panels under Protégé	Panels under UTILIS	Errors under Protégé	Errors under UTILIS
groups UTILIS-first	9.6	9	3.5%	4.9%
groups Protégé-first	8.8	8.6	5.4%	6.7%
groups set1-first	8.7	8.6	4.1%	5.2%
groups set2-first	9	8.5	4.9%	6.6%

instructed to reuse existing information as much as possible. Nevertheless, they could create new individuals when needed.

During the experiments, a log file of user actions was created, in particular it recorded the number of descriptions created in Sewelis and Protégé, and for Sewelis, the mode of selection of description items and the number of enlargements of suggestions.

5.2 Results

This section presents the results of the experiment. In particular, the nature of the errors made by users respectively under UTILIS and Protégé is commented. The objective and subjective reactions of users with respect to the suggestions are discussed.

No difference between the groups. As illustrated in Table 1, there are no significant differences in the number of created descriptions (9 in average) between users of the four groups. There are also no significant differences in the average number of errors related to the total number of added triples. The order in which tools were used and the order of the sets had, thus, no influence. As a consequence, in the following the results are globally reported.

Table 2. Percentages of elements of description selected in suggestions, found by completion and created, together with the range of suggestion enlargements and the percentage of elements selected in the first three sets of suggestions

	Properties and classes	"Collections"	"Characters"	"Speakers"	"Recipients"	"Places"	"Other elements"
Selection in suggestions (%)	91	91	90	89	82	58	21
Auto-completion (%)	9	9	8	11	15	25	21
Creation (%)	0	0	2	0	3	17	58
Selection in first 3 sets (%)	91	91	92	89	84	68	44
Range of enlargements	0-4	0-1	0-2	0-1	0-3	0-4	0-6

Suggestions were relevant. UTILIS suggestions have been used by users during the creation of new panels. Table 2 shows the proportion of description elements chosen by selecting a suggestion, by auto-completion and by creation. It also shows the range of suggestion enlargements and the percentage of suggestions selected in the first three sets of suggestions. The three modes of selection have been used. One can see, for example, that adding characters was made in 90% of the cases by selecting a suggestion, in 8% by auto-completion and in 2% by creation. The elements related to the collection, to the characters and to bubble interlocutors were, if they existed, selected more than 80% of the cases in the first 3 sets of suggestions. For locations and other elements, the suggestions were less useful. These descriptive elements are more subjective than the above mentioned ones and they often have been created by users. The questionnaires completed after the experiments give a consistent view. Indeed, 16 subjects found suggestions relevant for characters, 15 for the collection, 12 for interlocutors, but only 6 for the locations and 3 for other elements.

Suggestions were appreciated. Subjects were asked which of the elements of each editor they would like to see in an ideal editor. Ten subjects wished to have the properties in a form as in Protégé. From UTILIS, 14 subjects wished to retain the suggestions adapted to the object being created, and 11 subjects wanted the auto-completion search mode. Nine (resp. 7) subjects wished to have UTILIS suggestions for some of (resp. all) properties.

Consistency was better ensured under UTILIS. Table 3 shows the number of errors per error type introduced in the base by the users, respectively under Protégé and UTILIS. The number of users who made the errors is given between parentheses. The most important result is that 27 duplicates were introduced in the base under Protégé, while none was created under UTILIS. A duplicate is a new individual, created whereas an existing one would have been relevant. Furthermore, under Protégé 50% more wrong values were introduced than under

Table 3. Number of errors (number of users who made them) per error type in the base, under Protégé and UTILIS

	Duplicates	**Wrong values**	Forgotten elements	Cancelled insertions	**Wrong format**	**Wrong properties**	**Misplaced focus**	Total
Protégé	**27 (11)**	**46 (10)**	33 (11)	39 (10)	**0 (0)**	**0 (0)**	**0 (0)**	145 (18)
UTILIS	**0 (0)**	**30 (9)**	33 (3)	30 (9)	**15 (1)**	**48 (3)**	**22 (9)**	178 (16)

UTILIS. It should also be noted that the duplicates have been created by more than half of the users (11 subjects) under Protégé. Moreover, 9 of the 18 users did not introduce any duplicate or value errors at all in the base under UTILIS against 2 only under Protégé. Consistency was thus better ensured with UTILIS. Both the suggestion and auto-completion mechanisms of UTILIS help users to find existing individuals without having to browse through entire lists as in Protégé. As a matter of fact, 13 subjects reported being bothered to have to traverse the entire lists of individuals in Protégé. Furthermore, suggestions and auto-completion provide examples of formats and types of similar cases. It helps users create new individuals that are consistent with existing individuals. Note that the identification and reuse of resources is what distinguishes five stars linked open data (LOD)[4] from four stars LOD, according to the star scheme of LOD.

More handling errors under UTILIS but made by few users. Table 3 shows other types of errors. Cancelled elements are elements which were not committed, or were deleted and not added afterwards. Both the number of cancelled and forgotten elements are similar under both tools. However, under UTILIS the number of users who forgot elements is small (3 users) and significantly lower than under Protégé. Forgotten elements under Protégé are individuals for which several occurrences of the same property were needed, for example several characters in a given panel. There are three types of error made only under UTILIS. Wrong types is a format error, for example the creation of a literal instead of an individual. Examples of wrong properties are using a superproperty or using a property in the wrong direction. A misplaced focus results in an extension attached to a wrong place. These errors cannot occur under Protégé, the forms to be filled are static. It should be noted that all the wrong types have been introduced by a single user and the wrong properties by only 3 users. Regarding the misplaced focus, Section 3.3 presented the default strategy for focus positioning. Users may want to fill in the descriptions in a different order than the default strategy. One objective of this study was to observe how users would cope with focus positioning. Half (9) of the users made focus errors but most (7) of them made that mistake only once. The error occurred at any time, not necessarily

[4] http://www.w3.org/DesignIssues/LinkedData.html

while describing the first panels. Once the error made, seven users thus did not repeat it. In the end, only two users were still abashed by the focus positioning. Altogether, two users managed to make no error at all under UTILIS whereas none did under Protégé. In conclusion, few users were still disconcerted by the unconventional user interaction model of UTILIS at the end of the session. We conjecture that those difficulties came from the extra flexibility of UTILIS. In further work we will study how to help those users manage with the flexibility of UTILIS.

6 Related Work

Editors for Semantic Web data can be dedicated tools (e.g., Protégé [12], OKM (OKM Ontology Management) [4], Gino [2], QuiKey [7]) or integrated into Wiki environments (e.g., SMW (Semantic Media Wiki) [17], KiWI [16], ACEWiki [10]). Their interface can be based on forms (e.g., Protégé, OKM, SMW), on natural language (e.g., Gino, ACEWiki), or only allow for the creation of one triple (or semantic link) at a time (e.g., KiWI, Quikey). We discuss in the following the key differences with UTILIS.

UTILIS does not need any preliminary preparation, nor any schema. Protégé requires the definition of domains and ranges to make any suggestion, because forms are derived from that information. The Semantic Forms[5] of SMW have to be defined manually. Editors based on natural language require the definition of a lexicon, and its relation to an ontology. Like OKM and Quikey, UTILIS can be applied from scratch on any RDFS graph.

UTILIS suggestions are based on individuals rather than on ontology axioms. All above editors, except OKM and Quikey, only look at ontology axioms, i.e. the schema, to compute suggestions, generally starting from the class of the new object. This implies that, at step 5 of our scenario, Protégé lists all characters in alphabetical order, whatever the existing objects in the base. With UTILIS, characters that are already linked to a panel in the base would be suggested first; and if the collection of the panel has already been specified, then characters of that collection are suggested before other characters.

UTILIS suggestions depend on the full description of the new object. The flexible forms of OKM only depend on, and require, the class of the new object, and Quikey simply suggests (through auto-completion) all existing properties and individuals. UTILIS can provide suggestions from the start, when nothing is known about the new object, and refine them whenever an information is added to its description. At step 2 of our scenario, knowing the class improves the suggestion of properties. At step 5, knowing the collection of the panel improves the suggestion of characters. This works for description elements at arbitrary depth, e.g., for suggesting the locutor of a bubble of the panel.

[5] http://www.mediawiki.org/wiki/Extension:Semantic_Forms

UTILIS suggestions are available before any user input. All editors, except Protégé and ACEWiki, require the user to enter a few letters in order to get suggestions of individuals (e.g., characters). This can be explained by the fact that suggestions are not fine-tuned to the new object, and the alternative is then to list all instances of some class, like in Protégé. UTILIS can suggest short lists of individuals as soon as the description becomes specific. For long lists, UTILIS also provides auto-completion, similarly to Quikey.

7 Conclusion

We propose a method, UTILIS, which guides users through the creation of objects in an RDFS graph. The guidance takes benefit of existing objects and of the current description of a new object. At each step, the current description is used to find similar objects, whose properties are then used as suggestions to complete the description. Similar objects are the results of queries that are generalized from the current description. These queries are obtained by relaxation rules. An efficient algorithm to compute suggestions has been designed and implemented in Sewelis.

Compared to other RDFS editors, UTILIS suggestions are based on existing objects, rather than only on the RDF schema. They are, therefore, well adapted to each object being created. One advantage is that UTILIS does not need an ontology to be defined, although it can use one to improve its suggestions when one is available. An experiment has shown that subjects found the suggestions useful and they actually used them. In most cases, they could find the desired item in the first three sets of suggestions. Even if some users were disconcerted, the base resulting from the use of UTILIS was more consistent and contained less errors than when using Protégé. In addition 14 of the 18 subjects wanted to keep UTILIS suggestion mechanism in an editor.

Acknowledgements. We thank Marie Levesque (aka Missbean) to have allowed us to use the panel taken from *Ma vie à deux : Pour le meilleur et pour le pire !* and her publisher *City Editions*. We also thank the 18 students of the INSA institute of technology who participated in the experiment.

References

1. Beckett, D., Berners-Lee, T., Prud'hommeaux, E.: Turtle - Terse RDF Triple Language. W3C Recommendation (January 2010)
2. Bernstein, A., Kaufmann, E.: GINO - A Guided Input Natural Language Ontology Editor. 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. 144–157. Springer, Heidelberg (2006)
3. Cardoso, J.: The semantic web vision: Where are we? IEEE Intelligent Systems, 84–88 (2007)

4. Davies, S., Donaher, C., Hatfield, J.: Making the Semantic Web usable: interface principles to empower the layperson. Journal of Digital Information 12(1) (2010)
5. Ferré, S., Hermann, A.: Semantic Search: Reconciling Expressive Querying and Exploratory Search. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 177–192. Springer, Heidelberg (2011)
6. Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing & Management 40(3), 383–419 (2004)
7. Haller, H.: QuiKey – An Efficient Semantic Command Line. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 473–482. Springer, Heidelberg (2010)
8. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press (2009)
9. Hurtado, C.A., Poulovassilis, A., Wood, P.T.: Query relaxation in RDF. J. Data Semantics 10, 31–61 (2008)
10. Kuhn, T.: How controlled english can improve semantic wikis. In: Semantic Wiki Workshop (SemWiki), vol. 464. CEUR-WS.org (2009)
11. Limpens, F., Gandon, F., Buffa, M.: Sémantique des folksonomies: structuration collaborative et assistée. In: Ingénierie des Connaissances (IC), pp. 37–48. Presses Universitaires de Grenoble (2009)
12. Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., Musen, M.A.: Creating semantic web contents with protege-2000. IEEE Intelligent Systems 16(2), 60–71 (2001)
13. Passant, A., Laublet, P.: Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. In: Workshop Linked Data on the Web (LDOW). CEUR-WS (2008)
14. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (2008)
15. Sacco, G., Tzitzikas, Y. (eds.): Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience. The Information Retrieval Series, vol. 25. Springer, Berlin (2009)
16. Schaffert, S., Eder, J., Grünwald, S., Kurz, T., Radulescu, M., Sint, R., Stroka, S.: Kiwi - a platform for semantic social software. In: Semantic Wiki Workshop (SemWiki). CEUR-WS.org (2009)
17. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic Wikipedia. In: International conference on World Wide Web (WWW), pp. 585–594. ACM Press (2006)

Effective Retrieval Model for Entity with Multi-valued Attributes: BM25MF and Beyond*

Stéphane Campinas, Renaud Delbru, and Giovanni Tummarello

Digital Enterprise Research Institute
National University of Ireland, Galway
Galway, Ireland
firstname.lastname@deri.org

Abstract. The task of entity retrieval becomes increasingly prevalent as more and more structured information about entities is available on the Web in various forms such as documents embedding metadata (RDF, RDFa, Microdata, Microformats). International benchmarking campaigns, e.g., the Text REtrieval Conference or the Semantic Search Challenge, propose entity-oriented search tracks. This reflects the need for an effective search and discovery of entities. In this work, we present a multi-valued attributes model for entity retrieval which extends and generalises existing field-based ranking models. Our model introduces the concept of multi-valued attributes and enables attribute and value-specific normalization and weighting. Based on this model we extend two state-of-the-art field-based rankings, i.e., BM25F and PL2F, and demonstrate based on evaluations over heterogeneous datasets that this model improves significantly the retrieval performance compared to existing models. Finally, we introduce query dependent and independent weights specifically designed for our model which provide significant performance improvement.

Keywords: RDF, Entity Retrieval, Search, Ranking, Semi-Structured Data, BM25, BM25F, BM25MF, PL2, PL2F, PL2MF.

1 Introduction

Despite the fact that the Web is best known as a large collection of textual documents, it also provides an increasing amount of structured data sources in various forms, from HTML tables to Deep Web databases, XML documents, documents embedding semantic markups, e.g., Microformats, Microdata, RDF, RDFa. Structured data on the Web covers a large range of domains, e.g., e-commerce, e-government, social network, scientific, editorial world, ..., and can describe any kind of *entities*, e.g., people, organisations, products, locations, etc.

Until now, search on the Web was mostly concerned with the retrieval of documents (i.e., unstructured text). However, the task of entity retrieval, that is, returning "entities"

* Preliminary results of the approach was presented in a technical report at SemSearch 2011 — http://semsearch.yahoo.com/9-Sindice.pdf. We have extended it with (1) an extension of the PL2F ranking function; (2) a study of optimised normalisation parameters, and (3) a comparison against two other field-based approaches over additional datasets.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 200–215, 2012.

in response to users' information needs, becomes increasingly prevalent as more and more structured information is available on the Web. This calls for systems providing effective means of searching and retrieving structured information. As mentioned in [1], searching over a large collection of structured data is an unsolved problem and is still an area in need of significant research. Entity retrieval has received considerable attention recently from various research communities [2, 3, 4, 5]. According to a recent study, more than half of web queries target a particular entity or instances of a given entity type [6]. Supporting effectively the search and retrieval of entities, therefore, is essential for ensuring a satisfying user experience.

One of the challenges in entity retrieval is to extend existing web search methods by exploiting the rich structure of the data. In this paper, we extend with a model that considers multi-valued attributes existing field-based ranking models, i.e., BM25F [7] and PL2F [8]. We define a multi-valued attribute as an attribute that has more than one value. For example, the email address of a person can be a multi-valued attribute since a person has possibly more than one. Our rationale for this new model is that field-based ranking models do not make a difference between single and multi-valued attributes. Usually, a multi-valued attribute is converted into a single-valued attribute by aggregating all the values into a single bag of words. Such a simplification of the underlying data model is inadequate for structured data and we will demonstrate that it can lead to a less effective ranking. In this paper, we introduce the MF ranking model which tackles specifically this problem, extend two popular ranking frameworks based on this model, and demonstrate its performance in comparison to existing field-based ranking models. Moreover, we introduce and evaluate additional extensions for combining attribute and value-specific weights.

In Section 2, we start by introducing our "Web of Data" model and then explain how we adapt existing field-based ranking models to this model. In Section 3 we review existing approaches for ranking entities in the Web of Data. In Section 4, we introduce the multi-valued attribute model MF and extend two state-of-the-art field-based models, i.e., BM25F and PL2F. Next, we present query dependent and independent weights specifically designed for our model. In Section 5, we compare field-based ranking models against their extension to the MF model on three large and heterogeneous datasets, and evaluate the effectiveness of the introduced weights.

2 Background

In this section, we first introduce a model for the Web of Data and define what is an entity, i.e., the unit of information that is queried and retrieved. Then we explain how to adapt two field-based ranking frameworks, i.e., the Probabilistic Relevance Framework [9] and the Divergence From Randomness [10], to this model.

2.1 The Web of Data

We define the *Web of Data* as the collection of structured data sources that are exposed on the Web through various forms such as HTML tables, Deep Web databases, XML documents, documents embedding semantic markups, e.g., Microformats, Microdata, RDF, RDFa. Since each data source might have its own defined schema, ranging

from loosely to strictly defined, the data structure does not follow strict rules as in a database. Even within a given data source, the schema might not be fixed and may change as the information grows. The information structure evolves over time and new entities can require new attributes. We therefore consider the Web of Data as being *semi-structured* [11].

Web of Data Model. In the rest of this paper, we assume that a common graph-based data model, based on the Resource Description Framework (RDF) model [12], is used for all the semi-structured data sources based on the Web. RDF is a generic data model that can be used for interoperable data exchange. A resource, i.e., an entity, is described in such a way that it can be processed by machines automatically. In RDF, a resource description is composed of statements about the resource. A statement is a triple consisting of a subject, a predicate and an object, and asserts that a subject has a property with some object. A set of RDF statements forms a directed labelled graph. In an RDF graph, as displayed in Figure 1, a node can be of three types: URI, literal and blank node. A URI serves as a globally-unique identifier for a resource. A literal is a character string with an optional associated language and datatype. A blank node represents a resource for which a URI is not given.

Entity Extraction. There are multiple ways to extract entities from an RDF graph. Here, we use the approach described in [13], where we consider an entity as a star graph, i.e., a subgraph with a central node and its direct neighbor nodes it links to. Figure 1 displays how the RDF graph can be split into four entities *me*, *_:b1*, *_:b2* and *paper/547*. Each entity description forms a subgraph containing the incoming and outgoing edges of the entity node. In order to simplify the extraction process, we only consider the outgoing edges of a star graph.

Entity Model. In the remainder of the paper, the unit of information which is retrieved and ranked is an *entity* [13] and is formalised as a list of attribute-value pairs:

Entity represents a set of attribute-value pairs and is identified by the entity node label, e.g., *paper/547*;

Attribute is an edge linking the entity node to one of its neighbor nodes and is identified by the edge label, e.g., *title*, *name* or *creator*;

Value is a neighbor node of the entity node and is identified by the node label, e.g., *Object-* or *paper/547*. A value is always associated to one attribute. Multiple values can be associated to a same attribute, such as the nodes *_:b1* and *_:b2* with the attribute *knows* of the entity *me*.

2.2 Field-Based Ranking Models

In this section, we explain how to adapt two existing field-based ranking frameworks to the entity model. In the Probabilistic Relevance Framework (PRF), BM25F [7] is a popular web-search ranking function for field-based document retrieval where a document is composed of multiple normalized weighted fields. For example, a field can be the title, the author or the body of the document. The Divergence From Randomness (DFR) framework gives birth to many ranking models, in particular PL2F [8] which considers field-based documents similarly to BM25F. The mapping from the field-based document model to the entity model is straightforward. An entity can be seen as a document

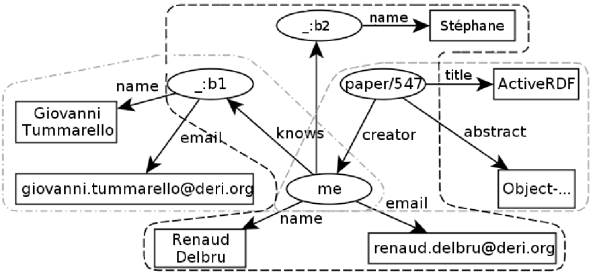


Fig. 1. An RDF graph divided into four entities identified by the nodes *me*, *_:b1*, *_:b2* and *paper/547*

and an entity attribute as a document field. In presence of a multi-valued attribute, the common approach is to merge the content of all the values into one single value, i.e., creating a single bag of words.

Ranking Features. The following features are used in the field-based ranking functions:

attribute length refers to the number of terms in a value node label. In case of a multi-valued attribute, it refers to the number of terms across all the values associated to the attribute.

average attribute length is equal to the mean of *attribute length* across entities.

BM25F Ranking Function. Using BM25F, an entity e is scored with regard to a query q as follows:

$$Score(e,q) = \alpha_e \times \sum_{t \in q} q_t \times tfn \times \omega_t \tag{1}$$

$$tfn = \frac{f_{t,e} \times (k_1 + 1)}{f_{t,e} + k_1} \tag{2}$$

$$f_{t,e} = \sum_{a \in e} \frac{\alpha_a \times f_{t,e,a}}{1 + b_a \times \left(\frac{l_{e,a}}{l_a} - 1\right)} \tag{3}$$

where q_t is the weight of the query q for the term t, i.e., its frequency within the query q, tfn is the term frequency normalization function, ω_t is the Inverse Document Frequency (IDF) function of the term t, k_1 is the saturation parameter, $f_{t,e,a}$ is the frequency of the term t in the attribute a of the entity e, α_a is a weight of the attribute a and α_e a weight of the entity e, b_a is the normalization parameter for the attribute a with $b_a \in [0,1]$, $l_{e,a}$ is the *attribute length* of the attribute a in the entity e, l_a is the *average attribute length* of the attribute a. The IDF is defined as $\omega_t = 1 + log\left(\frac{N}{N_t+1}\right)$, where N is the total number of entities in the collection and N_t is the total number of entities that have occurrences of the term t.

PL2F Ranking Function. DFR weighting models are based on the combination of three components, i.e., the information gain, the randomness model and the term frequency normalization model. PL2F bases the information gain on the Laplace after-effect model, uses Poisson as the model for randomness, and the *normalization 2F* for

the term frequency normalization. Using PL2F, an entity e is scored with regard to a query q as follows:

$$
\begin{aligned}
Score(e,q) &= \alpha_e \times \sum_{t \in q} qtw \times w_{e,t} \\
w_{e,t} &= (1 - P_{risk}) \times (-log_2 (P_P)) \\
P_{risk} &= \frac{tfn}{1 + tfn} \\
P_P &= \frac{\lambda^{tfn}}{tfn!} \times e^{-\lambda} \text{ where } \lambda = \frac{TF}{N}
\end{aligned}
$$

where $qtw = \frac{q_t}{q_{t,max}}$ is the weight of the query q for the term t with $q_{t,max}$ the maximum of q_t in q, $w_{e,t}$ is the weight of the term t in the entity e, $1 - P_{risk}$ estimates the information gain of a term t, $-log_2 (P_P)$ evaluates the importance of a term t in the entity e thanks to the Poisson model and TF is equal to the frequency of the term t in the collection. The factorial is approximated with the Stirling's formula $tfn! = \sqrt{2\pi} \times tfn^{tfn+0.5} \times e^{-tfn}$. The term frequency of the term t in the entity e is normalized as follows:

$$
tfn = \sum_{a \in e} \alpha_a \times f_{t,e,a} \times log_2 \left(1 + c_a \times \frac{l_a}{l_{e,a}}\right) \tag{4}
$$

where c_a is a per-attribute hyperparameter with $c_a \in]0, +\infty[$.

3 Related Work

The Web of Data consists of a wide range of heterogeneous datasets, where the schema and the ontology can vary from one to the other. To overcome this diversity of attributes, different approaches for defining document fields have been proposed. In [14], the authors consider five weighted fields to represent the RDF structure of an entity: literals (textual values), keywords extracted from the entity label, i.e., the subject URI, keywords extracted from the incoming links, entity's types and keywords extracted from object URIs. Compared to the BM25F and PL2F approaches defined in 2.2, this approach is not able to grasp the rich structure of the data since attributes are completely discarded. The BM25F and PL2F approaches we use in our experiments are similar in nature to the BM25F adaptation proposed in [15], where the authors consider one field per attribute in the data collection and can assign a different weight to each attribute. However, they restrict their approach to attributes with literal values, discarding attributes with URI values. In contrast to [15], we consider both attributes with literal and URI values. URIs in the RDF graph carry relevant keywords, with regards to (1) the entity in general when considering the subject URI; (2) the attribute when considering the predicate URI; and (3) the resource the entity relates to when considering the object URI. We also consider in our approach the entity and the attribute labels, i.e., the predicate URIs, using special entity attributes. This aspect is discussed in the Section 5.4.

However, all these approaches are an adaptation of the field-based ranking model in which multiple values associated to a same attribute are aggregated into a single value. This simplification of the underlying data model is inadequate for structured data as we will demonstrate in this paper. Therefore, we propose an extension of field-based ranking models in Section 4 to take into consideration multi-valued attributes and show that our model can be effectively applied to different ranking frameworks such as the PRF and the DFR.

4 MF Ranking Model

In this section we present the multi-valued attribute ranking model, denoted by "MF", which generalizes field-based ranking models to semi-structured data. The MF ranking model leads to a new definition of the term frequency normalization function that aggregates and normalises the term frequencies, first over all the values of an attribute, then over all the attributes of an entity. We introduce next the formal model then present two extensions to the MF model. Finally, we describe weights developed for the MF model.

Multi-Valued Attributes. The MF model integrates multi-valued attributes with an additional intermediate computational step in the ranking function. Although values are related to a same attribute, the relevancy of each value with regard to the query is different. We can assume that, given a same attribute, an entity where two terms occur in a single value is more relevant than another entity where each term occurs in two values. This can be seen as a way to integrate some kind of term proximity measure in the retrieval model, i.e., two words are considered close to each other when they occur in the same value. Reflecting this difference into the importance of a term using an appropriate value-specific weight can improve the ranking efficiency. Therefore, we consider an attribute not as a bag of words but as a *bag of values*, each value being a *bag of words*.

Eliteness. In [16], Harter introduced the notion of *eliteness* in order to model content-bearing terms: a document is said to be *elite* in term t if it is somehow "about" the topic associated with t. In [17], Robertson et al. introduce the relationship between the eliteness of a term in a document and its frequency: an elite term is most likely to be reused in the document, hence the term frequency is used as evidence of the term eliteness in the document. In [7], Zaragoza et al. extend the notion of eliteness to documents with multiple fields. In [18], the authors argue that the normalized frequencies of a term in each field should be combined before applying the term weighting model. Similarly in our MF model, the term eliteness in an entity is shared between its attributes. The values related to a same attribute are associated to a same topic, described by the attribute label. Therefore, a term eliteness in an attribute is shared between its values. As a consequence, for each term, we (1) accumulate the term's evidence of eliteness across an attribute's values; then (2) accumulate its evidence across the attributes; and finally (3) apply the term weighting model on the total term's evidence. The entity score is then derived by combination across the terms.

4.1 MF Ranking Functions

In this section, we describe *BM25MF* and *PL2MF*, the MF extensions of BM25F and PL2F, respectively. We present first the new features needed for the MF ranking model and then define both extensions.

Ranking Features. The MF ranking model requires the following features:
value length refers to the number of terms in a value node label;
attribute length is equal to the *mean* of its *value length*;
average attribute length is the mean of *attribute length* across the entities where that attribute appears;
attribute cardinality is equal to the number of values an attribute possesses;
average attribute cardinality is equal to the mean of the *attribute cardinality* across the entities where that attribute appears.

BM25MF. BM25F is extended by adapting the Equation (3) as follows:

$$f_{t,e} = \sum_{a \in e} \frac{\alpha_a \times f_{t,e,a}}{1 + b_a \times \left(\frac{|a|_e}{|a|} - 1 \right)} \tag{5}$$

$$f_{t,e,a} = \sum_{v \in a} \frac{\alpha_v \times f_{t,e,v}}{1 + b_v \times \left(\frac{l_{e,v}}{l_a} - 1 \right)} \tag{6}$$

where $f_{t,e,v}$ is the frequency of the term t within the value v in the entity e, $l_{e,v}$ is the *value length* of the value v in the entity e, $|a|_e$ is the *attribute cardinality* of the attribute a in the entity e, $|a|$ is the *average attribute cardinality* of the attribute a, α_v and α_a are respectively value and attribute specific weights, b_a and b_v are parameters of the term frequency's normalization, where b_v is value-specific and b_a attribute-specific with $(b_a, b_v) \in [0, 1]^2$.

PL2MF. PL2F is extended by adapting the Equation (4) as follows:

$$tfn = \sum_{a \in e} \alpha_a \times tfn_a \times log_2 \left(1 + c_a \times \frac{|a|}{|a|_e} \right) \tag{7}$$

$$tfn_a = \sum_{v \in a} \alpha_v \times f_{t,e,v} \times log_2 \left(1 + c_v \times \frac{l_a}{l_{e,v}} \right) \tag{8}$$

where c_a and c_v are hyperparameters, with c_a specific to the attribute a and c_v to the value v, with $(c_a, c_v) \in]0, +\infty[^2$.

In Equations (6) and (8), we normalize the term frequency based on the *average attribute length* l_a. In Equations (5) and (7), we further normalize the term frequency based on the *average attribute cardinality* $|a|$. In addition to attribute-specific weights α_a, the MF ranking model allows value-specific weights in its implementations with the parameter α_v. We will present value and attribute specific weights in the next section.

If we assume a single value per attribute to match field-based ranking models, then the Equations (6) and (5) are transformed into the Equation (3), with $\alpha_a \times \alpha_v$ as the

BM25F's attribute weight, and b_v as the attribute normalization parameter. BM25MF is under this condition equivalent to BM25F. Under the same assumption, the Equations (8) and (7) are transformed into the Equation (4), with $\alpha_a \times \alpha_v \times log_2(1 + c_a)$ as the PL2F's attribute weight. PL2MF is under this condition equivalent to PL2F. Therefore, the MF model is a generalisation of field-based models for semi-structured data with multi-valued attributes.

4.2 Weights

In this section, we introduce several weights for the MF model. We first present two query-dependent weights: (1) the *Query Coverage* weight which indicates how well the query terms are covered by an entity, an attribute or a value; and (2) the *Value Coverage* weight which indicates how well a value node is covered by a query. Next, we describe the *Attribute and Entity Labels* query-independent weights.

The Query Coverage Weight. The purpose of the Query Coverage (QC) weight is to lower the importance given to an entity, an attribute or a value with respect to the number of query terms it covers. This weight is combined with the ranking function using α_e, α_a and α_v. For example, given a query composed of three terms, if a value contains only occurrences of one query term, this value will then weight less than a value containing occurrences of more than one query term. It integrates the IDF weight of query terms so that the coverage takes into account the importance of the terms it covers. For example, if two entities have occurrences of one of the three query terms, the coverage would then be $\frac{1}{3}$ for both. Thanks to the IDF weights, the entity with the more important term will have a higher coverage weight than the other one. The QC weight is computed as $\frac{\sum_{t \in X \cap q} \omega_t^2}{\sum_{t \in q} \omega_t^2}$, where X is either a value, an attribute set or the entity and q is the query.

The Value Coverage Weight. The Value Coverage (VC) weight reflects the proportion of terms in a value node matching the query, i.e., how much a query covers a value node. We assume that more the query terms match a large portion of the value node, the more this value node is a precise description of the query. This weight is combined with the ranking function using α_v. VC is defined as the quotient of the query terms frequencies in the value over the *value length*: $c' = \frac{\sum_{t \in v \cap q} f_{t,e,v}}{l_v}$. This definition disadvantages longer values over shorter ones: given a query with two terms, a small value with occurrences of one term would receive a higher weight than a larger value with the two terms occurring, because of the *value length* division. In order to have a better control over the effect of VC, we developed a function which (1) imposes a fixed lower bound to prevent short values receiving a higher weight than long ones; and (2) increases as a power function to ensure a high coverage weight only when c' is close to 1.

$$c_\alpha(c') = \frac{\alpha}{1 + (\alpha - 1) \times c'^B} \tag{9}$$

where $\alpha \in]0, 1[$ is a parameter that sets the lower bound of VC, and B is a parameter that controls the effect of the coverage on the value. The higher B is, the higher the coverage needs to be for the value node to receive a weight higher than α.

The Attribute and Entity Labels Weights. The Attribute and Entity Labels (AEL) weights balance the importance of an entity or an attribute depending on its label. This weight is combined with the ranking function using α_a. The weight value is defined empirically. Comparing the label to a regular expression, the weight is equal (a) to 2 if the label matches ". $\star$ $[label \mid name \mid title \mid sameas]$$"; (b) to 0.5 if the label matches ". $\star$ $[seealso \mid wikilinks]$$"; and (c) to 0.1 if the label matches "`http://www.w3.org/1999/02/22-rdf-syntax-ns#_`$[0-9]+$$"; (d) to 1 otherwise. For instance, if an attribute label is `http://xmlns.com/foaf/0.1/name` then a weight of 2 is assigned. The regular expression (c) matches an attribute URI defining items of a collection in RDF[1]. It is assigned a low weight of 0.1 to reduce the importance of terms occuring in each item of the collection. The entity label is treated as a special attribute of the entity and is assigned a weight of 2.

5 Experiments

In order to evaluate the MF model, we perform several experiments using three different datasets. We start by experimenting on the normalization parameters in order to study their impact on the effectiveness of the approach. We then compare the MF ranking functions against other traditional ones. We finally discuss the consequence of not considering the attribute label as a source of potential relevant terms and demonstrate the effectiveness of the proposed weights.

While there are other ways to perform entity ranking on the Web of Data, e.g., one can look at the other SemSearch 2011 candidates, in this evaluation we concentrate on demonstrating how the MF model specifically extends and improves the very popular PRF and DFR frameworks.

5.1 Datasets

The datasets we are using in our experiments are the following:

INEX09 a dataset of 2,491,134 triples from DBpedia containing the description of entities in English, and converted for the INEX evaluation framework [14];

SS10 the "Billion Triple Challenge"[2] (BTC) dataset, containing more than one billion triples, with the assessments of the SemSearch2010[3] challenge;

SS11 the BTC dataset with the assessments of the "Entity Search track" of the SemSearch2011[4] challenge.

The INEX09 dataset is significantly different than the other two based on BTC. Indeed, BTC is a heterogeneous dataset, created from web crawls of several search engines. INEX09 is a highly curated dataset from DBpedia.

[1] RDF Container: `http://www.w3.org/TR/rdf-schema/#ch_container`
[2] Billion Triple Challenge: `http://vmlion25.deri.ie`
[3] SemSearch2010: `http://km.aifb.kit.edu/ws/semsearch10`
[4] SemSearch2011: `http://km.aifb.kit.edu/ws/semsearch11`

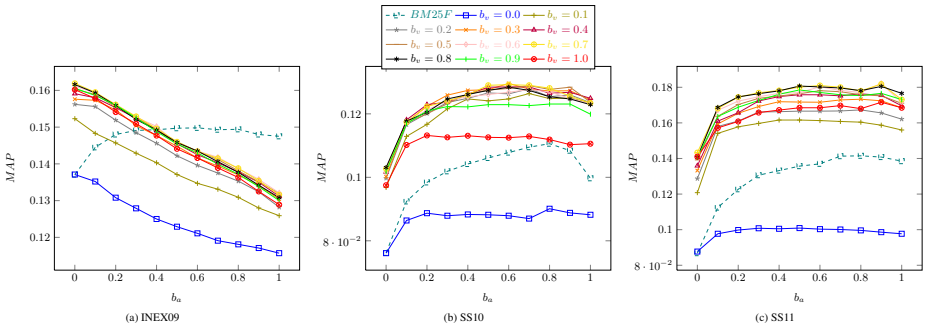


Fig. 2. Experiment with the BM25MF normalization parameters. The figures report the MAP values of the respective datasets, where a curve plots a fixed b_v value with b_a varying from 0 to 1 with a precision step of 0.1.

5.2 Effectiveness of the MF Ranking Model

In this section, we study the impact on the retrieval performance of the normalization parameters. Next we perform a comparison between the MF extensions, i.e., BM25MF and PL2MF, against other ranking approaches.

The Normalization Parameters. The effectiveness of the methods from the PRF and DFR frameworks depends on finding the right values for the normalization parameters. However, these parameters are highly dependent on the dataset. In addition to the length normalization of field-based ranking function, the MF ranking function offers an additional normalization on the attribute's cardinality. The Figures 2 and 3 depict the impact of the normalization parameters on the retrieval performance of BM25MF and PL2MF respectively. Each figure depicts the Mean Average Precision (MAP) scores on the three datasets for BM25MF (resp., PL2MF), with the value normalization parameter b_v (resp., c_v) on the x axis and the MAP score on the y axis. Each curve plots the results with a fixed attribute's normalization parameter b_a (resp., c_a). The grid of parameters values in Figure 2 ranges from 0 to 1 with a step of 0.1. In Figure 3, the grid ranges from 0.5 to 10.5 with a step of 1. Although these parameters can be attribute and value-specific, this experiment considers a constant parameter in order to reduce the number of combinations and to lower the variability of the results. Dashed lines depict the MAP scores of BM25F and PL2F and solid lines the scores of their MF extension, BM25MF and PL2MF respectively. These plots show that using a normalization on the value node provides improved performance. Indeed, the attribute normalization parameters b_a and c_a alone do not grasp the heterogeneity in the data,which results in lower performance when compared to the MF extensions. This indicates that the distinction of an attribute being a set of values has a positive effect on the ranking.

5.3 Comparison between MF and Field-Based Models

In this section, we evaluate and compare the performance of BM25 and PL2 ranking model against their MF extensions, BM25MF and PL2F respectively, and show the superiority of the MF model. TF-IDF is used as baseline.

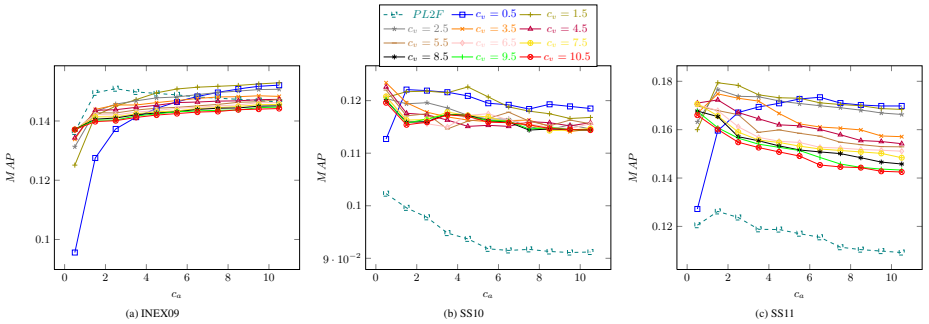


Fig. 3. Experiment with the PL2MF normalization parameters. The figures report the MAP values of the respective datasets, where a curve plots a fixed c_v value with c_a varying from 0.5 to 10.5 with a precision step of 1.

TF-IDF is a logarithmic function of the term frequency and defines the Equation (2) as $tfn = log(f_{t,e}) + 1$, where $f_{t,e}$ is the number of occurrences of the term t in the entity e.

BM25 [19] considers the document as a simple bag of words. It is a function of the term frequency derived from a two-Poisson model and using an entity-length normalization. The entity length is computed as the sum of the *attribute length* defined in the Section 2.2. It defines the Equation (2) as $tfn = \frac{f_{t,e} \times (k_1+1)}{f_{t,e} + k_1 \times \left(1 + b \times \left(\frac{l_e}{l_{avg}} - 1\right)\right)}$, where l_e is the *entity length* of the entity e, l_{avg} is the average of the *entity length* in the collection and b is a normalization parameter.

BM25F is defined in Equation (2). It considers documents as composed of fields, each field being a bag of words.

PL2 [10] considers the document as a simple bag of words. It is a model derived from the DFR framework, with the Equation (4) formulated as $tfn = f_{t,e} \times log_2\left(1 + c \times \frac{l_{avg}}{l_e}\right)$, where c is a normalization parameter.

PL2F is defined in Equation (4). It considers a document as a set of fields, each field being a bag of words.

Comparison. The Table 1a reports the values of the normalization parameters of each ranking function found through a constrained particle swarm optimization [20] on each dataset. Using such parameters, we report in Table 1b the performance of the ranking functions on the three datasets. The p-Value is computed with the two-tailed Wilcoxon matched-pairs signed-ranks test [21, 22], where a statistically significant difference at level 0.10 is marked with one star $*$ and at level 0.05 with two stars $**$. BM25MF and PL2MF are used as a baseline in this test. $\Delta\%$ indicates the difference in percentage between the two MAP values compared in that test. We note that for field-based ranking models and their MF extensions, the attribute label is considered as a value node, in order to be a source of potential relevant terms.

TF-IDF provides a clear-cut discrepancy between INEX09 and the datasets based on BTC, i.e., SS10 and SS11, the reason being it is not suited to heterogeneous datasets. On SS10, BM25MF (resp., PL2MF) does not report a significant difference with BM25

(resp., PL2). On INEX09 and SS11, the MF extensions provide an increase of at least 10% in retrieval performance compared to BM25 and PL2. On SS10 and SS11, the MF extensions provide better retrieval performance with a significant difference than the field-based ranking functions with an increase of 15% at the minimum. On INEX09, BM25MF provides slightly better results than BM25F. Overall, the experiments show that the MF model improves significantly the ranking effectiveness.

Table 1. Comparison of state-of-the-art candidates against the MF generalizations

(a) Normalization parameters values, found through a constrained particle swarm optimization.

	INEX09		SS10		SS11	
BM25MF	$b_a = 0.00$	$b_v = 0.75$	$b_a = 0.58$	$b_v = 0.75$	$b_a = 0.58$	$b_v = 0.75$
BM25	$b = 0.20$		$b = 0.20$		$b = 0.20$	
BM25F	$b_a = 0.82$		$b_a = 0.82$		$b_a = 0.82$	
PL2MF	$c_a = 9.19$	$c_v = 0.76$	$c_a = 1.52$	$c_v = 1.03$	$c_a = 1.79$	$c_v = 1.88$
PL2	$c = 17.01$		$c = 10.09$		$c = 10.09$	
PL2F	$c_a = 1.87$		$c_a = 0.51$		$c_a = 1.51$	

(b) Mean Average Precision (MAP) and the Precision at 10 (P@10) scores of PL2MF and BM25MF and the other state-of-the-art candidates; a p-Value is computed using the two-tailed Wilcoxon matched-pairs signed-ranks test, where one star $*$ marks statistically significant difference at level 0.10, and two stars $**$ at level 0.05, with BM25MF (resp., PL2MF) used as a baseline; $\Delta\%$ indicates the difference in percentage between the two MAP values compared in that test.

	INEX09				SS10				SS11			
	MAP	P@10	p-Value	$\Delta\%$	MAP	P@10	p-Value	$\Delta\%$	MAP	P@10	p-Value	$\Delta\%$
BM25MF	0.1593	0.3982	-	-	0.1303	0.3783	-	-	0.1811	0.1880	-	-
TF-IDF	0.1246	0.3109	$3.2e^{-04**}$	-21.78	0.0581	0.2304	$2.1e^{-10**}$	-55.41	0.0655	0.1040	$1.4e^{-06**}$	-176.49
BM25	0.1330	0.3309	$8.4e^{-05**}$	-16.51	0.1350	0.4000	$3.7e^{-01}$	-	0.1625	0.1920	$1.4e^{-01*}$	-11.45
BM25F	0.1489	0.3764	$5.7e^{-03**}$	-6.53	0.1100	0.3283	$3.0e^{-05**}$	-15.58	0.1401	0.1680	$1.1e^{-04**}$	-29.26
PL2MF	0.1525	0.3800	-	-	0.1232	0.3707	-	-	0.1797	0.1880	-	-
TF-IDF	0.1246	0.3109	$5.4e^{-03**}$	-18.30	0.0581	0.2304	$5.4e^{-10**}$	-52.84	0.0655	0.1040	$6.9e^{-07**}$	-174.35
PL2	0.1331	0.3218	$6.0e^{-03**}$	-12.72	0.1289	0.3946	$4.1e^{-01}$	-	0.1614	0.2000	$3.3e^{-01*}$	-11.34
PL2F	0.1514	0.3473	$7.7e^{-01}$	-	0.1023	0.3163	$2.4e^{-04**}$	-16.96	0.1264	0.1560	$4.5e^{-05**}$	-42.17

5.4 Effectiveness of the Weights

In this section, we discuss the impact of discarding the attribute label as a source of possible relevant terms on the ranking performance. Then we evaluate the weights from Section 4.2 developed for the MF model. The Table 2 reports the MAP scores of BM25MF and PL2MF combined with each weight individually and using the normalization parameters values from the Table 1a. Apart from the row "Without Attribute Label", all runs consider the attribute label as an additional value as in the previous experiments.

The Impact of Attribute Label. In this section, we investigate the consequence of not considering the attribute label as a source of relevant terms. The Table 2 reports under the *BM25MF* and *PL2MF* methods the results of considering or not the attribute label as an additional value. We can see that removing the attribute label (*Without Attribute Label* row) lowers the performance of the ranking with a statistical significance on INEX09 with BM25MF and PL2MF, and on SS11 with PL2MF only. This shows that the attribute labels can be a source of possible relevant terms.

Table 2. Evaluation of the weights effectiveness on PL2MF and BM25MF

Method	INEX09 MAP	INEX09 P@10	INEX09 *p*-Value	INEX09 Δ%	SS10 MAP	SS10 P@10	SS10 *p*-Value	SS10 Δ%	SS11 MAP	SS11 P@10	SS11 *p*-Value	SS11 Δ%
BM25MF												
With Attribute Label	0.1593	0.3982	-	-	0.1303	0.3783	-	-	0.1811	0.1880	-	-
Without Attribute Label	0.1484	0.3800	$6.9e^{-04**}$	-6.84	0.1241	0.3783	$4.5e-01$	-	0.1763	0.1940	$8.6e-01$	-
BM25MF + QC												
Value	0.1482	0.3545	$2.0e^{-01*}$	-6.97	0.1325	0.3793	$8.2e^{-01}$	-	0.1841	0.2020	$1.6e^{-01*}$	$+1.66$
Attribute	0.1624	0.3818	$8.8e^{-01}$	-	0.1339	0.3815	$3.5e^{-01*}$	$+2.76$	0.1841	0.2060	$5.3e^{-02*}$	$+1.66$
Entity	0.1514	0.3782	$4.0e^{-02*}$	-4.96	0.1236	0.3728	$2.8e^{-02**}$	-5.14	0.1744	0.188	$4.7e^{-01}$	-
All	0.1506	0.3545	$1.5e^{-01*}$	-5.46	0.1263	0.3717	$3.1e^{-01*}$	-3.07	0.1810	0.2080	$7.1e^{-01}$	-
BM25MF + VC												
With Function (9)	0.1606	0.3964	$2.8e^{-01*}$ $B=1,\ \alpha=0.7$	$+0.82$	0.1321	0.3761	$3.7e^{-01}$ $B=2,\ \alpha=0.4$	-	0.1802	0.1920	$5.4e^{-01}$ $B=1,\ \alpha=0.9$	-
Without Function (9)	0.1293	0.3055	$8.9e^{-04**}$	-18.83	0.1260	0.3609	$5.3e^{-01}$	-	0.1296	0.1820	$2.1e^{-03**}$	-39.74
BM25MF + AEL												
Entity Label Weight	0.1574	0.3909	$9.1e^{-02*}$	-1.19	0.1401	0.4011	$7.8e^{-05**}$	$+7.52$	0.1937	0.2000	$6.1e^{-03**}$	$+6.96$
Attribute Label Weight	0.1604	0.3982	$4.3e^{-01}$	-	0.1504	0.4228	$2.6e^{-06**}$	$+15.43$	0.2173	0.2360	$6.8e^{-06**}$	$+19.99$
Both	0.1593	0.3982	$4.4e^{-01}$	-	0.1584	0.4391	$2.0e^{-07**}$	$+21.57$	0.2274	0.2420	$2.7e^{-05**}$	$+25.57$
BM25MF + AC + VC + AEL	0.1589		$5.9e^{-01}$	-	0.1720	0.4620	$3.8e^{-06**}$	$+32.00$	0.2416	0.2560	$1.1e^{-05**}$	$+33.41$
PL2MF												
With Attribute Label	0.1525	0.3800	-	-	0.1232	0.3685	-	-	0.1797	0.1880	-	-
Without Attribute Label	0.1401	0.3618	$2.3e^{-04**}$	-8.13	0.1192	0.3674	$5.3e^{-01}$	-	0.1680	0.1840	$1.6e^{-01*}$	-6.51
PL2MF + QC												
Value	0.1348	0.3382	$1.0e^{-03**}$	-11.61	0.1276	0.3750	$2.8e^{-01*}$	$+3.57$	0.1818	0.1980	$1.6e^{-01*}$	$+1.17$
Attribute	0.1569	0.3793	$8.4e^{-01}$	-	0.1299	0.3761	$1.6e^{-01*}$	$+5.44$	0.1815	0.1960	$6.5e^{-02*}$	$+1.00$
Entity	0.1499	0.3727	$3.9e^{-01}$	-	0.1168	0.3609	$3.8e^{-02**}$	-5.19	0.1655	0.1900	$1.2e^{-01*}$	-7.90
All	0.1374	0.3309	$1.5e^{-02**}$	-9.90	0.1257	0.3728	$6.4e^{-01}$	-	0.1743	0.2020	$9.4e^{-01}$	-3.01
PL2MF + VC												
With Function (9)	0.1494	0.3673	$1.5e^{-02**}$ $B=1,\ \alpha=0.7$	-2.03	0.1253	0.3663	$2.4e^{-01*}$ $B=2,\ \alpha=0.4$	$+1.70$	0.1802	0.1900	$1.4e^{-01*}$ $B=1,\ \alpha=0.9$	$+0.29$
Without Function (9)	0.1182	0.2909	$5.4e^{-07**}$	-21.84	0.1179	0.3391	$3.8e^{-01}$	-	0.1466	0.1900	$7.3e^{-03**}$	-18.42
PL2MF + AEL												
Entity Label Weight	0.1523	0.3800	$2.5e^{-01*}$	-0.13	0.1305	0.3848	$6.4e^{-06**}$	$+5.93$	0.1824	0.1920	$4.4e^{-02*}$	$+1.50$
Attribute Label Weight	0.1521	0.3673	$1.9e^{-02**}$	-0.26	0.1471	0.4163	$3.7e^{-07**}$	$+19.40$	0.2150	0.2320	$4.0e^{-05**}$	$+19.64$
Both	0.1516	0.3709	$1.2e^{-02**}$	-0.59	0.1542	0.4337	$3.4e^{-09**}$	$+25.16$	0.2187	0.2380	$1.2e^{-05**}$	$+21.70$
PL2MF + QC + VC + AEL	0.1492	0.3582	$1.5e^{-01*}$	-2.16	0.1717	0.4620	$2.0e^{-08**}$	$+39.36$	0.2360	0.2520	$2.5e^{-05**}$	$+31.33$

The Query Coverage Weight. In order to evaluate the benefit of the QC weight, we first analyse its effect separately when applied as an entity, an attribute or a value-specific weight. Then we study the consequence of applying it on all nodes at the same time (*All* row). The Table 2 reports the results under the *BM25MF + QC* and *PL2MF + QC* methods. QC improves the retrieval performance when applied on the attribute node, with a statistical significance on SS10 and SS11.

The Value Coverage Weight. The evaluation of the VC weight investigates its efficiency with and without the function (9). The results are reported in Table 2 under the *BM25MF + VC* and *PL2MF + VC* methods. We provide for each dataset the best performing B and α parameters. We can observe that VC with the function (9) improves slightly the retrieval performance on SS10 and SS11. The reason is that, without this function, VC assigns a low weight to long values even if they have occurrences of all query terms.

The Attribute and Entity Labels Weights. We evaluate the AEL weights first by considering the Attribute and the Entity Label weights separately, then both at the same time. The Table 2 reports the results of applying such query-independent weights under the *BM25MF + AEL* and *PL2MF + AEL* methods. We note that the Attribute Label weight gives significant benefits to the ranking in SS10 and SS11, while it decreases

the ranking performance in INEX09. This indicates that carefully defined weights for important and non-important attributes can contribute significantly to the effectiveness of the approach. We note also that the same can be seen with the Entity Label weight applied alone. The reason is similar to the Attribute Label weight. Except in INEX09, using both weights at the same time increases the performance of MF ranking functions noticeably.

The Combination of Weights. In this section, we investigate the retrieval performance when all four weights are used together. We report the results in Table 2 under the methods *BM25MF + AC + VC + AEL* and *PL2MF + AC + VC + AEL*, with the weights configuration (1) QC applied on the attribute node; (2) VC with dataset-specific B and α parameters; and (3) AEL weights. The weights applied on a same node are combined by the multiplication of each weight value on that node, i.e., either b_a (resp., c_a) or b_v (resp., c_v) weight values. The attribute label being considered as a value, and the entity label being an additional attribute, we apply also the QC weight on those two labels in this experiment. On INEX09 their combination decreases slightly the performance for PL2MF. On SS10 and SS11, although the QC and VC weights applied separately do not improve the effectiveness of the MF ranking functions by much, their combination with AEL increases the retrieval performance by at least 30% on SS10 and SS11.

6 Conclusion

In this paper, we introduced the MF model for ad-hoc retrieval of semi-structured data with multi-valued attributes. We have discussed how this model extends and generalises existing field-based models. Based on our MF model, we have developed two extensions BM25MF and PL2MF of two popular field-based ranking models, respectively BM25F and PL2F. We have shown throughout evaluations on three large and heterogeneous datasets that the MF model provides significant performance improvement over the field-based ranking models. In addition, the proposed approach provides additional extensions for combining attribute and value-specific weights. We have presented query dependent and independent weights specifically designed for our model and have demonstrated that such weights improve the overall performance.

We have shown in the evaluations that normalization parameters in the MF model are highly dependent on the datasets. A dynamic approach for finding such parameters would improve the performance of the approach and will be investigated in a future work. Also, in order to improve the task of entity search, we could further extend the model to consider not only the entity but also other neighbor entities. This would mean extending the MF model to consider larger graphs than the current entity's star-graph. Furthermore, we will investigate methodologies for asserting the importance of an attribute automatically.

Acknowledgement. This material is based upon works supported by the European FP7 project LOD2 (257943).

References

[1] Cafarella, M.J., Halevy, A., Madhavan, J.: Structured Data on the Web. Communications of the ACM 54(2), 72 (2011)

[2] Balog, K., Serdyukov, P., de Vries, A.P.: Overview of the TREC 2010 Entity Track. In: Proceedings of the Nineteenth Text REtrieval Conference (TREC 2010), NIST (2011)

[3] Demartini, G., Iofciu, T., de Vries, A.P.: Overview of the INEX 2009 Entity Ranking Track. In: Geva, S., Kamps, J., Trotman, A. (eds.) INEX 2009. LNCS, vol. 6203, pp. 254–264. Springer, Heidelberg (2010)

[4] Tran, T., Mika, P., Wang, H., Grobelnik, M.: Semsearch'11: the 4th semantic search workshop. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India (Companion Volume), March 28-April 1, pp. 315–316. ACM (2011)

[5] Blanco, R., Halpin, H., Herzig, D.M., Mika, P., Pound, J., Thompson, H.S., Tran, D.T.: Entity search evaluation over structured web data. In: Proceedings of the 1st International Workshop on Entity-Oriented Search at SIGIR 2011, Beijing, PR China (Juli 2011)

[6] Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Proceedings of the 19th International Conference on World Wide Web, pp. 771–780. ACM Press, New York (2010)

[7] Zaragoza, H., Craswell, N., Taylor, M.J., Saria, S., Robertson, S.E.: Microsoft Cambridge at TREC 13: Web and Hard Tracks. In: TREC 2004, p. 1–1 (2004)

[8] Macdonald, C., Plachouras, V., He, B., Lioma, C., Ounis, I.: University of Glasgow at WebCLEF 2005: Experiments in Per-Field Normalisation and Language Specific Stemming. In: Peters, C., Gey, F.C., Gonzalo, J., Müller, H., Jones, G.J.F., Kluck, M., Magnini, B., de Rijke, M., Giampiccolo, D. (eds.) CLEF 2005. LNCS, vol. 4022, pp. 898–907. Springer, Heidelberg (2006)

[9] Robertson, S., Zaragoza, H.: The Probabilistic Relevance Framework: BM25 and Beyond. Found. Trends Inf. Retr. 3, 333–389 (2009)

[10] Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. 20(4), 357–389 (2002)

[11] Abiteboul, S.: Querying Semi-Structured Data. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 1–18. Springer, Heidelberg (1996)

[12] Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. Changes 10, 1–20 (2004)

[13] Delbru, R., Campinas, S., Tummarello, G.: Searching Web Data: an Entity Retrieval and High-Performance Indexing Model. Web Semantics: Science, Services and Agents on the World Wide Web 10(0) (2012)

[14] Pérez-Agüera, J.R., Arroyo, J., Greenberg, J., Iglesias, J.P., Fresno, V.: Using BM25F for semantic search. In: Proceedings of the 3rd International Semantic Search Workshop, SEMSEARCH 2010, pp. 2:1–2:8. ACM, New York (2010)

[15] Blanco, R., Mika, P., Vigna, S.: Effective and Efficient Entity Search in RDF 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. 83–97. Springer, Heidelberg (2011)

[16] Harter, S.: A probabilistic approach to automatic keyword indexing. PhD thesis, The University of Chicago (1974)

[17] Robertson, S.E., van Rijsbergen, C.J., Porter, M.F.: Probabilistic models of indexing and searching. In: Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval, pp. 35–56. Butterworth & Co, Kent (1981)

[18] Robertson, S., Zaragoza, H., Taylor, M.: Simple BM25 extension to multiple weighted fields. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM 2004, pp. 42–49. ACM, New York (2004)
[19] Robertson, S.E., Walker, S.: Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1994, pp. 232–241. Springer-Verlag New York, Inc., New York (1994)
[20] Hu, X., Eberhart, R.: Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In: 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), pp. 203–206 (2002)
[21] Sheskin, D.J., Hall, C.: Handbook of Parametric and Nonparametric Statistical Procedures, 3rd edn. CRC (2003)
[22] Büttcher, S., Clarke, C., Cormack, G.V.: Information Retrieval: Implementing and Evaluating Search Engines. The MIT Press (2010)

Ontology Testing - Methodology and Tool

Eva Blomqvist[1,2], Azam Seil Sepour[3], and Valentina Presutti[2]

[1] Linköping University, Sweden
eva.blomqvist@liu.se
[2] Semantic Technologies Lab, ISTC-CNR, Italy
firstname.lastname@istc.cnr.it
[3] Jönköping University, Sweden
baran.gems@gmail.com

Abstract. Ontology engineering is lacking methods for verifying that ontological requirements are actually fulfilled by an ontology. There is a need for practical and detailed methodologies and tools for carrying out testing procedures and storing data about a test case and its execution. In this paper we first describe a methodology for conducting ontology testing, as well as three examples of this methodology for testing specific types of requirements. Next, we describe a tool that practically supports the methodology. We conclude that there is a need to support users in this crucial part of ontology engineering, and that our proposed methodology is a step in this direction.

Keywords: Ontology engineering, ontology evaluation, testing.

1 Introduction

A number of ontology engineering methodologies are available, both for general formal ontologies and application specific models, e.g. for Semantic Web applications; we focus on the latter. Although not always made explicit, some methodologies target specific aspects of ontologies, e.g. terminological coverage or general abstract notions, while others focus on application ontologies for performing a specific task within some software, e.g. on the Semantic Web. Common ways of expressing ontological requirements include Competency Questions (CQs) [8] and descriptions of reasoning tasks, however, very few approaches focus on how to verify that such requirements are fulfilled. When using CQs as requirements for different types of ontologies the nature of the questions differ, where for application ontologies, requirements are detailed and precise, e.g. representing actual queries. Similarly, if inferences are needed, this is known due to software requirements, although not always made explicit in the CQs. In such cases, it is important to be able to check whether the ontology fulfills all those requirements (both CQs and additional inference requirements), i.e. can perform its intended task within the software. In addition, it is important to determine what data will be problematic to handle. Currently, there are as far as we are aware no detailed guidelines how to perform ontology requirement verification,

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 216–226, 2012.

and the notion of a *test case* is poorly investigated. Developers need practical and detailed guidelines for carrying out testing and storing test cases.

Ontology evaluation is the process of assessing an ontology with respect to certain criteria, using certain measures [17]. The criteria can range from selection criteria for reuse, to detailed criteria of the internal logical structure. In this paper we are concerned with evaluating the functional perspective [4]. To evaluate the functionality of an ontology, we can use its ontological requirements as a specification of its intended task, hence, comparable to requirements verification in software engineering. We can thereby describe ontology testing as a *task-focused evaluation* of ontologies against their *ontological requirements*, such as competency questions and inference requirements. This notion of ontology testing, and the methodology to be presented later, is related to current literature as described in Sect. 2. The paper continues by describing the notion of a "test case" in Sect. 3, our proposed methodology in Sect. 4, as well as our tool in Sect. 5, before concluding in Sect. 6.

2 Background and Related Work

In software engineering, verification focuses on checking that the software meets its specified requirements [18], hence, our notion of ontology testing corresponds to *verification*. As opposed to validation, where the software is checked also against user expectations [18], or in the case of ontologies verified against the "intended model" of the world [17], which is a quite abstract notion. Software testing has two main purposes [18]; (i) demonstrating that the software fulfils its requirements, and (ii) detecting faults, e.g. cases when the behaviour is inaccurate or undesirable. Ontology testing also needs to cover both cases. Although sometimes felt to be unintuitive, the result of a software test run is considered as *positive* if a fault is detected, and *negative* if the system behaves as expected [18]. Unit testing [9] focuses on the functionality of a small piece of code, as opposed to integration and system testing. Commonly, each unit test is stored as one separate software class, together with metadata [9].

Numerous ontology engineering methodologies exist [7,20,3,15,14,19], where *agile* development is the most test-intensive, e.g. eXtreme Design (XD) [16]. Ontology Design Patterns (ODPs) [5] support various ontology engineering tasks, where some ODPs are available as small ontologies (i.e. Content ODPs - CPs), and are a focus of the XD methodology. The use of ODPs introduces a natural modularization, where *module* simply refers to an ontology but restricted in terms of the functionality (i.e. requirements) it provides. The test methodology in this paper can, for instance, be used as a part of the XD methodology.

Most methodologies mention evaluation, either for ontology selection or for assessing the resulting ontology, but specific methods are rarely prescribed (for an overview of ontology evaluation see instead, e.g., [17]). This paper focuses on ontology evaluation from a functional perspective [4], where for application ontologies we need to evaluate the query- and inference capabilities. Unit testing for ontologies has also been proposed [21], applying ideas similar to ours,

e.g. queries as unit tests for ontologies, but no comprehensive methodology and guidelines for ontology engineers were provided.

The OWL Unit Test Framework [22,10] is one of few testing tools. It contains tests ranging from heuristics to detection of OWL-DL modelling problems. The tool attaches unit tests to individual named classes, through annotation properties. The method focuses on syntactical and logical constructs, i.e. task-independent features. Similarly, there are a number of other ontology "debugging" tools, for instance see [11,12,2], as well as the XD Analyzer within XD Tools itself [1]; many of them exploiting similar heuristics as the OWL Unit Test Framework. These are, however, all focused on the structural level rather than the functional one, since they do not take application (or domain) specific requirements into account. The tool that is most similar to ours is OntologyTest [6], however, it does not apply principles for separation of concerns, i.e., for separating test cases from test runs, testing one requirement at a time, and separating test data from the ontology. Additionally, the test types of OntologyTest are not focused on how users perform testing but rather on reasoning tasks, and the tool is not implemented within any ontology engineering environment.

3 The Notion of a *Test Case*

A test case is a container for storing information about a test, analogous with software unit testing [9], applying best practices such as keeping tests separate from the actual program code, and creating separate test cases for each requirement. We choose to represent a "test case" as a separate OWL ontology, identified through its unique URI, normally being only an ABox relying on the TBox of the ontology to be tested and our test case metamodel (see below). This constitutes a considerable difference compared to the OWL Unit Test Framework [10], where annotations were associated with single elements, or the OntologyTest [6] where test information is stored as application-specific XML.

In order to realize this in practice we have created an OWL ontology containing concepts for describing tests , i.e. a test metamodel. The main concepts are illustrated in Figure 1. `TestCase` instances are of type `owl:Ontology`, and each test case is related to a requirement, and a test procedure, e.g., a SPARQL query in case of CQ verification. A `TestCase` is a member of a `TestSuite`, which is a collection of test cases for testing an ontology. The `TestSuite` relates to what we call an `OntologyHistory`, which is a collection of `owl:Ontology` consisting of all the versions of that ontology during its development. When a `TestCase` is executed over one specific version of the ontology, with one specific set of test data, at a certain time, it constitutes an instance of `TestCaseRun`, again of type `owl:Ontology`. For each time the test is executed, a new `TestCaseRun` is described. The metamodel provides the opportunity to store several pieces of information about test cases and runs, such as the tested requirement, type of test, test procedure, test data, expected as well as actual output, person responsible, in what environment it was executed, execution date, comments, etc.

For unit testing, i.e. testing of single requirements, it is recommended to create one test case per requirement in order to separate concerns and ease the analysis

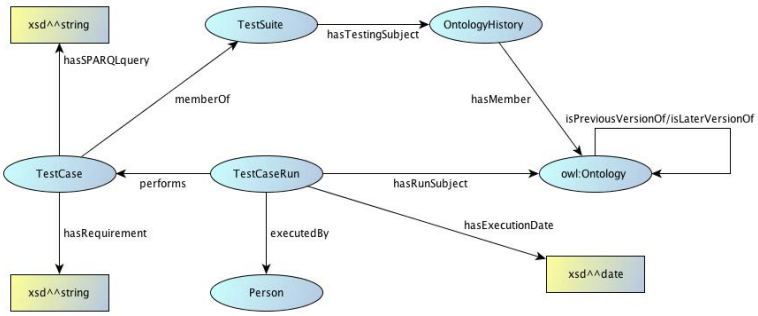


Fig. 1. An illustration of parts of the `TestCase` meta model in OWL

of potential errors. For integration testing, i.e. testing the overall ontology, a test case can contain larger sets of test procedures and potentially utilize test data for the complete ontology, in order to also capture interactions between requirements. The collection of `TestCaseRun`s creates an incrementally growing test dataset for the project. With minimal effort the data can be composed to ensure the ontology can deal with the complete set in the end.

4 Methodology for Ontology Testing

Based on related work and our own experience with both performing and teaching ontology engineering, a general methodology for performing ontology testing has emerged. In addition, we have observed three main instances of this methodology; (1) CQ verification, (2) inference verification, and (3) error provocation. The first two are concerned with verifying the correct implementation of a requirement, e.g., a CQ or an inference task, while the third concerns is intended to expose faults. The latter is important since the aim of testing, c.f. software testing, is not only to review the successful cases but to find the cases where the ontology fails. Testing each requirement, e.g., each CQ, can be considered as a *unit test* for the particular part of the ontology realizing that requirement [21]. Nevertheless, the same methodology can also be used for *integration testing* and testing the capabilities of the *overall ontology*, although it is not exemplified in this paper. The methodology is described in Table 1, and below we exemplify the three types of tests.

CQ Verification Example: Assume that we are creating an OWL ontology about families, e.g. parents and children. One module of the ontology realizes the following CQ: "Who are the children of this person?". A possible reformulation of the CQ into a SPARQL query could be (disregarding namespaces): `SELECT * WHERE {?x a :Person . ?y a :Person . ?x :hasChild ?y}` Test data is added to the test run file, e.g., individuals of the `Person` class such as `Bob`, `Mary`, and `Jill`, where some of the instances are related through `hasChild`, e.g., `Bob - hasChild - Jill`. Expected result is the pair `Bob` and `Jill`. Actual results of

Table 1. Overview of the methodology

1	Gather requirements	For a specific type of test, retrieve all the requirements of the current module that are relevant to this test type.
2	Select requirement	Following the principle of unit testing, select one requirement to test in each *test case*.
3	Test procedure	Determine how to test that particular requirement.
4	Create test case	Create the test case OWL-file, and an additional OWL-file for storing the first test run (importing the version of the ontology to be tested), and describe both using the test case metamodel and its properties.
5	Add test data	Add the test data, needed to perform the procedure according to step 3, in the test run OWL-file (or a separate data module).
6	Determine expected results	Depending on the test data, what would be the output of a negative test run, i.e. where the requirement is fulfilled?
7	Run test	Execute the procedure from step 3 on the test run OWL-file with its data from step 5, and record the results.
8	Compare results	Verify the expected output (step 6) against the actual result from step 7.
9	Analyze unexpected result	If the result is not the expected one, i.e. a positive test result, analyze why and document any change suggestions or issues.
10	Document	Store all information about the *test run* and its related *test case* by using the properties of the test metamodel.
11	Iterate	If there are more requirements to test, return to step 2.

the SPARQL query can be automatically verified against the expected output. Documentation is added to the test run using metamodel properties.

Inference Verification Example: CQs usually do not specify how information is produced, i.e., entered as assertions or derived from other facts through inferencing. Additional inference requirements can complement CQs. Using the above example we may want to define the class `Parent` and say that any person who has at least one child is a parent. This can be derived from the presence of `hasChild` relations through classification, rather than adding this information explicitly, assuming the ontology includes the appropriate axioms. Test data include `Person`s that have `hasChild` relations, and some that have none. If `Bob - hasChild - Jill`, the expected result is that `Bob` is classified as a parent. Using an OWL reasoner expected results are verified, and then documented.

Error Provocation Example: Ideally the ontology allows exactly the desired inferences and queries, while avoiding any irrelevant or erroneous side-effects. This category of testing, "error provocation", is comparable to software testing when a system is fed randomly generated or deliberately incorrect data, or data considered as "boundary values". Detecting undesired effects in an ontology can be done in several ways, one way is consistency checking. Assume a common sense constraint that a person cannot be both male and female, whereas instances of `MalePerson` and `FemalePerson` are added to the test data, and at least one is an instance of both. The expected result is an inconsistency.

5 Ontology Testing Tool Support

XD Tools is an Eclipse plugin (perspective), compatible with the NeOn Toolkit ontology engineering environment. It mainly supports the XD ontology engineering methodology, but contains several components that can be useful for ontology engineering in general [1], e.g., the XD Analyzer. The ontology testing tool is implemented as a part of the XD Tools.

5.1 Gathering Requirements of Tool Support

To investigate how the methodology is actually used, and to determine what tool support is needed, we have used the methodology (during its development) for teaching advanced ontology engineering in several settings, e.g., PhD courses[1] as well as tutorials in other organizations, such as the Swedish Defence Research Agency[2]. In addition to observing the use of the methodology, in one of the PhD courses we performed a small study, where half of the group were given the methodology (including the three variants) described above, and half were asked to test by inventing their own methods. No specific tool was provided. After the session they filled out a questionnaire. It turned out that both groups focused mostly on formulating SPARQL queries, while only the group given detailed guidelines also tested inferencing capabilities. Formulating SPARQL queries was also found quite tedious, and some support and "templates" were requested. It was also found quite unintuitive by the participants having to create their own annotation properties (at the time our metamodel was not available), or use general comments for storing queries and other information.

5.2 Ontology Testing Plug-in for XD Tools

The testing functionality[3] appears in the user interface as items in the contextual menus of XD Tools, i.e., when right-clicking on an ontology in the ontology navigator view. These options include to *create a new test case*, which will open a dialogue that in the end results in creating a new test case. The second menu option is to *open as test case*, which assumes that the marked ontology imports the test metamodel, i.e., is a test case, from which metadata will be loaded. Finally, a test case ontology can be fetched through the *import test case* option.

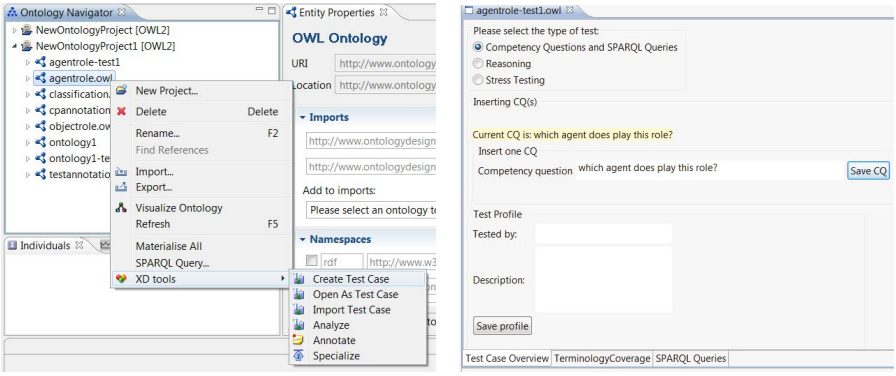
Once a test case is opened, the test case view will appear at the bottom of the XD Tools perspective as a set of tabs. The first tab is the overview (see Figure 2), where general information and the requirement to be tested is entered.

When selecting the CQ verification test variant, the tab following the overview gives the opportunity to assess the terminological overlap between the CQ and ontology elements. The third tab then provides an interface for writing, saving, and executing SPARQL queries. The user can enter expected results, and when the SPARQL engine is executed (on the asserted or inferred model), the actual results are automatically compared to the expected ones and displayed to the user (additional unexpected results are marked in yellow). If one of the other test variants is selected, only one additional tab is shown, where the user can

[1] http://ontologydesignpatterns.org/wiki/Training:PhD_Course_on_Computational_Ontologies_%40_University_of_Bologna_2011

[2] http://ontologydesignpatterns.org/wiki/Training:Advanced_Ontology_Engineering_at_FOI_-_2011

[3] Currently the software requires the use of NeOn Toolkit version 2.3.2. To install the test functionality, first install XD Tools and then place the two jar-files in your plugin folder (replacing existing ones with the same name), files can be found at: https://sourceforge.net/projects/ontologytesting.



(a) Menu alternatives for testing (b) The overview tab, with one CQ

Fig. 2. Creating a new test case using the context menu (a) and overview tab (b)

enter expected results, run an inference engine and then verify the actual results against the expected ones, e.g., see Figure 3.

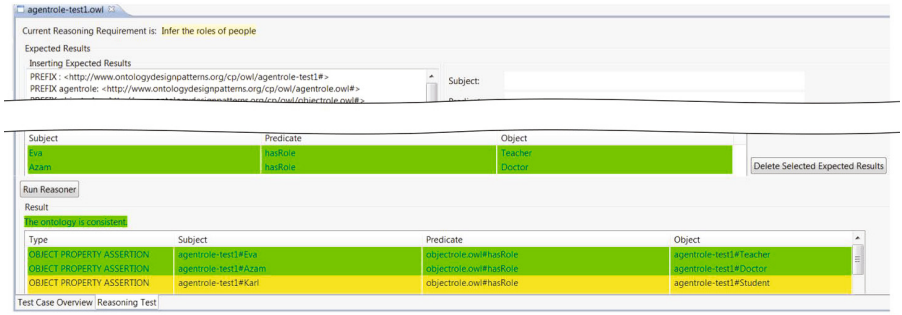


Fig. 3. The tab for reasoning tests, where expected results inserted through the topmost form are shown in green (middle) if confirmed in the actual results (bottom). Yellow indicates additional inferences.

5.3 SPARQL Suggestions

Although experiments show that testing CQs through SPARQL queries finds simple mistakes such as missing elements (c.f. [1]), manually writing SPARQL queries from scratch is perceived as tedious routine work (Sect. 5.1). Methodologies, such as XD, suggest formulating tests already at design-time, and directly formalize them as SPARQL queries, however, in practice this is not often the case. In our experience the most common way to document requirements is informal sentences. The aim of our SPARQL generation method is to produce good suggestions that are useful to the user, which should be seen as "templates" (although sometimes we are able to generate the query completely automatically).

SPARQL suggestions are activated via a button in the third tab of the CQ verification test type in our tool. The algorithm relies on the following steps[4]:

1. Analysing the CQ linguistically in order to form a set of linguistic query triples, and identifying the type of question (for details see [13]).
2. Matching the linguistic triples to the triples of the ontology, to find candidate classes and properties to be involved in the query.
3. Among those properties, finding structural paths that connect the most relevant classes. Path structure depending on the type of the query.
4. Selecting the best matching paths, based on ranking their length (short paths are preferred over long ones), structural match to the query type, and linguistic matches to the CQ terminology.
5. Generating a query, based on selected classes and property path(s).

Step 1 is realised through methods for question analysis from the AquaLog and PowerAqua systems [13]. As an example, when running this method over the OWL building block of the `AgentRole`[5] CP containing the CQ "Which agent does play this role?", the linguistic analysis results in the linguistic triple: (agent ; play ; this role). Classes `Agent` and `Role` will be selected as candidates, and paths are found between those classes based on logical axioms, e.g. domain and range, restrictions on classes, as well as subclass reasoning for "inheriting" such axioms. For our example, there are four object properties available in `AgentRole` (including all its imports); `hasRole` and `isRoleOf`, as well as `classifies` and `isClassifiedBy`. All of them can connect instances of `Agent` and `Role`, hence, four paths are found. The paths are all of length one, resulting in the `hasRole` property being selected due to its string similarity with a term in the linguistic triple. Using the two classes and the `hasRole` property, the following SPARQL query is generated (for readability reasons we omit all namespace prefixes): `SELECT * WHERE {?x a :Agent . ?y a :Role . ?x :hasRole ?y}`.

The SPARQL suggestions were evaluated to confirm that the suggestions given are likely to be useful. The evaluation was performed over ontologies and repositories where CQs are readily available, such as the Content ODP (CP) OWL building blocks present in the ODP portal[6], the IKS AmI ontologies [7], and two slightly larger ontologies from different domains, i.e., the FSDAS ontology network[8], and an ontology of small molecules within the biological domain[9].

In summary, the evaluation showed that there are very few erroneous results (0 in all cases but ODP, which has 2% error rate), however, between 38% (AmI

[4] The implementation is available as source code for download at `https://sourceforge.net/projects/ontologytesting`

[5] Available at: `http://ontologydesignpatterns.org/wiki/Submissions:AgentRole`

[6] `http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs`

[7] For details on the project see `http://www.iks-project.eu/` Ontologies available at: `http://www.ontologydesignpatterns.org/iks/ami/2011/02/`

[8] Ontology of the FAO, which models the fish stock domain: `http://www.ontologydesignpatterns.org/cp/owl/fsdasnetwork.owl`

[9] `http://www.owl-ontologies.com/SMO.owl`

and FSDAS) and 61% (CP) of the CQs do not yield any suggestions. Between two thirds and 100% of the suggestions that are given are correct (even if not complete in some cases). It seems much more difficult to match CQs in the CP repository, than in the other datasets, which is not highly surprising since most of the CPs are very general and at that level there is not a specific enough terminology. While in the most specific cases, i.e., the AmI repository and FSDAS, we are able to produce useful suggestions in over 60% of the cases. We also see that the missing suggestions are not evenly distributed, but usually either an ontology has some suggestions for all its CQs or it has none at all. This could be attributed to the modelling style of that ontology, e.g., if no domain, range or other axioms are given, our method will not find any paths. This highlights that providing SPARQL suggestions is not only a matter of convenience, but also gives us important information about the ontology, e.g., concerning lack of terminology coverage, or that modelling best practices are not being followed. These observations point at the possibility of also using our method for such debugging, but this is still future work.

6 Conclusions

This paper has described the notion of ontology testing for application-oriented ontologies, analogous to software testing. We show that ontology test cases and test runs can be represented as ontologies themselves, described through a metamodel. We have presented a general test methodology and three examples, as well as a tool to support ontology testing. In addition, we have shown that automatic test generation, through suggesting SPARQL queries directly from requirement CQs, is feasible and can provide interesting insights even when no suggestion is found. The main contributions of this paper include the practical notion of a test case and the metamodel to describe it, the detailed description of the methodology, as well as the method for automatically generating test suggestions as starting points for writing SPARQL queries.

Future work includes to thoroughly evaluate the testing tool with a larger user base. Focusing on the details of the tool support, the next step is to add support for distinguishing between the test case and the test run, and with respect to the SPARQL suggestions we will also investigate how to provide useful debugging-information to the user even when no suggestion is available. So far the focus has been on support for unit testing, it still needs to be further investigated what modifications are needed to fully support integration testing.

Acknowledgements. Thanks to Vanessa Lopez, KMI (Open University), for providing the linguistic component for the SPARQL suggestions. This project was supported by the European Commission, through project IKS (FP7 ICT-2007-3/No. 231527), and The Swedish Foundation for International Cooperation in Research and Higher Education, project DEON (#IG2008-2011).

References

1. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with eXtreme Design. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 120–134. Springer, Heidelberg (2010)
2. Djedidi, R., Aufaure, M.-A.: *ONTO-EVOAL* an Ontology Evolution Approach Guided by Pattern Modeling and Quality Evaluation. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 286–305. Springer, Heidelberg (2010)
3. Fernández, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: from Ontological Art towards Ontological Engineering. In: Proceedings of the AAAI 1997 Spring Symposium Series on Ontological Engineering (1997)
4. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Modelling Ontology Evaluation and Validation. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 140–154. Springer, Heidelberg (2006)
5. Gangemi, A., Presutti, V.: Ontology Design Patterns. In: Handbook on Ontologies, 2nd edn. International Handbooks on Information Systems. Springer (2009)
6. García-Ramos, S., Otero, A., Fernández-López, M.: OntologyTest: A Tool to Evaluate Ontologies through Tests Defined by the User. In: Omatu, S., Rocha, M.P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J.M. (eds.) IWANN 2009, Part I. LNCS, vol. 5518, pp. 91–98. Springer, Heidelberg (2009)
7. Grüninger, M., Fox, M.: Methodology for the Design and Evaluation of Ontologies. In: Proc. of IJCAI 1995, Ws. on Basic Ontological Issues in Knowledge Sharing (1995)
8. Gruninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: Proc. of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice (1994)
9. Hamill, P.: Unit Test Frameworks - Tools for High-Quality Software Development. O'Reilly Media (2004)
10. Horridge, M.: The OWL Unit Test Framework, `http://www.co-ode.org/downloads/owlunittest/`
11. Liu, Q., Lambrix, P.: Debugging the Missing Is-A Structure of Networked Ontologies. 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. 478–482. Springer, Heidelberg (2010)
12. Lambrix, P., Liu, Q.: A System for Debugging is-a Structure in Networked Taxonomies. In: Demo at the International Semantic Web Confrerence-ISWC 2011, Bonn, Germany (2011)
13. Lopez, V., Uren, V.S., Motta, E., Pasin, M.: Aqualog: An ontology-driven question answering system for organizational semantic intranets. Journal of Web Semantics 5(2), 72–105 (2007)
14. Nicola, A.D., Missikoff, M., Navigli, R.: A software engineering approach to ontology building. Journal of Information Systems 34(2), 258–275 (2009)
15. Pinto, H.S., Staab, S., Tempich, C.: DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: Proceedings of of ECAI 2004, Valencia, Spain (2004)
16. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme Design with Content Ontology Design Patterns. In: Proc. of WOP 2009, Collocated with ISWC 2009. CEUR Workshop Proceedings, vol. 516 (2009)
17. Sabou, M., Fernandez, M.: Ontology (Network) Evaluation. In: Ontology Engineering in a Networked World, pp. 193–212. Springer (2012)

18. Sommerville, I.: Software Engineering, 8th edn. Addison-Wesley (2007)
19. Suarez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M.: The NeOn Methodology for Ontology Engineering. In: Ontology Engineering in a Networked World, pp. 9–34. Springer (2012)
20. Uschold, M.: Building Ontologies: Towards a Unified Methodology. In: Proceedings of Expert Systems 1996, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK (December 1996)
21. Vrandečić, D., Gangemi, A.: Unit Tests for Ontologies. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1012–1020. Springer, Heidelberg (2006)
22. Wang, H., Horridge, M., Rector, A.L., Drummond, N., Seidenberg, J.: Debugging OWL-DL Ontologies: A Heuristic Approach. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 745–757. Springer, Heidelberg (2005)

A Model of Derived Roles

Kouji Kozaki, Yoshinobu Kitamura, and Riichiro Mizoguchi

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

Abstract. Roles are important concepts in order to consider practical instance managements. Although roles have been discussed by many researchers, there remains some room to investigate to clarify ontological characteristics of them. This paper focuses on roles which are dependent on the future or past event/process, such as candidate, departing passenger, murderer, and product etc. In order to deal with such kinds of roles based on an ontological theory of roles, we introduce a model of derived roles with its temporal model. It could provide a computational model to represent temporal characteristics of roles.

Keywords: ontological engineering, role, derived roles, temporal model.

1 Introduction

It is very important for ontology building to distinguish clearly among concepts in the real world based on their characteristics. Dependences of things are one of key characteristics which we should pay attention when we develop ontologies. So, how to deal with "dependency" is one of the key technologies in ontology building. Among them roles such as *customer*, *president*, *pedestrians*, etc. are dependent on other entities. A company can be a *customer* of another company while being a *supplier* to others. Proper treatment of roles is crucial to building a good ontology. This is why the topic of roles has been investigated extensively in several areas of ontology engineering[1-6], biomedical[7], database model[8], software engineering[9], and agent systems[10] etc. In practical point of view, in order to consider identity of some entities for instance management, it is important to understand characteristics of roles[11].

Although roles have been discussed by many researchers, there remains some room to investigate to clarify ontological characteristics of them. For examples, is *murderer* a role? Some answer no. A reason would be because one cannot stop being a *murderer* once he/she has started to play it. Another would be "it is odd to say "he plays a murderer" if not in a drama". Although these reasons are reasonable to some extent, we need a convincing explanation of what *murderer* is ontologically. This paper focuses on roles which are dependent on the future or past event/process, such as *candidate*, *departing passenger*, *murderer*, and *product* etc. In order to deal with such kinds of roles based on an ontological theory of roles, we introduce a model of derived roles with its temporal model.

This paper is organized as follows. The next section overviews our model of roles discussed in [3] to provide readers with background of the discussion of a new model

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 227–236, 2012.

of roles. In Section 3, we propose a new view of roles to distinguish **derived roles** from **original roles**. In Section 4, we ontologically analyze temporal issues of derived roles in the context of enumeration of kinds of roles. Finally, we present concluding remarks together with future work.

2 Overview of Our Role Theory

The fundamental scheme of our roles at the instance level is the following (see the lower diagram in Figure 1.):

"In Osaka high school, John plays teacher role-1 and thereby becomes teacher-1"

This can be generalized to the class level (see the upper diagram in Fig.1):

*"In schools, there are **persons** who play **teacher roles** and thereby become **teachers**."*

By **play**, we mean that something "acts as", that is, it contingently acts as according to the role (role concept). By "**teacher**", we mean a class of dependent entities which roughly correspond to a person who is playing teacher role and which is often called a *qua individual* [2] or *relational tropes* [6]. Our theory introduces a couple of important concepts to enable finer distinctions among role-related concepts: ***role concept***, ***role holder***, ***potential player*** and ***role-playing thing***.

By **context**, we mean a class of things that should be considered as a whole. Unitary entities and relations can be a context of its parts and participants, respectively. **Role concept** is defined as a concept which is played by some other entity within a context. So, it essentially depends on the context[4]. By **potential player**, we mean a class of entities which are able to play an instance of a role concept. In many cases, potential players are basic concepts (natural types). When an instance of potential player is playing an instance of a role concept, we call the instance a **role-playing thing**. In this example, we say a person can play an instance of a teacher role. In particular, John is actually playing a specific teacher role, *teacher role-1*. By doing so, he is associated with the instance *teacher-1*, an individual teacher **role holder**. A role-holder class is a class whose instances include, say, *teacher-1*. As such, it is neither a specialization of a potential player class (e.g., person) nor that of a role concept class (e.g., teacher role), but an abstraction of a composition of a role-playing thing and an instance of role concept, as is shown in Fig.1, which is the heart of our role model.

In this paper, although the term "role concept" is important, we use "role" to denote" role concept" for notational simplicity.

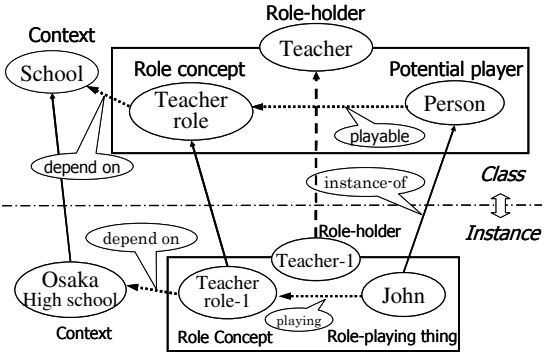


Fig. 1. Fundamental scheme of a role concept and a role holder

3 Original and Derived Roles

3.1 Classification of Role

The issue here is the fact that there are many kinds of roles other than what we usually see in the literature on roles such as *teacher*, *president*, *wife*, *patient*, etc. According to the types of the context on which roles are dependent, we can identify two kinds of roles such as those dependent on continuant and those on occurrent. Many of the popular roles including those mentioned above belong to the former. On the other hand, there are quite a few roles depending on processes or events. They include actor roles such as *driver* and *runner*, task roles such as *symptoms* and *fault hypothesis* played by states in the context of diagnostic tasks, functional roles such as *level-control valve* and *steering wheel* of a bike, artifact role such as *table role* which a box plays, etc. All these roles mentioned thus far are **ongoing**, by which we mean the context the role depends on is present when the role is being played. Surprisingly, at first glance, there seem to be quite a few roles whose contexts are not present when they are played. Typical examples include *murderer*, *culprit*, *witness*[1], *victim*, *product*, *residue*, etc[5]. However, as we see it below, those roles are not **original roles** but **derived roles** derived from the corresponding original roles. We have to be very careful not to be caught by language expression-based justification/understanding of roles. Roles should be understood as the original definition defined directly in the context.

Original and derived role holders are defined as follows:

Def. 1 Original role holder (ORH) =def *role holder defined as a participant of its depending occurrent or relation as its context or role holder defined as a part of an object as its context.*

Def. 2 Derived role holder (DRH) =def *non-original role holder which forms a role based on a content-oriented reference to corresponding original role holder (ORH), shares the same name with ORH but has different meaning (role) from ORH, and the time of playing of DRH differs from that of ORH.*

A typical example is *murderer* which means "a person who killed a person" in English and it seems to be a role holder with a historical property. Considering the role model discussed in section 2, however, we can find an original *murderer* role holder which means "a person who has just completed a killing action". Then, we can consider "a person who killed a person" as a derived role holder derived from the original role holder.

Linguistically, *examinee* means a person/student who works/studies hard to pass an (entrance) exam, so it seems to be a prospective role holder. However, it should be a derived role holder because it has no direct context of the taking exam as a process. Its original role holder should be "a person/student who is taking an (entrance) exam" defined in the context of an exam-taking process. The former role holder which has a prospective property should be understood as a derived one from the original one.

Another example suggesting a danger of relying on linguistic justification would be *(biological) child* in which people would ignore the difference between the instantaneous event of being born and its persisting existence after it. It is true that it is very

[1] This is not a person who testifies what his/her saw in a court but a person who saw the event.

special ontologically in the sense that the player (John) is born at the same time when the new entity as a *(biological) child* (role holder) of his parents is born. Concerning the above difference, however, *(biological) child* and *murderer* share the same characteristic, and hence *(biological) child* should be understood as a role holder defined at the very time of having been born. Both *child* and *murderer* carry the property after the appearance event.

We know the above explanation is rather informal. Although we deeply discusses on this topic in other paper[12], let us here summarize the above observation as follows: All original roles are ongoing. Roles which seem to be retrospective and/or prospective roles are derived roles derived from original roles. These are represented in the taxonomy of roles shown in Fig.2 which clear distinction between original and derived roles is made at the top-level.

Role
- **Original role (ongoing)**
  - non-participatory role (continuous)
    - *wife, patient, teacher, friend, etc.*
  - occurrent-dependent role
    - process-dependent role (continuous)
      - dynamic
        - *speaker/listener,runner,symptom, witness, examinee*
      - static
        - *baby/infant/adult, the sick*
    - event-dependent role (instantaneous)
      - *murderer,victim,residue,conclusion, mother,(biological) child, departing passenger*
- **Derived role (non-ongoing)**
  - retrospective
    - *murderer as a person who had killed a person*
  - prospective
    - *examinee as a person who studies hard to pass an exam*

Fig. 2. The taxonomy of roles

4 Ontological Analysis of a Temporal Aspect of Derived Roles

We built an ontological model of the temporal aspect of derived role holders as shown in Fig.3 to examine them in detail. In terms of the temporal model of derived role holders, it seems we can introduce a two-class classification into role holders, in which we call ***prospective derived role holder*** such as *examinee* as a person who studies hard to pass an exam and *departing passenger* and ***retrospective derived role holder*** such as *murderer* as a person who had killed a person, *victim* and *(biological) mother*. As will be discussed below, when we consider role holders derived from an occurrent- dependent role holder, any original role holder has three variants associated with it. In the following, in order to represent them systematically, we introduce three names: *derived role holder1* to *derived role holder3* as follows:

Derived role holders (denoted as ***DRH***)

*Derived role holder1 (denoted as **DRH_ev**): Retrospective or prospective reference to the player of an event-dependent role holder*

*Derived role holder2 (denoted as **DRH_pr**): Retrospective or prospective reference to the player of a process-dependent role holder*

*Derived role holder3 (denoted as **DRH_ag**) : Aggregation of derived role holders from an occurrent-dependent role holder*

In the following sections, we denote *derived role holder1* to *derived role holder3* as ***DRH_ev***, ***DRH_pr*** and ***DRH_ag*** respectively, and we denote the roles of these derived role holders as ***DR_ev***, ***DR_pr*** and ***DR_ag***. For example, by

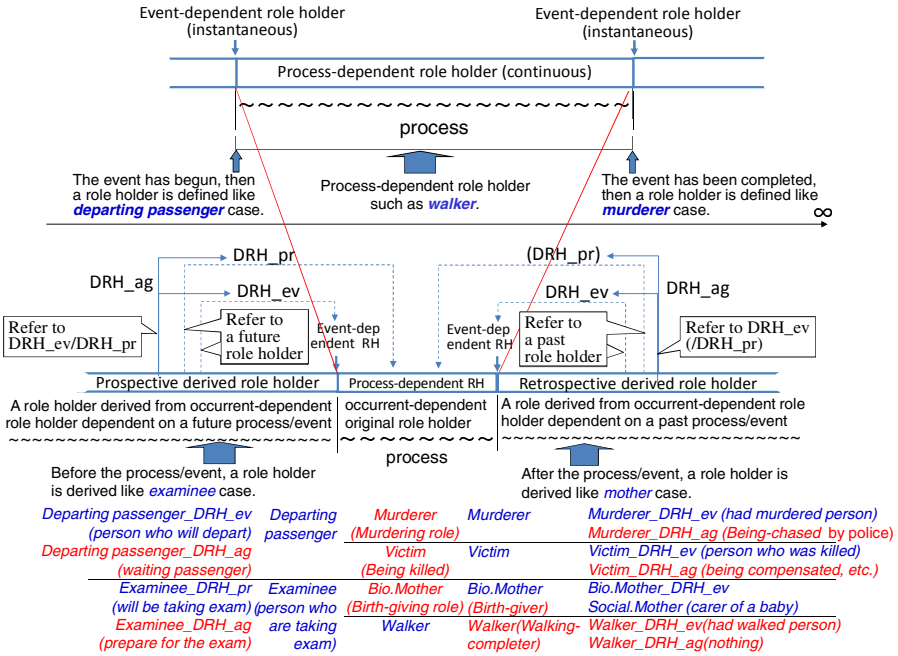


Fig. 3. An ontological model of occurrent-dependent roles

murderer-DRH_ev and *murderer-DR_ev* we mean *murderer as a derived role holder1 (DRH_ev)* and *its role,* respectively.

4.1 Retrospective Derived Role Holder

By retrospective derived role holders, we mean role holders which are derived from an original role holder defined dependently on past processes/events. By past, we mean before the playing time. For example, a *murderer-DRH_ev* as a person who killed a person is derived from an original role holder, *murderer,* dependent on a past killing event in which its player participated. To consider how retrospective derived role holders are dependent on past processes/events, we define the above three kinds of derived role holders for each original occurrent-dependent role holder. For simplicity, we take up the case of actor role in the following discussion.

In the case of *walker*, *walker as a process-dependent role holder* corresponds to normal *walker* which denotes "a person who is an actor of walking process", and *walker as an event-dependent role holder* denotes "a person who has just completed a walking action" which is odd in reality. Event-dependent role is the role defined as being born at the very moment of the completion of an event and hence it cannot have any chance to work during the event, though it is played only at that instance of time. Similarly, a *murderer as event-dependent role holder* appears only at the very moment of the completion of the killing event. How can we refer to the *murderer* after the murder event, then? Here, we introduce *DRH_ev* to represent the reference to an *event-dependent role holder*. *DRH_ev* is a role holder defined as "a person who had

completed an action" to refer to the person and we call it "**content-oriented reference**[2]". That is, *DRH_ev* is derived from an action completer role holder (*event-dependent role holder*). For example, when people say "He is a murderer", they do not mean a *murderer*, which is realized at only the very end of the killng event, but a person who had committed the killing event. *Murderer-DRH_ev* can be understood as a "name" to specify the referent in a content-oriented way. When we refer to an entity through *DRH_ev*, it is not necessary that a player of the referent role is present in reality at the time of reference, since the reference to the player of *the original role holder* which the *DRH_ev* specifies is always successful independently of the existence of the player who must have existed when the *original role* was played.

On the other hand, a *murderer* is expected to play a *target role* to be caught by the police and/or to be punished by justice. But, it is not a *murderer-DRH_ev* since a *murderer-DRH_ev* is a referent to a participant of a past event while the target of police must be a real person who killed someone. That is, we can consider the role as another kind of *murderer* role which is expected to be played by the player of *murder*. In order to represent this kind of role, we introduce *DRH_ag*. In the murderer case, *murderer-DRH_ag* is defined as aggregation of "*a person who is being chased by police"* or "*a person who is being punished by justice"*, etc.

The problem, however, is that they are not what "murderer" literally means. Those role holders are derived from the *murder* and their roles are expected to be played by the person who plays *murderer role*. Therefore, these roles are not only "*being chased by police"-role* or "*being punished by justice"-role* but also "*being chased as a murderer by police"-role* or "*being punished as a murderer by justice"-role*. *Murderer-DR_ag* should be defined as an aggregation of these roles which are expected to be played as *murderer* in reality.

Here, we have to note that a *DRH_ag* does not have to have real players in some cases. For example, a *victim* might be a person who was killed by a murderer. If we consider such a victim as a *DRH_ag*, the player of *victim-DRH_ag*, which is the person who was killed, does not exist in reality because he/she has been dead. Although most of the derived role holders such as "*a person who is being chased by police"* or *experience-teller* need a real player, target of action by others such as "*target of sympathy*" do not need real players if the target can be identified. In the case of *murderer-DRH_ag*, "*blameworthy target as a murderer"-role* could be played even if the murderer is dead, while "*being chased as a murderer by police"-role* needs a real player. It eventually depends on derived roles whether or not a real player is necessary to play the role.

Next, we introduce *DRH_pr*. Although *DRH_pr* is also a content-oriented reference related to a past event, it focuses on a different time point from *DRH_ev*. *DRH_pr* does not refer to *an event-dependent role holder*, but *a process-dependent role holder*. That is, *DRH_pr* is a reference to a role holder dependent on an ongoing process which is a constituent of the past event while *DRH_ev* is reference to a role holder dependent on a completion of the process/event. This is the difference between *DRH_ev* and *DRH_pr*. Note that both can be either retrospective or prospective references. For example, *DRH_pr* could be used as references to actors who are asked to explain how they performed the process in a situation where a *murderer* is accused

[2] The initial idea of "content-oriented reference" in this paper is also mentioned as "facons de parler" in P. 6 in the previous paper (but it is regarded as not a role but just a name) [5].

of the process of how he/she killed a person, or an *Everest summiteer* talks about how he/she overcame his/her difficulties in the climbing process[5].

However, murderer-DRH_pr which means a person who was murdering does not make sense because "murder" is essentially an accomplishment action. Moreover, in general, *DRH_pr* is minor in the retrospective case since such a role holder is realized by using the original role holder in the past tense such as "*a* walker was humming a song". This is partly because when we retrospect past events, we tend to regard their results as more important than their process in many cases. For example, a *murderer* is blamed for the result of killing a person, and *Everest summiteer* is respected for the result of reaching the summit of Mt. Everest. Therefore, *DRH_pr* and *DRH_ag* for it do not make sense to define theoretically.

As discussed above, in theory, we can define all of derived role holders for each of occurrent-dependent roles. However, all kinds of retrospective roles do not make sense well. While *walker as a process-dependent role holder* makes sense, *walker as an event-dependent role holder* and *walker-DRH_ev, walker-DRH_pr, and walker-DRH_ag* do not. While *murderer as an event-dependent role holder, murderer-DRH_ev* and *murderer-DRH_ag* do, *murder as a process-dependent role holder* and *murderer-DRH_pr* do not. *Mother-DRH_ag* makes sense very well, since a mother is expected to take care of her new-born baby. This clear difference is derived from the difference of the nature of actions constituting the event or process, that is, from the difference between actions which are of continuous process-oriented type like *walk* and those which are of achievement-oriented type like *murder*.

4.2 Prospective Derived Role Holder

By prospective derived role holders, we mean derived role holders which are defined dependently on future events. We can define three kinds of prospective derived role holders such as *DRH_ev*, *DRH_pr* and *DRH_ag* for each original occurrent-dependent role holder in the same way as retrospective role holders while directions to the depended process/event on the temporal axis are symmetrical (see Fig. 3).

In prospective role holders, an *event-dependent role holder* is defined at the very beginning of an event. For example, a *departing passenger* is defined dependently on a departure or travel event in the future. In the same way as *murderer*, *departing passenger role* is played only at the instant that the departure event begins. That is, *departing passenger-DRH_ev* represents reference to departing passenger as *an original event-dependent role holder*. Furthermore, we can define *departing passenger-DR_ag*, which is role expected to be played by the *departing passenger* such as "*waiting passenger*"-role, in the same way as retrospective roles. Although they make sense, *DRH_ev* and *DRH_ag* of *departing passenger* seem less meaningful than that of retrospective role holders such as *murderer-DRH_ev* and *murderer-DRH_ag*. This is caused by the point of time difference on which we focus to define these roles. After a completion of an event, we tend to consider its result to be more important than the process of completing the event. On the other hand, when we discuss about a future event, we consider that not only the beginning of the event but also the progress of the event are important. In order to investigate this tendency, we take an *examinee* as an example. An examination is an accomplishment type of event, so it fits better to talk about prospective roles than departure which finishes in a moment.

We can define an *examinee* as a prospective role holder which depends on the future examination. For a person who takes an exam, its progress is more important than its beginning. This suggests that we should define the *examinee* as not *DRH_ev* but *DRH_pr* which refers to the original process-dependent role holder depended on a future examination (taking an exam) process. *Examinee-DRH_pr* refers to the person who is going to take an examination and it is the prospective version of the content-oriented reference. The common meaning of the word examinee is a person who studies hard for preparation for an exam. Similarly to the murderer case, the actor role holder of such a derived role can be defined as *DRH_ag*, which is *examinee-DRH_ag*, to be played by the *examinee-DRH_pr*. Here, it is obvious that the purpose of the preparation is not to start to take an exam but to do his/her best through the whole process of the examination. This shows that *DRH_pr* is more important than *DRH_ev* in prospective derived role holders while *DRH_ev* is more important than *DRH_pr* in retrospective derived role holders. We suppose it is because most prospective roles are played to prepare for the future process on which they depend.

4.3 Discussion

In our co-authored previous paper [5], formal constraints of definitions of specific roles and some kinds of roles are discussed in a formal way. In particular, its main aim is to solve the famous counting problem. In the discussion of the kinds of roles, the initial idea of derived roles is discussed and then formalized as historical roles using an example of *the-climber-of-Everest* whose salient characteristic is that it lasts forever even after the death of him as well as possessing a historical property.

In this paper, we elaborate the ideas of "derived roles" based on "content-oriented reference" further and then propose a new theory of derived roles based not on a historical property but on a reference to (the player of) a past event. We found such kind of roles which seem to have a historical property should be decomposed into two types of the derived roles based on references, i.e., a reference to the player of an event-dependent role holder (*DR_ev*) and aggregation of roles based on such a reference (*DR_ag*). In addition the proposed model is symmetrical with respect to time as shown in Fig.2, which can cope with not only historical roles but also "future roles" mentioned in the footnote 16 in [5].

4.4 Computational Model of Derived Role Holder

Based on the above considerations, we built a computational model of derived role holders in our role model discussed in Section 2. Fig.4 shows an example of retrospective role holders represented by the extended role model. Although it is informal description, we can implement it using our ontology development tool Hozo and export in OWL format if a formal representation is needed.

An event-dependent role can be represented using the conventional model. The context for the event-dependent role is the completed event, and the relationship between them is represented by a usual depend-on link since the event exists at the time when the event-dependent role is defined. On the other hand, though the same event becomes the context for *DR_ev* and *DR_ag* as well, it is regarded as a past event when they are played. This is why relationships between the event and *DR_ev* and *DR_ag*

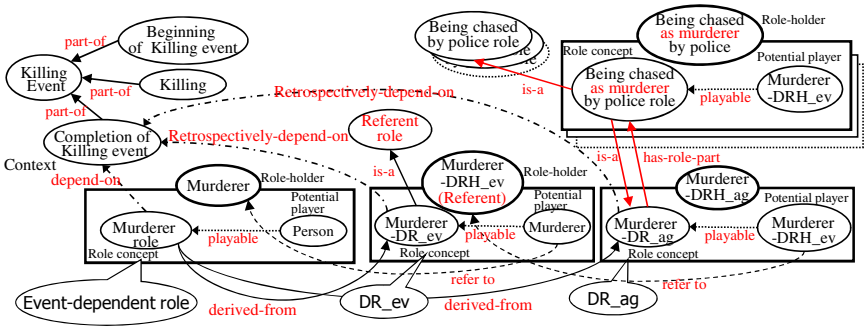


Fig. 4. An example of retrospective role represented by the computational model

are represented by retrospectively-depend-on links. Relationships between a derived role holder and its original role holder imply the derived role is derived from its original role. Derived-from links represent that *DR_ev* and *DR_ag* are derived from the original event-dependent role.

DR_ev is defined as a subclass of a referent role. The referent role holder represents the role holder such as "something to which is referred by the role as name". In Fig.4, *murderer-DRH_ev* represents the referent for "the person who is referred to as murderer". The player of *DRH_ev* is its original event-dependent role holder because it has to be the same instance with the player of the original role holder. It is represented by a framework for compound role we discussed in [3].

DRH_ag is defined as a role holder whose player is *DRH_ev* as referent for the event-dependent role holder. As we discussed, *DR_ag* is an aggregation of a couple of sub-roles. By has-role-part links between *DR_ag* and these sub-roles, we mean the *DR_ag* has those sub-roles as its pats. We also identify *is-a* relation between *DR_ag* and those sub-roles. It looks strange at first glance. However, it is not so peculiar considering cases where both *is-a* and *part-of* relations hold between a pie and piece of pie and between finger composed of five fingers and a finger. In Fig.4, *has-role-part* links represent the *murderer-DR_ag* is defined as the aggregation of its parts such as "being chased as murderer by police"-role or "being punished as murderer by justice"-role, etc. Each part of the *murderer-DR_ag* is defined as the subclass of *murderer-DR_ag* and expected roles such as "being chased by police"-role or "being punished by justice"-role, etc. using multiple inheritance.

Though Fig.4 shows only *DRH_ev* and *DRH_ag* in this example, we can model also *DRH_pr* in the same way by replacing the event-dependent role holder with the process-dependent role holder. We can also define prospective roles using the same model with several minor changes. To put it concretely, the context for *DRH_ev* becomes the future starting event, and relationships between the future event and *DR_ev* and *DR_ag* are represented by *prospectively depend-on* link.

5 Concluding Remarks

In this paper, we introduced a distinction between original and derived roles and a temporal aspect of them. Deriver roles derived from occurrent-dependent roles are

classified into two kinds such as prospective role and retrospective role. Basically, there is the symmetry between retrospective and prospective derived roles, and each of them has three variants such as *DRH_ev*, *DRH_pr* and *DRH_ag*. Then, we built a computational model to represent these roles based on our role model discussed in [3]. We believe this model could capture important characteristics of occurrent-dependent roles and it could contribute toward understanding temporal aspect of them.

Future work includes deeper investigation on derived roles and other kinds of roles. One of important issue is how to deal with derived roles which are derived from roles other than occurrent-dependent roles. What we precisely mean by "explicit role assignment", what is a vacant role, etc. are other topics to be discussed.

Acknowledgement. The authors are grateful for Nicola Guarino, Laure Vieu and Claudio Masolo for their comments. This research partially supported by *Eu-Joint:European-Japanese Ontology Interaction, Marie Curie IRSES Exchange Project n. 247503* and *Grant-in-Aid for Scientific Research (A)* 22240011.

References

1. Guarino, N.: Concepts, attributes and arbitrary relations: Some linguistic and ontological criteria for structuring knowledge bases. Data & Knowledge Engineering 8(2), 249–261 (1992)
2. Masolo, C., Guizzardi, G., et al.: Relational roles and qua-individuals. In: AAAI Fall Symposium on Roles, an Interdisciplinary Perspective, Hyatt Crystal City, Arlington, Virginia, November 3-6 (2005)
3. Mizoguchi, R., Sunagawa, E., Kozaki, K., Kitamura, Y.: The model of roles within an ontology development tool: Hozo. J. of Applied Ontology 2, 159–179 (2007)
4. Loebe, F.: Abstract vs. social roles - towards theoretical account of roles. Applied Ontology 2, 127–258 (2007)
5. Masolo, C., Vieu, L., Kitamura, Y., et al.: The Counting Problem in the Light of Role Kinds. In: AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning - Papers From, pp. 76–82 (2011)
6. Guizzardi, G.: Ontological Foundations for Structural, Conceptual Models. Phd, University of Twente (2005)
7. Arp, R., Smith, B.: Function, Role, and Disposition in Basic Formal Ontology. In: Proc. of Bio-Ontologies Workshop (ISMB 2008), Toronto, pp. 45–48 (2008)
8. Steimann, F.: The role data model revisited. J. of Applied Ontology 2, 89–103 (2007)
9. Baldoni, M., Boella, G., Torre, L.: Introducing Ontologically Founded Roles in Object Oriented Programming: powerJava. In: AAAI Fall Symposium on Roles, an Interdisciplinary Perspective, Hyatt Crystal City, Arlington, Virginia, November 3-6 (2005)
10. Colman, A., Han, J.: Roles, players and adaptable organizations. Applied Ontology 2, 105–126 (2007)
11. Kozaki, K., Endo, S., Mizoguchi, R.: Practical Considerations on Identity for Instance Management in Ontological Investigation. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS (LNAI), vol. 6317, pp. 16–30. Springer, Heidelberg (2010)
12. Mizoguchi, R., et al.: Ontological Analyses of Roles. In: 1st Workshop on Well-founded Everyday Ontologies-Design, Implementations & Applications (WEO-DIA) (to appear)

ONSET: Automated Foundational Ontology Selection and Explanation

Zubeida Khan and C. Maria Keet

School of Mathematics, Statistics, and Computer Science,
University of KwaZulu-Natal and
UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa
zkhan@csir.co.za, keet@ukzn.ac.za

Abstract. It has been shown that using a foundational ontology for domain ontology development is beneficial in theory and practice. However, developers have difficulty with choosing the appropriate foundational ontology, and why. In order to solve this problem, a comprehensive set of criteria that influence foundational ontology selection has been compiled and the values for each parameter determined for DOLCE, BFO, GFO, and SUMO. This paper-based analysis is transformed into an easily extensible algorithm and implemented in the novel tool ONSET, which helps a domain ontology developer to choose a foundational ontology through interactive selection of preferences and scaling of importance so that it computes the most suitable foundational ontology for the domain ontology and explains why this selection was made. This has been evaluated in an experiment with novice modellers, which showed that ONSET greatly assists in foundational ontology selection.

1 Introduction

Ontology development and usage is increasing, which puts higher demands on sound ontology engineering guidelines and practices. One component of ontology development may involve selection and usage of a foundational ontology (FO) that provides high-level categories and generic relationships that are common among ontologies. If used, it increases the domain ontology's quality and interoperability, and it has been shown to speed up ontology development [1,2]. Although there are a variety of FOs available since several years, such as DOLCE [3], BFO, GFO [4], SUMO [5] and YAMATO, most existing methodologies, such as METHONTOLOGY, NeOn, On-To-Knowledge, and the MeltingPoint methodology, do not include their usage explicitly, whereas OntoSpec [6] contains specific guidelines on how to use DOLCE only. Even when a domain ontology developer wants to consider using a FO, there is a prohibitive learning curve due to the considerable quantity of documentation and the new terminology it introduces. Seeing that FOs are beneficial and sometimes necessary in domain ontology development and that there is no methodology or tools which consider FOs in general, there is a need for assistance. We focus on the essential step before including the use of a FO in an extended ontology development methodology: a method for selecting the appropriate FO for the existing or prospective

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 237–251, 2012.

domain ontology and being able to explain why that one is, comparatively, the best choice. To date, there have been a few partial, paper-based, comparisons of the ontologies tailored to an individual project only [7,8,9], but not such that it can be used regardless the scenario and subject domain.

We aim to fill this gap by providing *software-supported selection* of the most suitable FO based on the developer's input, and therewith also generate an *automated explanation* based on the FO's features and the input previously provided by the developer. To realise this, we analysed the characteristics of the DOLCE, BFO, GFO, and SUMO FOs, such as their philosophical distinctions, and availability in certain ontology languages, and on what criteria domain ontology developers have based their decisions in existing projects. A first, extensible, selection and explanation algorithm was developed and implemented in the ONtology Selection and Evaluation Tool ONSET handling DOLCE and BFO only, and its extensibility was evaluated with relative ease of adding the values for the criteria of GFO and SUMO, and corresponding explanations. In addition, to assist the developer in commencing using the FO, ONSET lists relevant references of existing ontologies that also used the proposed FO. This version of ONSET was evaluated with existing ontology development projects and in a controlled experiment with novice ontology developers. Those who used ONSET selected the appropriate FO in most cases and, unlike the null group, they did it quicker and they were able to provide much more reasons for selecting the FO.

Ontology recommender systems are available (e.g., [10,11,12]), but they concern guidance in finding suitable domain ontologies, which is a different scenario for which other questions and requirements hold than for selecting a FO. They share the idea of assisting and informing developers about the criteria and properties associated with ontologies, but ONSET also provides explanation why some FO suits better than another—therewith also providing a basis for argumented comparison of the FOs—, how it relates to the domain ontology to be created, and relevant references of ontologies that also used that FO. ONSET and supplementary material can be accessed from `http://www.meteck.org/files/onset/`.

The remainder of the paper is structured as follows. A brief methodology is described in Section 2, which is followed by the paper-based comparison in Section 3. The design and implementation of ONSET is presented in Section 4 and its evaluation with results and discussion in Section 5. We conclude in Section 6.

2 Methodology

To solve the problems and realise ONSET, we followed a standard software development methodology, and adapted it to the problem at hand:

1. Conduct a literature review on the usage of FOs.
2. Carry out a comparative study of popular FOs.
3. Select suitable FOs to implement in the tool
4. Create an initial list of criteria to assess the ontologies on and, hence, on why one would use either ontology.

5. Contact the creators of the selected ontologies to verify and contribute to the initial criteria list.
6. Devise tool requirements.
7. Based on these criteria, produce an algorithm in order to assist the user in development.
8. Design and implement the algorithm in an easily extendable application.
9. Perform a qualitative evaluation of the tool by FO usage scenarios
10. Test extensibility of the approach by including another FO and analyse changes required, if any.
11. Perform a qualitative evaluation of the tool by FO usage scenarios and a quantitative evaluation with novice ontology developers.

3 Foundational Ontologies and Their Comparison

We conducted a literature review of about 60 papers, covering the publications and documentation by the FO's developers, other comparisons, and case reports on actual usage in domain ontologies, of which we discuss a notable selection.

3.1 Related Works

The WonderWeb deliverable D18 [3] contains extensive information on DOLCE, OCHRE and BFO v1. The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) is based on common-sense principles, heavily informed by Ontology, and axiomatised in FOL. The Basic Formal Ontology (BFO) is a small taxonomy commonly used for scientific research and data integration purposes. The Suggested Upper Merged Ontology (SUMO) [5] is a descriptive ontology of universals and particulars, developed within the context of the IEEE-sanctioned working group and having merged several ontologies. The General Formal Ontology (GFO) [4] is an ontology of universals, concepts and symbols, and is used mainly in the health-care and medical fields.

There are several comparisons of FOs [9,13,14,15], which focus on selected topics, such as technical aspects and representation languages, which can be extended with more recent topics for comparison, such as modularity. However, these comparative studies did not offer a range of categories to compare. Ontological commitments, ontology languages, subject domain examples and other categories were all grouped and compared together to suit the one-off scenario. Furthermore, these comparisons used only a subset of relevant criteria to substantiate usage of a particular FO.

The areas of the domain ontologies using a FO are diverse, yet some differences can be observed. Scientific ontologies such as those used in the biomedical and life science domains mainly use BFO and GFO [16,17] which is partially due to the OBO Foundry [18], which has recommended that ontologies registered on the OBO Foundry use BFO. DOLCE and SUMO have been applied to a variety of subject domains including engineering [14], biomedical [19,20], government and military [15], and landscape [21].

3.2 Categories of Criteria and Comparison

By analysing the comparative studies, documentation of FOs, and their usage in domain ontology projects, an initial list of criteria for selection was created, and grouped into five categories, which are most relevant for the 'information systems perspective' of ontology development (cf. choosing, e.g., which mereology and axioms to incorporate in the ontology [16,22]):

- *Ontological Commitments*: The philosophical choices taken by FOs; e.g., an ontology of particulars vs. universals, multiplicative or not.
- *Representation Language*: The languages used to represent an ontology; e.g., KIF, OBO, or OWL DL.
- *Software engineering properties*: General properties associated with FOs; e.g., licensing and whether the FO is modularised.
- *Subject Domain*: Existing domains represented in a domain ontology using a FO; e.g., the biomedical domain.
- *Applications*: The application scenarios of domain ontologies; e.g., whether the intended purpose of the domain ontology is for the Semantic Web, data integration, NLP.

The final lists of criteria for the ontological commitments and their corresponding values for the four selected FOs is shown in Table 1. Note some idiosyncracies in terminology usage; e.g., an 'ontology of particulars' may well focus on adding classes to the high-level categories provided in the FO and while 'descriptive' and 'realist' is important philosophically, one can include them both in different sections of an ontology (in GFO; or, from a logicians' viewpoint: it all ends up as a logical theory anyway). The final lists of criteria for ontology representation languages for the four selected FOs is as follows. DOLCE is available in FOL, KIF, OWL DL (and therewith also OWL 2 DL), BFO is available in OBO, FOL, KIF, and all OWL species, GFO in OWL DL (and also OWL 2 DL), and SUMO in SUO-KIF and OWL DL. Software engineering properties as per the final criteria lists are compared in Table 2 for their dimensions and modularity; regarding licencing: they are all freely available, and they are all actively being maintained. The complete list of subject domain and application criteria can be accessed at `http://www.meteck.org/files/onset/`.

4 Design of ONSET

A systematic and rigorous approach is employed for the design of the ONtology Selection and Evaluation Tool ONSET in order to ensure proper functioning and useful functionality. The design concerns the tool's requirements, selection algorithm, and implementation aspects.

4.1 Requirements for ONSET

The tool has a number of functional requirements. Primarily, it must select an appropriate FO to be used, and a neat summary of why the particular FO was selected is to be produced as an output (hence, it also must store a user's answers

Table 1. Comparison of ontological commitments

Term (and very brief descriptions of its meaning)	**DOLCE**	**BFO**	**GFO**	**SUMO**
Universals vs. Particulars (Universals can have instances, particulars do not)	Particulars	Universals	Universals, concepts, and symbols	Universals and particulars
Descriptive vs. Realist (Descriptive: represent the entities underlying natural language and human common-sense; Realist: represent the world exactly as is)	Descriptive	Realist	Descriptive and Realist	Descriptive
Multiplicative vs. Reductionist (Mutiplicative: different objects can be co-located at the same time; Reductionist: only one object may be located at the same region at one time)	Multiplicative	Reductionist	Unclear	Multiplicative
Endurantism vs. Perdurantism (Endurantism: an object is wholly present at all times; Perdurantism: an object has temporal parts)	Endurantism and perdurantism	Endurantism and perdurantism	Endurantism and perdurantism	Endurantism and perdurantism
Actualism vs. Possibilism (everything that exists in the ontology is real vs. objects are allowed independent of their actual existence)	Possibilism	Actualism	Unclear	Unclear
Eternalist stance (the past, present and future all exist)	Eternalist stance	Eternalist stance	Eternalist stance	Eternalist stance
Concrete & Abstract entities (Concrete: entities that exist in space and time; Abstract: entities that exist neither in space nor time)	Concrete, abstract	Concrete	Concrete, abstract	Concrete, abstract
Mereology (theory of parts)	GEM	Own mereology	Own mereology	Own mereology
Temporal aspects (e.g., time-indexed axioms)	Provided	Not provided	Provided	Provided
Granularity (different levels of detail contained in an ontology)	High level	Sensitive to granularity	Unclear	Unclear
Properties and values ('attribute'; e.g., the colour of an apple)	Included	Not included	Included	Included
Model for space and time (Consists of time and space regions and boundaries)	Not included	Not included	Included	Not included
One-layered vs. Three-layered architecture (a basic level only; an abstract top level, abstract core level and basic level)	One-layered	One-layered	Three-layered architecture	One-layered
Situations and situoids (Situation: an aggregate of facts that can be comprehended as a whole and satisfies certain conditions of unity; Situoid: is a part of the world that is a comprehensible whole and can exist independently)	Not included	Not included	Included	Not included

Table 2. Comparison of 2 of the 4 software engineering properties

	Dimensions	**Modularity**
DOLCE	100 categories and 100 axioms + relations, quality properties and qualia to represent attributes	Lighter/expressive versions, endurants and perdurants are separate, built-in domain-specific ontologies
BFO	in OWL - 39 universals; in OBO- 23 terms and 33 typedefs; with RO-33 universals and 34 object properties	Endurants and perdurants are separate
GFO	Full- 79 classes, 97 subclass axioms and 67 object properties; Basic- 44 classes, 28 subclass axioms, 41 object properties	Lighter/expressive versions, modules for functions and roles
SUMO	1000 terms, 4000 axioms, 750 rules	Endurants and perdurants separate, built-in domain-specific ontologies

corresponding to each question). If the user has requirements relating to more than one FO, the conflicting results—what is provided by the selected FO compared to what the user wants—is to be compared and displayed. In addition, it has to provide a list of existing ontology references of the domain chosen by the user, if available. It must include 'additional questions', when applicable: these are the questions where all implemented FOs have the same value and thus will not affect the results of ONSET at present, but may in the future, and the user must be given a choice about whether to include or exclude these questions from the program run. Optional scaling, which involves assigning a rating of importance to each category by the user, must be implemented.

In addition, several non-functional requirements are essential. The tool must be designed and implemented such that maintaining and modifying it is a quick and simple process and it must be able to run on different operating systems and platforms. Users must feel comfortable and at ease using the tool. The tool should be divided into windows, tabs and panels to minimize possible cognitive overload and make it easy to understand and follow, and promote usability. The time taken between submitting answers and calculating results must be minimal.

4.2 Algorithm

The general idea of the algorithm is that it takes the largest counter variable to select the FO and it uses arrays of ontological choices to display the motivation for the selected FO and conflicting results. Algorithm 1 decides whether additional questions are to be shown in the interface (lines 1-5), and assigns an optional scaling per category (lines 6-10), both according to the user's input. The algorithm then applies the selected scaling values to each category (line 19). It displays questions and options per category (lines 16,17), and accepts and stores the answers of the user (line 20). Based on this, in Algorithm 2, the selected FO is calculated and displayed, alongside reasons as to why it was chosen (lines 1-5). If present, conflicting results are displayed (lines 8-21). In addition, it provides a list of existing ontology references of the domain chosen by the user, if available

(lines 6,7). The algorithm is easily extensible in the sense that, when a new FO has to be added, the structure and data contained in the algorithm change little: if all the (manually) identified criteria for using such a foundational already exists in ONSET's repository, only references to subject domains using that FO have to be added, and if there is some criterion that does not exist in ONSET yet, a question and options are to be added.

Algorithm 1. ONtology Selection and Evaluation Tool Algorithm 1

```
   DolceCount = 0; BFOCount = 0; GFOCount = 0; SUMOCount = 0;
   DolceAnswers[] = null; BFOAnswers[] = null; GFOAnswers[] = null;
   SUMOAnswers[] = null; DOLCEdomain[] = null; BFOdomain[] = null;
   GFOdomain[] = null; SUMOdomain[] = null; k = 0; ScalingValues[] = null;
   output: Include additional questions?
 1 if input is yes then
 2 |  Show additional questions
 3 else
 4 |  Hide additional questions
 5 end
   output: Assign scaling per category?
 6 if input is yes then
 7 |  for i ← 0 to numOfCategories do
 8 |  |  Read scaling value;
 9 |  |  Store scaling value in ScalingValues[i];
10 |  end
11 else
12 |  ScalingValues[i] = 1;
13 end
14 for i ← 0 to numOfCategories do
15 |  for j ← 0 to numOfQuestionsPerCategory do
16 |  |  Display question;
17 |  |  Display options;
18 |  |  if option corresponds to DOLCE then
19 |  |  |  DOLCECount = DOLCECount+(1* ScalingValues[i]);
20 |  |  |  DOLCEANSWERS[j]= option text;
21 |  |  |  if numOfCategories == 3 then
22 |  |  |  |  for k ← 0 to numberofreferences do
23 |  |  |  |  |  DOLCEdomain[k] = Subject domain reference;
24 |  |  |  |  end
25 |  |  |  end
26 |  |  else if option corresponds to BFO then
27 |  |  |  %% analogous to lines 18-25
28 |  |  else if option corresponds to GFO then
29 |  |  |  %% analogous to lines 18-25
30 |  |  else if option corresponds to SUMO then
31 |  |  |  %% analogous to lines 18-25
32 |  |
33 |  end
34 end
```

Algorithm 2. ONtology Selection and Evaluation Tool Algorithm 2

```
   input : Calculate result
 1 if DOLCECount > (BFOCount AND GFOCount AND SUMOCount) then
       output: Selected FO is FO
       output: Reasons why DOLCE was chosen:
 2     for i ← 0 to DOLCEAnswers.length do
 3         if DOLCEAnswers[i]! = null then
               output: DOLCEAnswers[i]
 4         end
 5     end
       output: Existing ontologies with the specified subject domain
 6     for k ← 0 to DOLCEDomain.length do
           output: DOLCEDomain[k]
 7     end
 8     if BFOAnswers OR GFOAnswers OR SUMO not empty then
           output: Conflicting results: The tool has detected that some of your
                   criteria matches with other foundational ontologies.
 9         for i ← 0 to BFOAnswers.length do
10             if BFOAnswers[i]! = null then
                   output: BFOAnswers[i]
11             end
12         end
13         for i ← 0 to GFOAnswers.length do
14             if GFOAnswers[i]! = null then
                   output: GFOAnswers[i]
15             end
16         end
17         for i ← 0 to SUMOAnswers.length do
18             if SUMOAnswers[i]! = null then
                   output: SUMOAnswers[i]
19             end
20         end
21     end
22 else if BFOCount > (DOLCECount AND GFOCount AND SUMOCount)
   then
23     %% analogous to lines 1-21
24 else if GFOCount > (DOLCECount AND BFOCount AND SUMOCount)
   then
25     %% analogous to lines 1-21
26 else if SUMOCount > (DOLCECount AND BFOCount AND GFOCount)
   then
27     %% analogous to lines 1-21
28 else
       output: :
29     The tool was not able to select a FO due to many a contradictory responses.
       Restart the process if you wish.
30 end
```

4.3 Implementation Specification

ONSET was developed in Java using Netbeans IDE. The .jar file that runs locally on the user's machine requires a minimum of 1MB of free disk space, 1GB of RAM, and JRE components installed.

The functional requirements are met as follows. All five categories with their criteria are converted into questions with pre-defined answers that a user can choose from. A simple if-else statement handles the optional additional questions. For each category, a scaling value between 0-5 is available where 0 represents omit and 5 represents most important, which is applied to each answered question. Answer are stored in an array, to which the calculations are applied, resulting in the ontology with the highest value being selected, and the array of the preferences and values of the selected FO is displayed to the user. Likewise, if the array of the FO that was not selected is not empty, conflicting answers are found, which are displayed alongside what is offered by the selected FO, offering a comparison to the user. Finally, when a user chooses a subject domain, all the available references are added to an array of subject domains and displayed on the results page.

The non-functional requirements are met as follows. Portability is achieved by having a .jar file produced by Netbeans IDE, which is platform-independent, and maintainability is achieved by means of useful comments throughout the code and generating java docs. Response time during usage of ONSET is minimal thanks to the non-frills, uniform, and neat design of the tool. Finally, usability was enhanced by helpful labels and "explain" buttons for explaining complicated terms, which are slightly longer than the explanations in Table 1.

5 Results and Discussion

ONSET was evaluated in three ways: (*i*) testing it against existing and simulated ontologies, (*ii*) a quantitative evaluation with novice ontology developers, and (*iii*) qualitative, expert user feedback from peers.

5.1 Evaluation of ONSET's Functionality

We illustrate ONSET with existing and simulated ontologies; more scenarios can be found at `http://www.meteck.org/files/onset/`.

Scenario 1: Semantic Management of Middleware. The tool was tested according to the requirements of [9] which is an application of the semantic web. Ontological choices of the test case include: descriptiveness, a multiplicative approach, possibilism, perdurantism, modularity (lightweight versions) and an executable language. An example of one of the questions in ONSET which corresponds to this scenario is "Descriptive or Realist Ontology?", where the user has to choose between "Descriptive", "Realist" and "Both". When all the choices are submitted to ONSET, it chooses DOLCE as a FO (see Fig. 1). This corresponds to the foundational ontology used in [9].

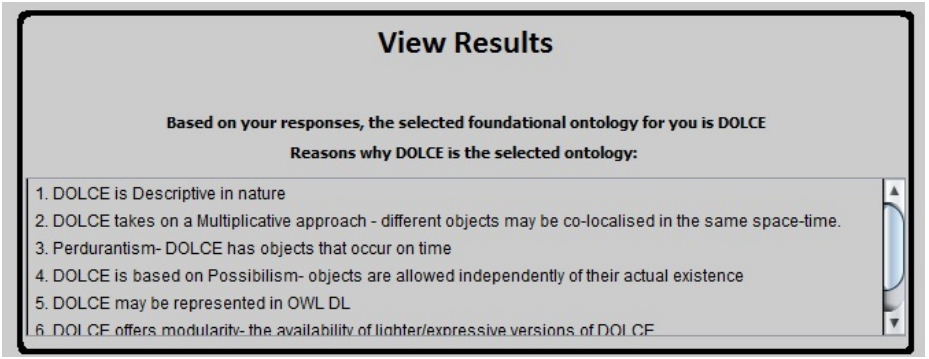


Fig. 1. Output of ONSET: Scenario 1

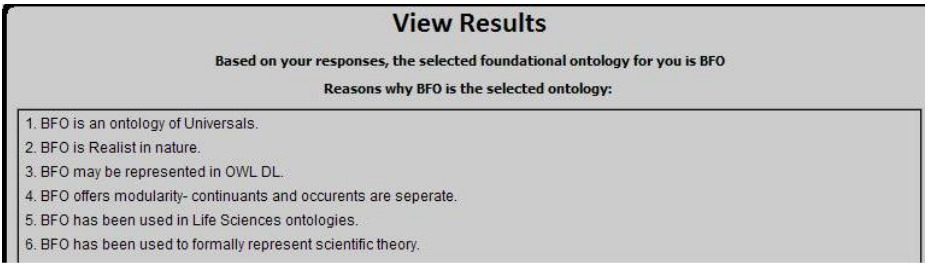


Fig. 2. Output of ONSET: Scenario 2 (without scaling)

Scenario 2: Scaling effects. We show the functioning of scaling in ONSET by simulation of a scenario. Let us assume that there is an ontology to be created with the following requirements: an ontology of universals, realist in nature, to be represented in OWL DL, modularity (endurants and perdurants separate, built-in domain specific ontologies), applying it to formally represent a scientific theory and a domain of life sciences. Without scaling, ONSET chooses BFO as the selected FO as can be seen in Fig. 2. We then use the same input but fill in priorities for the categories in the scaling section in ONSET; for instance, we assign ontological commitments a value of 1, representation languages 5, software engineering properties 3, subject domain 5, and applications a 4. Based on the results calculated, ONSET now chooses DOLCE as the selected FO (Fig. 3); hence, the results of ONSET changed for the same values as before but together with scaling. Observe also the reporting of conflicting answers in Fig. 3.

5.2 Experimental Evaluation

The purpose of the quantitative evaluation is to assess whether using ONSET makes a difference in selecting a FO compared to not using the tool, focusing on timing and correctness and completeness of the selections.

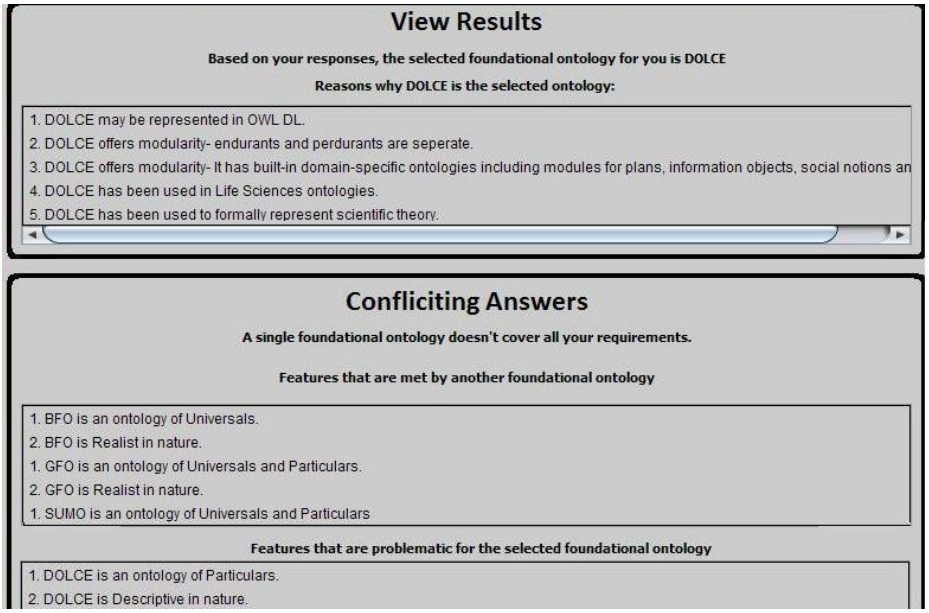


Fig. 3. Output of ONSET: Scenario 2 (with scaling)

Materials and methods. The set-up for the experiment is as follows:

1. Lecture on FOs (1.5h) and announcement of experiment to be held one week after the lecture during the lab time.
2. Divide class into groups A and B randomly (list generated by random.org).
3. Explain purpose of experiment and distribute the assessment to everyone.
4. Provide both the groups with instructions (soft-copy):
   - Group A: complete the given tasks in the prescribed time ($\leq$ 2h) to the best of their ability by using their lecture notes and resources found on the internet.
   - Group B: idem as Group A, and using ONSET.

   The tasks consisted of (*i*) five scenarios for domain ontology development, where for each scenario, a domain ontology had to be chosen and reasons given why, (*ii*) open-ended questions asking their opinion on the experiment, and (*iii*) for Group B only, feedback on the tool.
5. Participants upload their final answers to the course's Moodle, so that timing is captured.
6. Evaluate the tasks, which is performed as follows:
   i. Assessing and comparing the quality of answers of group A and B.
   ii. Comparing the time taken to complete the tasks of each group.
   iii. Collecting and analysing user opinions of all participants.

To measure the quality of the answers given by the participants, we use an accuracy measure: one mark is awarded if the FO corresponds to that of the scenario, and thereafter one mark is awarded for each correct reason provided

for the scenario ('correct' with respect to the scenario). Then, given a maximum amount of criteria for each question, the marks for all the scenarios are summed and converted to a percentage.

Results and discussion. The sample size was 18 honours (4th year) computer science students enrolled in UKZN's "ontologies and knowledge bases" course (comp718), with 9 students in group A and 9 in group B.

Group A (the null group) had submitted their tasks in, on average, 105 minutes and of the $(9 * 5 =)$ 45 scenarios, 8 scenarios were left unanswered. For each scenario, the participants had experienced difficulty in selecting a FO, and could not substantiate their choices. 67% of the participants stated that they had found it difficult to identify which FO would satisfy the criteria provided. Some participants thought that substantiating their choices for FO selection was time-consuming, and had difficulty in understanding all the ontological terms used.

Group B (using ONSET) had submitted their tasks in, on average, 97 minutes with all scenarios answered completely. Thus, the time taken for FO selection is somewhat less in Group B than in Group A. They were able to select a FO to use and provide solid reasons for their choice. The participants of Group B all had a good understanding of the ontological terms used, which, to a large extent, is because ONSET provides "explain" buttons. 33% of the participants of Group B stated that ONSET assisted with complex terms and concepts.

The accuracy rates for each task and of the total tasks are included in Table 3, from which can be seen that for each use-case, Group B was more than twice as accurate as Group A in FO selection for each scenario, and has about 3 times higher accuracy percentage overall. The largest difference in accuracy rates between the two groups is found in scenario 5. Scenario 5 is a complex problem involving the representation of more abstract knowledge. The participants of Group A struggled to identify a FO that would allow this. Group B, on the other hand, found it much easier because by answering with the correct criteria, ONSET selected the appropriate FO, with explanations. Overall, though, the accuracy rates of Group B are not as high as one may hope for, which is mainly because the participants were not able to provide every possible reason for selecting a FO. This is probably because it was their first exercise in working with such scenarios. Notwithstanding, the accuracy rates of Group B are much higher than that of Group A. Thus, overall, Group B performed better in FO selection than Group A and it is apparent that ONSET does provide assistance in FO selection.

Table 3. A comparison of the accuracy of the answers by Group A and Group B

Scenario	**Group A Average**	**Group B Average**
1.Ontology of heart diseases	22%	52%
2.Ontology for the integration of databases of a manufacturing factory	16%	43%
3.Ontology of economic systems	20%	48%
4.Ontology of banks	16%	37%
5.Ontology for conceptual data models	8%	51%
All Scenarios	*16%*	*46%*

Qualitative Feedback. Participants of Group B were impressed with the output of ONSET, and thought that it generates results effectively, provided that you input sufficient information and ontological choices about the proposed ontology (but recollect from Section 4.3 that one does not have to answer all questions to obtain a selection). All participants from Group B who had provided feedback on ONSET felt that they would not have been able to perform the task easily without ONSET and they agreed that the user-interface and navigation through the program was quick and simple.

Further, the alpha version of ONSET was presented informally at MAIS'11, which resulted in positive feedback, including usage of the tool also in an ontology engineering course at the University of South Africa (UNISA). The suggestion to implement a tooltip to explain relatively complicated ontological terms was implemented with the more comprehensive "explain" buttons. Positive feedback was received also from the DOLCE, BFO, and GFO ontology developers, in particular regarding the enhanced possibility for comparison with other ontologies and how a scenario has an effect on the selection, which opens avenues for further investigation.

5.3 Discussion

As the evaluation of ONSET demonstrates, it effectively lets the user select a FO and provides an explanation why. It being software-based, this makes it easy to run alternative scenarios and obtain selections, compared to the pre-existing situation with manual assessments of paper-based comparisons and where the developer had to read through all documentation before being able to make an informed selection. The method proposed here can be used at the start of ontology development, during improvement of an existing ontology with a FO, and for ontology interoperability scenarios.

It raises also some new questions and sheds light on existing ones. First, if the "FO library" envisioned in 2003 by [3] would have existed, selection of a FO would have been less of an issue, for their formalisations would have been aligned to the extent possible. Such a hypothetical library, however, would still need some management of, among others, modularity, availability of the ontology in a preferred, or deemed necessary, ontology language, and ontological choices. Hence, a library with mappings between concepts/universals and relationships/properties of the ontologies alone would not satisfy the ontology developers' needs. Second, the whole notion of 'foundational' is based on the assumption that there is one and that that ontology fits for all domain ontology development projects. Yet, when one runs different scenarios, conflicting answers may arise to the extent that there may well be no 'best fit', i.e., where more than one FO fits equally well (or badly) given the user input provided, or at least having to deal with minor conflicting answers (recollect Fig. 3), or another is proposed due to the given scaling of the categories. It is exactly here that the explanations become crucial: they are fact-based arguments for and against a particular FO for that scenario, and therewith compose a start for a scientific analysis of FO choice in domain ontology development projects.

While we evaluated several existing domain ontology projects, this is biassed toward the criteria described informally in the corresponding paper, hence geared toward confirmation of correct implementation of the ONSET algorithm; the selection may or may not be the same once the developers are offered the additional criteria available in ONSET. Furthermore, it would be interesting to know whether some category of criteria, or individual criteria, are always deemed more important than others, whether there exists one or more 'typical' combinations of criteria, and the incidence of conflicts and if so, which criteria they typically involve. ONSET clearly can be a useful aid investigating these questions, but answering them is left to future works.

6 Conclusions

The problem that ontology developers have severe difficulties in selecting which foundational ontology to use for domain ontology development and why, has been successfully solved with the ONSET tool. ONSET assists and informs developers about the criteria and properties associated with foundational ontologies and how they relate to the domain ontology to be created. It calculates a preferred foundational ontology based on the user-provided requirements and the values of the criteria for each foundational ontology. The compiled lists of criteria and implementation is, to the best of our knowledge, the first paper-based as well as software-assisted and subject-domain independent approach in foundational ontology selection. Effectiveness of ONSET was experimentally evaluated and shown to substantially improve selection and the user's capability to motivate why.

Future works pertain to extending functionalities of ONSET, such as allowing users to map their existing ontologies to a foundational ontology, and integrating foundational ontology selection and usage in existing ontology methodologies.

Acknowledgements. We wish to thank Pierre Grenon, Stefano Borgo, Claudio Masolo, Frank Loebe and Heinrich Herre for their feedback.

References

1. Keet, C.M.: The Use of Foundational Ontologies in Ontology Development: An Empirical Assessment. 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. 321–335. Springer, Heidelberg (2011)
2. Borgo, S., Lesmo, L.: The attractiveness of foundational ontologies in industry. In: Proc. of FOMI 2008, pp. 1–9. IOS Press, Amsterdam (2008)
3. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: Ontology library. WonderWeb Deliverable D18 (2003) (ver. 1.0, December 31, 2003), http://wonderweb.semanticweb.org
4. Herre, H.: General Formal Ontology (GFO): A foundational ontology for conceptual modelling. In: Poli, R., Healy, M., Kameas, A. (eds.) Theory and Applications of Ontology: Computer Applications, pp. 297–345. Springer, Heidelberg (2010)

5. Niles, I., Pease, A.: Towards a standard upper ontology. In: Proc. of FOIS 2001, October 17-19. IOS Press, Ogunquit (2001)
6. Kassel, G.: Integration of the DOLCE top-level ontology into the OntoSpec methodology. Technical Report HAL: hal-00012203/arXiv: cs.AI/0510050, Laboratoire de Recherche en Informatique d'Amiens (LaRIA) (October 2005), http://hal.archives-ouvertes.fr/ccsd-00012203
7. Grenon, P., Smith, B., Goldberg, L.: Biodynamic ontology: Applying BFO in the biomedical domain. In: Stud. Health Technol. Inform., pp. 20–38. IOS Press (2004)
8. Ermolayev, V., Keberle, N., Matzke, W.-E.: An Upper Level Ontological Model for Engineering Design Performance Domain. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 98–113. Springer, Heidelberg (2008)
9. Oberle, D.: Semantic Management of Middleware. Semantic Web and Beyond, vol. 1. Springer, New York (2006)
10. Jonquet, C., Musen, M.A., Shah, N.H.: Building a biomedical ontology recommender web service. Journal of Biomedical Semantics 1(suppl. 1), S1 (2010)
11. Martínez-Romero, M., Vázquez-Naya, J.M., Pereira, J., Pazos, A.: A Multi-criteria Approach for Automatic Ontology Recommendation Using Collective Knowledge. In: Pazos Arias, J.J., Fernández Vilas, A., Díaz Redondo, R.P. (eds.) Recommender Systems for the Social Web. ISRL, vol. 32, pp. 89–104. Springer, Heidelberg (2012)
12. Wang, X., Guo, L., Fang, J.: Automated ontology selection based on description logic. In: Proc. of CSCWD 2008, April 16-18, pp. 482–487. IEEE Computer Society, Xi'an (2008)
13. Mascardi, V., Cordí, V., Rosso, P.: A comparison of upper ontologies. Technical Report DISI-TR-06-21, University of Genova, Italy, pp. 55–64 (2007)
14. Jureta, I.J., Mylopoulos, J., Faulkner, S.: A core ontology for requirements. Applied Ontology 4(3-4), 169–244 (2009)
15. Semy, S.K., Pulvermacher, M.K., Obrst, L.J.: Toward the use of an upper ontology for us government and us military domains: An evaluation. Technical Report MTR 04B0000063. The MITRE Corporation (2004)
16. Keet, C.M.: Transforming semi-structured life science diagrams into meaningful domain ontologies with DiDOn. J. of Biomedical Informatics 45(3), 482–494 (2012)
17. Hoehndorf, R., Prüfer, K., Backhaus, M., Visagie, J., Kelso, J.: The design of a wiki-based curation system for the ontology of functions. In: Proc. of the Joint BioLINK and 9th Bio-Ontologies Meeting, Fortaleza, Brazil, August 5 (2006)
18. Smith, B., et al.: The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology 25(11), 1251–1255 (2007)
19. Ahmad, F., Lindgren, H.: Selection of Foundational Ontology for Collaborative Knowledge Modeling in Healthcare Domain. In: Dicheva, D., Dochev, D. (eds.) AIMSA 2010. LNCS, vol. 6304, pp. 261–262. Springer, Heidelberg (2010)
20. Keet, C.M.: Factors Affecting Ontology Development in Ecology. In: Ludäscher, B., Raschid, L. (eds.) DILS 2005. LNCS (LNBI), vol. 3615, pp. 46–62. Springer, Heidelberg (2005)
21. Sinha, G., Mark, D.: Toward a foundational ontology of the landscape. In: Proc. of GIScience 2010, Zürich, Switzerland, September 14-17 (2010)
22. Fernández-López, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: Selecting and customizing a mereology ontology for its reuse in a pharmaceutical product ontology. In: Eschenbach, C., Grüninger, M. (eds.) Proc. of FOIS 2008, pp. 181–194. IOS Press (2008)

Detecting and Revising Flaws in OWL Object Property Expressions

C. Maria Keet

School of Mathematics, Statistics, and Computer Science,
University of KwaZulu-Natal, and UKZN/CSIR-Meraka Centre for Artificial Intelligence Research, South Africa
keet@ukzn.ac.za

Abstract. OWL 2 DL is a very expressive language and has many features for declaring complex object property expressions. Standard reasoning services for OWL ontologies assume the axioms in the 'object property box' to be correct and according to the ontologist's intention. However, the more one can do, the higher the chance modelling flaws are introduced; hence, an unexpected or undesired classification or inconsistency may actually be due to a mistake in the object property box, not the class axioms. We identify the types of flaws that can occur in the object property box and propose corresponding compatibility services, *SubProS* and *ProChainS*, that check for meaningful property hierarchies and property chaining and propose how to revise a flaw. *SubProS* and *ProChainS* were evaluated with several ontologies, demonstrating they indeed do serve to isolate flaws and can propose useful corrections.

1 Introduction

There is considerable ongoing ontology development effort in various subject domains, such as the life sciences, medicine, e-learning, and the enterprise. New ontologies tend to be developed with the most recent W3C-standardised ontology language, OWL 2 [1]. OWL 2 DL is based on the Description Logics (DL) language $\mathcal{SROIQ}$, which, thanks to ontology engineers' requests for more features for object properties, now allows for object sub-properties, (inverse) functional, disjointness, equivalence, cardinality, (ir)reflexivity, (a)symmetry, transitivity, and role chaining. There are a few syntactic constraints on their usage, but still a lot is possible to declare, which also means there is now even more room to make mistakes with respect to the ontologist's intention in addition to those noted for modelling with OWL 1 [2–4]. For instance,

(i) Domain and range flaws; e.g. (simplified), hasParent $\sqsubseteq$ hasMother instead of hasMother $\sqsubseteq$ hasParent in accordance with their domain and range restrictions, or declaring a domain/range to be an intersection of disjoint classes;
(ii) Property characteristics flaws: e.g., the `family-tree`[1] has hasGrandFather $\sqsubseteq$ hasAncestor and Trans(hasAncestor) so that transitivity unintentionally is passed down the property hierarchy (hasGrandFather is intransitive, which cannot be asserted in OWL);

[1] `http://www.co-ode.org/roberts/family-tree.owl`; last accessed 12-3-2012.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 252–266, 2012.

(iii) Property chain issues; e.g., $\mathsf{hasPart} \circ \mathsf{hasParticipant} \sqsubseteq \mathsf{hasParticipant}$ in the pharmacogenomics ontology [5] that forces the classes in class expressions using these properties (DrugTreatment and DrugGeneInteraction) to be either processes due to the domain of hasParticipant, or they will be inconsistent.

Such flaws and unexpected or undesirable deductions are not properly recognised and implemented in explanation features by the extant reasoners and ontology development environments and therewith do not point to the actual flaw in the object property box. This is primarily because implemented justification and explanation algorithms, such as [6, 2, 7], focus only on logical deductions and errors, and take for granted that class axioms and assertions about instances have to take precedence over what 'ought to be' regarding the object property axioms. Therefore, with the standard reasoning, the object property expressions (inclusion axioms)—i.e., property hierarchy, domain and range axioms, a property's characteristics, and property chains—are assumed to be correct, but instances and classes can move about in the taxonomy. However, the modeller may well be certain where in the taxonomy a particular class belongs, or at least its main category, but not so sure about how to represent its properties. This is a reasonable assumption, given that many ontologies commence with just a bare taxonomy and only gradually add properties, and the advanced OWL 2 DL features for object properties are still relatively new. Therewith it has become an imperative to look at how one can get the modeller to choose the ontologically correct options in the object property box so as to achieve a better quality ontology and, in case of flaws, how to guide the modeller to the root defect from the modeller's viewpoint, and propose corrections. Overall, this requires one to be able to recognise the flaw, to explain it, and to suggest revisions.

We address these issues by introducing two non-standard reasoning services. First, we extend the RBox Compatibility Service for object subproperties from [8] to also handle the object property characteristics, called Sub-Property compatibility Service (*SubProS*), and, second, we define a new ontological reasoning service, Property Chain compatibility Service, (*ProChainS*), that checks whether the chain's properties are compatible. The compatibility services are defined in such a way as to exhaustively check all permutations and therewith pinpoint to the root cause in the object property box, where applicable. If a test of either service fails, proposals are made to revise the identified flaw. As such, *SubProS* and *ProChainS* can be considered *ontological reasoning services*, because the ontology does not necessarily contain logical errors in some of the flaws detected. The solution thus falls in the category of tools focussing on both logic and additional ontology quality criteria, alike the works on anti-patterns [4] and OntoClean [9], by aiming toward ontological correctness in addition to just a satisfiable logical theory. Hence, it is different from other works on explanation and pinpointing mistakes that concern logical consequences only [6, 2, 7], and *SubProS* and *ProChainS* also propose revisions for the flaws.

In the remainder of the paper, we address property subsumption and *SubProS* in Section 2, and property chaining with *ProChainS* in Section 3. We conclude in Section 4.

2 Sub-Properties in OWL

2.1 Preliminaries

Subproperties in OWL have a "basic form" and a "more complex form". The former is denoted in OWL 2 functional syntax as `SubObjectPropertyOf(OPE1 OPE2)`, which says that object property expression $\texttt{OPE}_1$ is a subproperty of $\texttt{OPE}_2$, meaning that "if an individual x is connected by OPE_1 to an individual y, then x is also connected by OPE_2 to y" [1]. The simple version is denoted in Description Logics (DL) as $S \sqsubseteq R$ and a typical use case is properPartOf $\sqsubseteq$ partOf. The more complex form concerns property chains, denoted with `SubObjectPropertyOf(ObjectPropertyChain(OPE1 ... OPEn) OPE)` in OWL 2 and several additional syntactic constraints hold. It is called "complex role inclusions" in DL, and is defined succinctly in [10]. More generally, (sub-)properties are constrained by the Role Inclusion Axioms as defined in [10] for $\mathcal{SROIQ}$, the base language for OWL 2 DL, which also provide the constraints for property chains in OWL 2, and is included below as Definition 1. Informally, case 1 covers transitivity of R, case 2 inverses, case 3 chaining of simple object properties provided the regularity constraint hold (a strict order) so as to maintain decidability, and for case 4 and 5, the property on the right-hand side either occurs first or last in the chain on the left-hand side of the inclusion axiom.

Definition 1 ((Regular) Role Inclusion Axioms ([10])). *Let $\prec$ be a regular order on roles. A **role inclusion axiom** (RIA for short) is an expression of the form $w \sqsubseteq R$, where w is a finite string of roles not including the universal role U, and $R \neq U$ is a role name. A **role hierarchy** $\mathcal{R}_h$ is a finite set of RIAs. An interpretation $\mathcal{I}$ **satisfies** a role inclusion axiom $w \sqsubseteq R$, written $\mathcal{I} \models w \sqsubseteq R$, if $w^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. An interpretation is a **model** of a role hierarchy $\mathcal{R}_h$ if it satisfies all RIAs in $\mathcal{R}_h$, written $\mathcal{I} \models \mathcal{R}_h$. A RIA $w \sqsubseteq R$ is $\prec$**-regular** if R is a role name, and*

1. $w = R \circ R$, *or*
2. $w = R^-$, *or*
3. $w = S_1 \circ \ldots \circ S_n$ *and* $S_i \prec R$, *for all* $1 \geq i \geq n$, *or*
4. $w = R \circ S_1 \circ \ldots \circ S_n$ *and* $S_i \prec R$, *for all* $1 \geq i \geq n$, *or*
5. $w = S_1 \circ \ldots \circ S_n \circ R$ *and* $S_i \prec R$, *for all* $1 \geq i \geq n$.

*Finally, a role hierarchy $\mathcal{R}_h$ is **regular** if there exists a regular order $\prec$ such that each RIA in $\mathcal{R}_h$ is $\prec$-regular.*

For reasons of conciseness and readability, we shall use this notation and "$\circ$" for chaining rather than the wordy OWL functional style syntax.

In the remainder of this section, we look into the "basic form" for subproperties, i.e., $S \sqsubseteq R$, and consider property chains (cases 3-5 of Def. 1) in Section 3. To increase readability, we shall use $R \sqsubseteq C_1 \times C_2$ as shortcut for domain and range axioms $\exists R \sqsubseteq C_1$ and $\exists R^- \sqsubseteq C_2$ where C_1 and C_2 are generic classes—`ObjectPropertyDomain(OPE CE)` and `ObjectPropertyRange(OPE CE)` in OWL, respectively. $R \sqsubseteq \top \times \top$ holds when no explicit domain and range axiom has been declared.

2.2 When a Property Is a Subproperty of Another

With class subsumption, the subclass is more constrained than its superclass, which can be applied to properties as well: subsumption for OWL object properties (DL roles) holds if the subsumed property is more constrained such that in every model, the set of individual property assertions is a subset of those of its parent property, or: given $S \sqsubseteq R$, then all individuals in the property assertions involving property S must also be related to each other through property R. There are two ways to constrain a property, where either one suffices: by specifying its domain or range, and by declaring the property's characteristics.

Subsumption due to domain and range axioms. Given a subproperty expression $S \sqsubseteq R$, then in order to have the instances of S to be always a subset of the instances of R, S's domain or range, or both, have to be subclasses of the domain and range of R. Although this might be perceived to have an object-oriented and conceptual data modelling flavour, it is widely used (see also Table 1) and declaring domain and range axioms has certain advantages in automated reasoning as as well constraining the admissible models and therewith being more precise in representing the knowledge. Let us first introduce the notion of *user-defined domain and range classes* for OWL ontologies:

Definition 2 (User-defined Domain and Range Classes). *Let R be an OWL object property and $R \sqsubseteq C_1 \times C_2$ its associated domain and range axiom. Then, with the symbol D_R we indicate the* User-defined Domain *of R—i.e., $D_R = C_1$—and with the symbol R_R we indicate the* User-defined Range *of R—i.e., $R_R = C_2$.*

Thus, we need to specify that it ought to be the case that, given an axiom $S \sqsubseteq R$, $D_S \sqsubseteq D_R$ and $R_S \sqsubseteq R_R$ holds, and propose to the ontologist ways to revise the flaw if this is not the case, as it may lead to undesired, or at least 'unexpected', behaviour of the reasoner, being that either the domain class D_S is classified elsewhere in the hierarchy as a subclass of D_R, or, if the classes were declared disjoint, then D_S becomes inconsistent. This was addressed in detail in [8] and solved with the *RBox Compatibility* service, for which the DL $\mathcal{ALCI}$ sufficed to define it. We shall adapt it to OWL 2 DL and extend that service and options to correct it in the next section.

Subsumption due to the property's characteristics. Relational properties—OWL object property characteristics—constrain the way objects relate to each other; e.g., if an ABox contains $\mathsf{connectedTo}(\mathsf{a}, \mathsf{b})$, then only if $\mathsf{connectedTo}$ is (correctly) asserted to be symmetric then it will infer $\mathsf{connectedTo}(\mathsf{b}, \mathsf{a})$. One can argue for a property hierarchy with respect to the characteristics of the properties; e.g., a property is asymmetric if it is both antisymmetric and irreflexive, hence, if a property is asymmetric, it certainly is also irreflexive but not vv. Again, a subproperty has be more constrained than its parent property and, as with inheritance of properties for classes, the property's characteristic(s) should

be inherited along the property hierarchy or overridden with a stronger constraint. With this line of reasoning and $S \sqsubseteq R$, then, e.g., $\mathsf{Asym}(S)$ and $\mathsf{Irr}(R)$ is admissible, but the other way around, $\mathsf{Irr}(S)$ and $\mathsf{Asym}(R)$ is a flaw where either both are, or neither one is, asymmetric or $R \sqsubseteq S$ would be the intended axiom. One can observe this also for the commonly used parthood and proper parthood relations: the former is reflexive, antisymmetric, and transitive, and the latter irreflexive, asymmetric, and transitive. Conversely, direct parthood (dpo) is intransitive, yet tends to be represented in an OWL ontology as a subproperty of parthood (po). However, one cannot represent intransitivity in OWL explicitly (not asserting transitivity means a property is non-transitive, not intransitive).

With respect to OWL 2 DL, the constraints become fairly straight-forward with respect to a property hierarchy, because antisymmetry, acyclicity (which is more constrained than asymmetric), and intransitivity cannot be represented. However, current OWL reasoners do not take into account inheritance of property characteristics even of the same relational property, except transitivity. That is, if, e.g., $\mathsf{Sym}(R)$ then this does not hold automatically for S, which is easy to check with any ontology editor: add $S \sqsubseteq R$, $\mathsf{Sym}(R)$, $C \sqsubseteq \exists R.D$, $E \sqsubseteq \exists S.F$, $r(c_1, d_1)$ and $s(e_1, f_1)$ and observe the inferences. However, in such a setting, S is non-symmetric, which is generally unintended, especially if there is some T where $T \sqsubseteq S \sqsubseteq R$ and $\mathsf{Sym}(T)$ and $\mathsf{Sym}(R)$. Note that adding $\mathsf{Asym}(S)$ is very well possible. This holds for Ref and Irr, too.

2.3 The Sub-Property Compatibility Service

We now can define the new reasoning service, *Sub-Property compatibility Service* (*SubProS*) by extending the basic notions from the *RBox compatibility* [8]. Informally, it first checks the 'compatibility' of domain and range axioms with respect to the object property hierarchy and the class hierarchy in the ontology. The RBox compatibility service is already necessary and sufficient for finding domain/range problems, because it exhaustively checks each permutation of domain and range of the parent and child property in the object property hierarchy. After that, *SubProS* checks whether the object property characteristic(s) conform to specification, provided there is such an expression in the ontology.

Definition 3 (Sub-Property compatibility Service (*SubProS*)). *For each pair of object properties, $R, S \in \mathcal{O}$ such that $\mathcal{O} \models S \sqsubseteq R$, and $\mathcal{O}$ an OWL ontology adhering to the syntax and semantics as specified in [1], check whether:*

Test 1. $\mathcal{O} \models D_S \sqsubseteq D_R$ *and* $\mathcal{O} \models R_S \sqsubseteq R_R$*;*
Test 2. $\mathcal{O} \not\models D_R \sqsubseteq D_S$*;*
Test 3. $\mathcal{O} \not\models R_R \sqsubseteq R_S$*;*
Test 4. *If* $\mathcal{O} \models \mathsf{Asym}(R)$ *then* $\mathcal{O} \models \mathsf{Asym}(S)$*;*
Test 5. *If* $\mathcal{O} \models \mathsf{Sym}(R)$ *then* $\mathcal{O} \models \mathsf{Sym}(S)$ *or* $\mathcal{O} \models \mathsf{Asym}(S)$*;*
Test 6. *If* $\mathcal{O} \models \mathsf{Trans}(R)$ *then* $\mathcal{O} \models \mathsf{Trans}(S)$*;*
Test 7. *If* $\mathcal{O} \models \mathsf{Ref}(R)$ *then* $\mathcal{O} \models \mathsf{Ref}(S)$ *or* $\mathcal{O} \models \mathsf{Irr}(S)$*;*
Test 8. *If* $\mathcal{O} \models \mathsf{Irr}(R)$ *then* $\mathcal{O} \models \mathsf{Irr}(S)$ *or* $\mathcal{O} \models \mathsf{Asym}(S)$*;*

Test 9. *If* $\mathcal{O} \models \mathsf{Asym}(R)$ *then* $\mathcal{O} \not\models \mathsf{Sym}(S)$*;*

Test 10. *If* $\mathcal{O} \models \mathsf{Irr}(R)$ *then* $\mathcal{O} \not\models \mathsf{Ref}(S)$*;*

Test 11. *If* $\mathcal{O} \models \mathsf{Trans}(R)$ *then* $\mathcal{O} \not\models \mathsf{Irr}(R)$*,* $\mathcal{O} \not\models \mathsf{Asym}(R)$*,* $\mathcal{O} \not\models \mathsf{Irr}(S)$*, and* $\mathcal{O} \not\models \mathsf{Asym}(S)$*;*

An OWL object property hierarchy is said to be compatible iff
- `Test 1` *and (*`2` *or* `3`*) hold for all pairs of property-subproperty in* $\mathcal{O}$*, and*
- `Tests 4-11` *hold for all pairs of property-subproperty in* $\mathcal{O}$*.*

An OWL ontology $\mathcal{O}$ that does not adhere to *SubProS* is considered to be *ontologically flawed* and is in need of repair. To arrive at the repair, we can avail in part of the extant OWL reasoners. The class subsumption checking for `Tests 1-3` can be done with any of the OWL reasoners, where the result is processed to check the conditions in the tests. `Test 9` and `Test 10` are already done by OWL 2 DL reasoners, but it now only returns a class inconsistency when S is used in a class expression, not regarding the property characteristics per sé; hence, the explanation and suggestions for revision has to be amended: with respect to the intended meaning, not the class expression is at fault, but there is a flaw in the property hierarchy. `Test 11` is included merely for purpose of exhaustiveness, as it is already prohibited by the syntax restrictions of OWL 2 and already has a corresponding warning (irreflexivity and asymmetry require simple object properties, but a transitive property is not simple), but it may become relevant for any future OWL version or modification of *SubProS* for another ontology language. By the same token, one could argue to remove `Test 6`, for it is already computed, but it makes it easier for any extensions of the service or its adoption for another ontology language.

The following sequence specifies what has to be done if any of the applicable tests fails. For `Tests 1-3`, we can reuse the basic 2-step idea from [8] and adapt it to the current setting, and we propose new corrections for `Tests 4-11`. Observe that "raising a warning" denotes that it is not a logical error but an ontological one, "forcing" a revision indicates there is a logical error that must be fixed in order to have a consistent ontology with satisfiable classes, and "propose" indicates suggestions how the flaw can be best revised.

A. If `Test 1` fails, raise a warning "domain and range restrictions of either R or S are in conflict with the property hierarchy", and propose to
   - ⋆ Change the object property hierarchy, i.e., either remove $S \sqsubseteq R$ and add $R \sqsubseteq S$ or add $S \equiv R$ to $\mathcal{O}$, or
   - ⋆ Change domain and range restrictions of R and/or S, or
   - ⋆ If the test on the domains fails, then propose a new axiom $R \sqsubseteq D'_R \times R_R$, where $D'_R \equiv D_R \sqcap D_S$ (and similarly when `Test 1` fails on the range).

B. If `Test 2` and `Test 3` fail, raise a warning "R cannot be a proper subproperty of S, but they can be equivalent", and propose:
   - ⋆ Accept the possible equivalence and, optionally, add $S \equiv R$ to $\mathcal{O}$, or
   - ⋆ Change domain and range restrictions of R and/or S.

C. Run *SubProS* again if any changes have been made in steps A or B, and record changes in the hierarchy (to be used in step I).

D. If $\mathsf{Asym}(R)$ is asserted in $\mathcal{O}$ and `Test 4` fails to detect $\mathsf{Asym}(S)$, raise a warning "R is asymmetric, but its subproperty S is not", proposing to remedy this with either:
   - ⋆ Add $\mathsf{Asym}(S)$ to obtain expected inferences;
   - ⋆ Remove $\mathsf{Asym}(R)$;
   - ⋆ Change the positions of R and/or S in the object property hierarchy;

   and similarly when `Test 6` fails,

E. If $\mathsf{Sym}(R)$ is asserted and `Test 5` fails to detect either $\mathsf{Sym}(S)$ or $\mathsf{Asym}(S)$, raise a warning "R is symmetric, but its subproperty S is not, nor is it asymmetric", proposing to remedy this with either:
   - ⋆ Add $\mathsf{Sym}(S)$ or $\mathsf{Asym}(S)$ to obtain expected inferences;
   - ⋆ Remove $\mathsf{Sym}(R)$;
   - ⋆ Change the positions of R and/or S in the object property hierarchy;

   and similarly when `Test 7` fails,

F. If $\mathsf{Irr}(R)$ and `Test 8` fails to detect either $\mathsf{Irr}(S)$ or $\mathsf{Asym}(S)$, raise a warning "R is irreflexive, hence S should be either $\mathsf{Irr}(S)$ or $\mathsf{Asym}(S)$, and propose:
   - ⋆ Add $\mathsf{Asym}(S)$ or $\mathsf{Irr}(S)$ to obtain expected inferences;
   - ⋆ Remove $\mathsf{Irr}(R)$;
   - ⋆ Change the positions of R and/or S in the object property hierarchy;

G. If `Test 9` fails, report "R is asymmetric so its subproperty, S, cannot be symmetric" and force the modeller to change it by either
   - ⋆ Remove $\mathsf{Asym}(R)$, or
   - ⋆ Remove $\mathsf{Sym}(S)$.
   - ⋆ Change the positions of R and/or S in the object property hierarchy;

   and similarly if `Test 10` fails, but then irreflexive and reflexive, respectively.

H. If `Test 11` fails, report "R (and by `Test 6`, S, too) is declared transitive, hence, not a simple object property, hence it is not permitted to participate in an irreflexive or asymmetric object property expression" and force the modeller to change it by either:
   - ⋆ Remove $\mathsf{Trans}(R)$, or
   - ⋆ Remove $\mathsf{Irr}(R)$, $\mathsf{Asym}(R)$, $\mathsf{Irr}(S)$, and $\mathsf{Asym}(S)$;

I. Run *SubProS* again if any changes have been made in steps D-H, and check any changes in the property hierarchy made against those recorded in step C. If a change from steps E or F reverts a recorded change, then report "unresolvable conflict on subproperty axiom. You *must* change at least one axiom to exit an otherwise infinite loop of swapping two expressions".

The reason for running *SubProS* again after `Test 1-3` and not only at the end is that those changes, if any, will affect the outcome of `Tests 4-11`, and in Step I to both push through the changes and prevent an infinite loop. *SubProS* was evaluated with several ontologies, one of which illustrated now.

Evaluation 1 (BioTop's inconsistent 'has process role') The TONES Ontology repository `http://owl.cs.manchester.ac.uk/repository/` contains 219 ontologies, of which we selected 10 ontologies semi-randomly based on having listed in the metrics to have a substantial amount of object properties, and being a real ontology (more precisely: no toy or tutorial ontology, not converted from OBO,

nor an OWLized thesaurus). Some relevant data is shown in Table 1. Interesting to observe is the plethora of object properties that are mostly in an hierarchy and predominantly based on changes in the domain or range. Due to space limitations, we shall analyse here only BioTop's inconsistent object property.

The 'has process role' in BioTop [11] version d.d. June 17, 2010 (last update) is inconsistent, yet there is no explanation for it computed by currently implemented explanation algorithms. We have in the ontology, among other things: 'has process role'$\sqsubseteq$'temporally related to', 'has process role'$\sqsubseteq$'processual entity'$\times$role, 'temporally related to' $\sqsubseteq$ 'processual entity'$\sqcup$quality $\times$ 'processual entity'$\sqcup$quality, role $\sqsubseteq \neg$quality, role $\sqsubseteq \neg$'processual entity', Sym('temporally related to').

Let us use *SubProS* to isolate the error and propose a revision.

- `Test 1` fails, because $\mathcal{O} \not\models R_{\texttt{hasprocessrole}} \sqsubseteq R_{\texttt{temporallyrelatedto}'}$, as the ranges are disjoint;
- `Test 2` and `3` pass.
- `Test 4` is not applicable.
- `Test 5` fails, because $\mathcal{O}$ does not contain Sym('has process role').
- `Test 6-11` are not applicable.

To remedy the issue with `Test 1`, we have three options (see item `A`, above). Thus far, 'has process role' has not been used in class expressions, and from the intention described in the annotation, it suits as subproperty of 'temporally related to', therefore, one opts for choice 2, being to change the range of 'temporally related to' into 'processual entity' $\sqcup$ quality $\sqcup$ role. (The same holds for the inverse 'process role of'.). Second, with `Test 5` failing, we have three options to revise the flaw (see item `E`, above). We already decided not to change the position of the property in the hierarchy, so we either add Sym('has process role') or Asym('has process role') or remove Sym('temporally related to'). Ontologically, Sym('temporally related to') is certainly true and Sym('has process role') certainly false, hence Asym('has process role') is the appropriate choice. After making these changes, we run *SubProS* again, and no issues emerge. ♢

3 Property Chaining in OWL

Property chaining is well-known in DL research as role composition [12–14], but in OWL its usage is more restricted and, except for a few elaborate cases (the family relations example [1] and metamodelling rewritings [15]), is typically used with only two object properties being chained (e.g., [5, 16–18]). The issues mentioned and solution proposed here applies to any OWL 2 DL admissible chain. Let us introduce an example of an undesirable deduction first.

Example 1. (DMOP ontology) The Data Mining and OPtimisation (DMOP) ontology of the e-LICO project (`http://www.e-lico.eu`) v5.2 (Sept. 2011) includes 11 property chains, including hasMainTable $\circ$ hasFeature $\sqsubseteq$ hasFeature, where the properties have domain and range axioms hasMainTable $\sqsubseteq$ DataSet$\times$ DataTable and hasFeature $\sqsubseteq$ DataTable $\times$ Feature, which is depicted in Fig. 1.

Table 1. Selection of some TONES Repository ontologies, retrieved on 12-3-2012 (see *Evaluation 1* for details); OP = object property; char. = object property characteristic

Ontology	**No. of OPs**	**No. of SubOPs axioms**	**No. more constrained by D or R**	**by char.**	**Comments** (partial)
DOLCE-lite	70	46	13	3	transitivity added
SAO 1.2	36	25	21	5	2 x transitivity; `Test 6` fails on has Vesicle Component
airsystem	111	56	43	2	imports DUL. *ProChainS* fails
process (SWEET)	102	10	7	0	
family-tree	52	25	14	2	fails `Test 6` of *SubProS*
propreo	32	20	17	2	beyond OWL 2 DL (non-simple prop. in max card.)
heart	29	18	9	0	many inconsistencies
mygrid-unclassified	69	39	0	3	1 x transitive added
building Architecture	28	24	0	0	imports rcc, fails `Test 5` of *SubProS* (omission Asym on properPartOf)
biotop	89	84	45	9	with transitivity; 'has process role' is inconsistent, see *Evaluation 1*

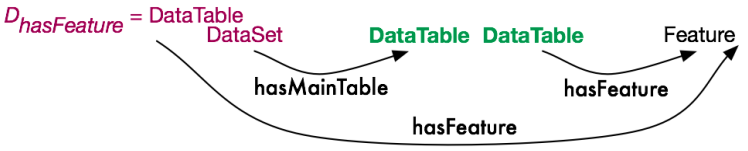


Fig. 1. Graphical depiction of the property chain hasMainTable ∘ hasFeature ⊑ hasFeature with the domain and range axioms of the two object properties

Thus, while the range of hasMainTable and domain of hasFeature match neatly—both are DataTable—this is not the case for their respective domains, because $D_{\text{hasMainTable}}$ = DataSet and $D_{\text{hasFeature}}$ = DataTable. Together with the represented knowledge that DataSet is a not-disjoint sister-class of DataTable, the combination of the domain and range axioms and the property chain causes the deduction that DataSet ⊑ DataTable, which is wrong with respect to the subject domain semantics. Put differently: the real flaw is either in the domain and range axioms of either one of the two object properties, or there is a flaw in the chain axiom. We will revisit this issue in Evaluation 4. ♢

To foster development of good quality ontologies, the ontologist should at least be informed about probable modelling flaws and be guided in the right direction to revise axioms in some way. This requires introduction of new constraints on

the use of property chains in order to guarantee correct and meaningful reasoning with respect to the subject domain, and explanation of the derivations. We shall address the issues with a non-standard reasoning service, called the *Property Chain Compatibility Service* (*ProChainS*).

3.1 The Property Chain Compatibility Service

Given Definition 1 on role inclusion axioms, we need to consider cases 3, 4, and 5 and, without loss of generality, for each OWL object property, if a domain and range is declared, exactly one domain and range axiom is provided. Recall the notation as in Definition 1 and 2: e.g., for Case 3, we may have a property chain $S_1 \circ S_2 \circ S_3 \sqsubseteq R$ where each property has corresponding domain and range D_{S1}, R_{S1}, D_{S2}, R_{S2}, D_{S3}, R_{S3}, D_R, and R_R. The three cases with the constraints that must hold are described formally in Definition 4, which concerns principally how the domain and range of the object properties used in the property chain expressions should relate with respect to their position in the class hierarchy.

Definition 4 (Property Chain Compatibility Service (*ProChainS*)). *For each set of object properties, $R, S_1, \ldots, S_n \in \mathcal{R}$, $\mathcal{R}$ the set of OWL object properties (V_{OP} in [1]) in OWL ontology $\mathcal{O}$, and $S_i \prec R$ with $1 \leq i \leq n$, $\mathcal{O}$ adheres to the constraints of Definition 1 (and, more generally, the OWL 2 specification [1]), and user-defined domain and range axioms as defined in Definition 2, for each of the property chain expression, select either one of the three cases:*

Case S. *Property chain pattern as $S_1 \circ S_2 \circ \ldots \circ S_n \sqsubseteq R$. Test whether:*
- **Test S-a.** $\mathcal{O} \models R_{S1} \sqsubseteq D_{S2}, \ldots, R_{Sn-1} \sqsubseteq D_{Sn}$;
- **Test S-b.** $\mathcal{O} \models D_{S1} \sqsubseteq D_R$;
- **Test S-c.** $\mathcal{O} \models R_{Sn} \sqsubseteq R_R$;

Case RS. *Property chain pattern as $R \circ S_1 \circ \ldots \circ S_n \sqsubseteq R$. Test whether:*
- **Test RS-a.** $\mathcal{O} \models R_{S1} \sqsubseteq D_{S2}, \ldots, R_{Sn-1} \sqsubseteq D_{Sn}$;
- **Test RS-b.** $\mathcal{O} \models R_R \sqsubseteq D_{S1}$;
- **Test RS-c.** $\mathcal{O} \models R_{Sn} \sqsubseteq R_R$;

Case SR. *Property chain pattern as $S_1 \circ \ldots \circ S_n \circ R \sqsubseteq R$. Test whether:*
- **Test SR-a.** $\mathcal{O} \models R_{S1} \sqsubseteq D_{S2}, \ldots, R_{Sn-1} \sqsubseteq D_{Sn}$;
- **Test SR-b.** $\mathcal{O} \models D_{S1} \sqsubseteq D_R$;
- **Test SR-c.** $\mathcal{O} \models R_{Sn} \sqsubseteq D_R$;

An OWL property chain expression is said to be compatible iff the OWL 2 syntactic constraints hold and either `Case S`*, or* `Case RS`*, or* `Case SR` *holds.*

ProChainS is tested against several domain ontologies in the following two evaluations. To simplify the explanations, let us assume that each ontology $\mathcal{O}$ contains an OWLized DOLCE taxonomy (relevant are $\mathsf{PD} \sqsubseteq \mathsf{PT}$, $\mathsf{ED} \sqsubseteq \mathsf{PT}$, $\mathsf{NPED} \sqsubseteq \mathsf{ED}$, $\mathsf{ED} \sqsubseteq \neg\mathsf{PD}$, $\mathsf{NPED} \sqsubseteq \mathsf{PED}$, $\mathsf{POB} \sqsubseteq \mathsf{PED}$, and $\mathsf{NPED} \sqsubseteq \neg\mathsf{POB}$), subject domain classes (e.g. $\mathsf{DrugTreatment}$) involved in class expressions, DOLCE's $\mathsf{hasPart} \sqsubseteq \mathsf{PT} \times \mathsf{PT}$ and $\mathsf{hasParticipant} \sqsubseteq \mathsf{PD} \times \mathsf{ED}$, a part-of between perdurants ($\mathsf{involvedIn} \sqsubseteq \mathsf{PD} \times \mathsf{PD}$), and a $\mathsf{structuralPart} \sqsubseteq \mathsf{POB} \times \mathsf{POB}$.

Evaluation 2 (DMOP chains, Case S) The aforementioned DMOP ontology v5.2 contains a property chain realizes ∘ addresses ⊑ achieves, and several domain and range axioms for the properties, which is depicted in Fig. 2; the classes are all subclasses of DOLCE's NPED. Going through the three applicable tests of *ProChainS*, we obtain:

- `Test S-a` passes, because $R_{\texttt{realizes}} \sqsubseteq D_{\texttt{addresses}}$;
- `Test S-b`, "$D_{\texttt{realizes}} \sqsubseteq D_{\texttt{achieves}}$?", holds: they are both DM-Operation;
- `Test S-c`, "$R_{\texttt{addresses}} \sqsubseteq R_{\texttt{achieves}}$?", fails because $R_{\texttt{addresses}}$ is the union of DM-Task and OptimizationProblem.

Thus, the chain can be instantiated, but if the range of addresses in a class expression is a subclass of OptimizationProblem, then all its instances will be classified as being a member of DM-Task as well, given that the two classes are not declared disjoint. If the two classes would have been declared disjoint, then the ontology would have become inconsistent, with as root problem a "bad individual" member of DM-Operation, instead of pointing to the issue with $R_{\texttt{addresses}}$ and $R_{\texttt{achieves}}$. In fact, even the current classification is undesirable: tasks and problems are clearly distinct entities. In this case, the lead ontology developer chose to revise the domain and range restrictions of the addresses property to have the chain functioning as intended (included in v5.3, d.d. 10-1-2012)[2]. ♢

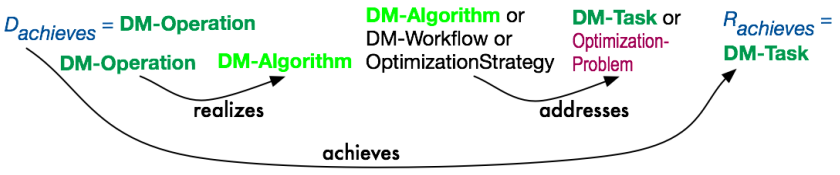


Fig. 2. The realizes ∘ addresses ⊑ achieves chain in DMOP v5.2, with the domain and range axioms of the participating properties; the matching chain is indicated in green, the problematic one in Arial narrow maroon font

Evaluation 3 (Pharmacogenomics chains, Case SR) Reconsider the pharmacogenomics ontology with the chain hasPart ∘ hasParticipant ⊑ hasParticipant and its use with drugs and treatments [5], and aforementioned axioms. With *ProChainS*:

- `Test SR-a` is trivially satisfied ($i = 1$);
- `Test SR-b` "$D_{\texttt{hasPart}} \sqsubseteq D_{\texttt{hasParticipant}}$?" does not hold because PD ⊑ PT;
- `Test SR-c` "$R_{\texttt{hasPart}} \sqsubseteq D_{\texttt{hasParticipant}}$?" does not hold because PD ⊑ PT.

If $\mathcal{O} \models$ DrugTreatment ⊑ PT and $\mathcal{O} \models$ DrugGeneInteraction ⊑ PT—not included in [5], however—then the OWL reasoners will infer they are subclasses of PD.

If we add DrugTreatment ⊑ ED and DrugGeneInteraction ⊑ ED to $\mathcal{O}$, then it deduces that the two classes are inconsistent because ED ⊑ ¬PD. Dumontier's hasPart thus holds only if DrugTreatment and DrugGeneInteraction are

[2] Whether this is the best option in an absolute sense is a separate issue; e.g., one can reify the relations and have whatever is achieved be subsumed by DOLCE's Achievement or whatever is realised be subsumed by BFO's Realizable and add corresponding class expressions for the participating classes.

subclasses of PD (perdurants or 'processes'), hence, we can refine the property chain into $\mathsf{involvedIn}^- \circ \mathsf{hasParticipant} \sqsubseteq \mathsf{hasParticipant}$. Assessing the tests for *Case SR* again, then `Test SR-b` and `Test SR-c` do hold, because $D_{\mathtt{involvedIn}^-} = R_{\mathtt{involvedIn}^-} = D_{\mathtt{hasParticipant}} = \mathsf{PD}$. Thus, we now have a property chain that is *guaranteed* not to lead to an inconsistency when the object properties are used in OWL class expressions. ♢

3.2 Managing Consequences of Property Chains

As Evaluation 3 shows, the ontology may not necessarily be inconsistent when viewed purely from a logic perspective, and, in fact, classify one or more of the participating classes elsewhere in the taxonomy with respect to where it was added originally (be this ontologically correct or not). Put differently, one cannot enforce *ProChainS*'s outcomes on the user. Be they undesired inferences or inconsistencies in the class hierarchy, it is important to have an explanation that those consequences are due to the property chain.

Now that we know what and how to check whether a declared property chain is logically and ontologically correct, it is also possible to devise support for identifying modelling defects, communicating this to the user, and suggest options to correct it in a similar way as for *SubProS*. Given that it can be considered an ontological reasoning service of one aspect of the ontology only, a less comprehensive approach can be taken compared to the formal foundations of computing explanations or root justifications [6, 2], as in this case we do not have to find the root anymore and, in fact, can make use of certain explanation features that are implemented already. We propose the following ontology updates, in case any of the tests fails:

A. If `Test S-a`, `Test RS-a`, or `Test SR-a` fails, check for any violating pair R_{Si}, D_{Si+1} whether:
   (i) $\mathcal{O} \models R_{Si} \sqsubseteq \neg D_{Si+1}$, then raise a warning "Incompatible domain and range of R_{Si}, D_{Si+1} in the property chain expression. This is *certain* to lead to an inconsistent class when the properties are used in class axioms, and an inconsistent ontology when used in assertions about instances", and propose the following minimal corrections:
      ⋆ Change the range of S_i such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$, or
      ⋆ Change the domain of S_{i+1} such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$;
      ⋆ Change the property chain such that a compatible property participates;
   (ii) $\mathcal{O} \models D_{Si+1} \sqsubseteq R_{Si}$, then raise a warning "Incompatible domain and range of R_{Si}, D_{Si+1} in the property chain expression. This *either* results in an inconsistent class when the properties are used in class axioms and an inconsistent ontology when used in assertions about instances, *or* results in a classification of D_{Si+1} elsewhere in the class hierarchy", and propose the following minimal corrections:

- ⋆ Change the range of S_i such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$, or
- ⋆ Change the domain of S_{i+1} such that $\mathcal{O} \models R_{Si} \sqsubseteq D_{Si+1}$;
- ⋆ Change the property chain such that a compatible property participates;
- ⋆ Let the reasoner classify D_R as a subclass of D_{S1} and accept this inference, provided $\mathcal{O} \not\models D_R \sqsubseteq \bot$;

B. If `Test S-b` fails, then raise a warning "Incompatible domain and range of D_{S1}, D_R in the property chain expression, which will induce a classification of D_R elsewhere in the taxonomy or an inconsistency" and propose the following options:
- ⋆ Change the domain of R or S_1 such that $\mathcal{O} \models D_{S1} \sqsubseteq D_R$, or
- ⋆ Let the reasoner classify D_{Si+1} as a subclass of R_{Si} and accept this inference, provided $\mathcal{O} \not\models D_{Si+1} \sqsubseteq \bot$;

and similarly for the respective ranges of R and S_n in `Test S-c`.

C. If `Test RS-b` fails, then raise a warning "Incompatible domain and range of D_{S1}, R_R in the left-hand-side of the property chain expression, which will induce a classification of R_R elsewhere in the taxonomy or in inconsistency" and propose:
- ⋆ Change the domain of S_1 or range of R such that $\mathcal{O} \models D_{S1} \sqsubseteq R_R$, or
- ⋆ Let the reasoner classify R_R as a subclass of D_{S1} and accept this inference, provided $\mathcal{O} \not\models R_R \sqsubseteq \bot$;

and similarly for the respective ranges of R and S_n in `Test RS-c`.

D. If `Test SR-b` fails then raise a warning "Incompatible domain and range of D_{S1}, D_R in the property chain expression, which will induce a reclassification or inconsistency of D_{S1}" and propose the following options:
- ⋆ Change the domain of S_1 or R such that $\mathcal{O} \models D_{S1} \sqsubseteq D_R$, or
- ⋆ Let the reasoner classify D_{S1} as a subclass of R_R and accept this inference, provided $\mathcal{O} \not\models D_{S1} \sqsubseteq \bot$;

and similarly for the range of S_n (compared to the range of R) in `Test SR-c`.

E. Run *ProChainS* again if any changes have been made in steps A-D.

ProChainS and the management of its consequences is evaluated with the DMOP ontology.

Evaluation 4 (*Assessing DMOP chains*) Recollect the property chain problem described in Example 1 with DMOP v5.2. DMOP uses the $\mathcal{SROIQ}$ features, has some 573 classes, 1021 subclass axioms, 113 object properties, and 11 property chains. Eight chains raise a warning with *ProChainS*, of which 3 cause a classification of classes elsewhere in the taxonomy due to the chain expressions. Among others, there is $\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}$ of Example 1, which is an instance of *Case SR*.

- `Test SR-a` passes trivially, for $i = 1$.
- `Test SR-b` fails: $D_{\mathsf{hasMainTable}} = \mathsf{DataSet}$ and $D_{\mathsf{hasFeature}} = \mathsf{DataTable}$, but DataSet is a not-disjoint sister-class of DataTable, so $\mathcal{O} \not\models D_{\mathsf{hasMainTable}} \sqsubseteq D_{\mathsf{hasFeature}}$, therefore `Test SR-b` fails and $\mathsf{DataSet} \sqsubseteq \mathsf{DataTable}$ is deduced, which is deemed undesirable. Step D's suggestions to revise the ontology are either to change the domain or to accept the new classification; the DMOP

domain experts chose the first option for revision and changed the domain of hasFeature into DataSet ⊔ DataTable, which is included in DMOP v5.3.

– `Test SR-c` passes, as both are DataTable.

No inconsistencies or unexpected classifications were detected with the other five chains, mainly thanks to the absence of disjointness axioms. For instance, the *Case S* described in Evaluation 2 with realizes ∘ addresses ⊑ achieves: `Test S-a` and `S-b` pass, but `S-c` does not, because $\mathcal{O} \not\models R_{\text{DM-Task}\sqcup\text{OptimizationProblem}} \sqsubseteq R_{\text{DM-Task}}$. A subclass of OptimizationProblem together with relevant property expressions declared for the chain results in an undesirable deduction that it is a subclass of DM-Task; hence, step B's first option (for `S-c`) to revise the ontology was the chosen option, i.e., removing OptimizationProblem from the range axiom of addresses (as well as removing OptimizationStrategy from the domain axiom), which is included as such in DMOP v5.3. ♢

Thus, *SubProS* and *ProChainS* together cover all types of modelling flaws with their root causes and options to revise them in OWL ontologies with respect to property hierarchies, domain and range axioms to type the property, a property's characteristics, and property chains.

4 Conclusions

We have identified exhaustively the type of flaws that can occur in the object property box regarding simple property subsumption and property chaining and proposed two compatibility services, *SubProS* and *ProChainS*, that both check for meaningful object property hierarchies and property chaining. Thanks to being able to identify the root cause, proposals for how to revise the ontology were made, including the options to change the object property expressions or the class hierarchy, and how, or accepting the deductions. This was evaluated with several ontologies were flaws could be detected and were solved, therewith improving the ontology's quality.

We are currently looking into an efficient algorithm to implement *SubProS* and *ProChainS* and a user-friendly interface to help revising flaws.

Acknowledgements. The author wishes to thank Melanie Hilario for her feedback on the subject domain and object properties in the DMOP ontology.

References

1. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C (October 27, 2009), http://www.w3.org/TR/owl2-syntax/
2. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Proceedings of the World Wide Web Conference (WWW 2005), Chiba, Japan, May 10-14 (2005)
3. Rector, A.L., Drummond, N., Horridge, M., Rogers, J.D., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 63–81. Springer, Heidelberg (2004)

4. Roussey, C., Corcho, O., Vilches-Blázquez, L.: A catalogue of OWL ontology antipatterns. In: Proc. of K-CAP 2009, pp. 205–206 (2009)
5. Dumontier, M., Villanueva-Rosales, N.: Modeling life science knowledge with OWL 1.1. In: Fourth International Workshop OWL: Experiences and Directions 2008 (OWLED 2008 DC), Washington, DC (metro), April 1-2 (2008)
6. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
7. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B.: Repairing Unsatisfiable Concepts in OWL Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 170–184. Springer, Heidelberg (2006)
8. Keet, C.M., Artale, A.: Representing and reasoning over a taxonomy of part-whole relations. Applied Ontology 3(1-2), 91–110 (2008)
9. Guarino, N., Welty, C.: An overview of OntoClean. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 151–159. Springer (2004)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings of KR 2006, pp. 452–457 (2006)
11. Beisswanger, E., Schulz, S., Stenzhorn, H., Hahn, U.: BioTop: An upper domain ontology for the life sciences - a description of its current structure, contents, and interfaces to OBO ontologies. Applied Ontology 3(4), 205–212 (2008)
12. Massacci, F.: Decision procedures for expressive description logics with intersection, composition, converse of roles and role identity. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 193–198 (2001)
13. Schmidt-Schauss, M.: Subsumption in KL-ONE is undecidable. In: Proceedings of 1st Conference on Knowledge Representation and Reasoning (KR 1989), pp. 421–431 (1989)
14. Wessel, M.: Obstacles on the way to qualitative spatial reasoning with description logics: some undecidability results. In: Goble, C.A., McGuinness, D.L., Möller, R., Patel-Schneider, P.F. (eds.) Proceedings of the International Workshop in Description Logics (DL 2001), Stanford, CA, USA, August 1-3, vol. 49. CEUR WS (2001)
15. Glimm, B., Rudolph, S., Völker, J.: Integrated Metamodeling and Diagnosis in OWL 2. 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. 257–272. Springer, Heidelberg (2010)
16. Boran, A., Bedini, I., Matheus, C.J., Patel-Schneider, P.F., Keeney, J.: Choosing between axioms, rules and queries: Experiments in semantic integration techniques. In: Eigth International Workshop OWL: Experiences and Directions (OWLED 2011), San Francisco, California, USA, June 5-6 (2011)
17. Koutsomitropoulos, D.A., Solomou, G.D., Papatheodorou, T.S.: Metadata and semantics in digital object collections: A case-study on CIDOC-CRM and Dublin Core and a prototype implementation. Journal of Digital Information 10(6) (2009)
18. Hilario, M., Nguyen, P., Do, H., Woznica, A., Kalousis, A.: Ontology-Based Meta-Mining of Knowledge Discovery Workflows. In: Jankowski, N., Duch, W., Grąbczewski, K. (eds.) Meta-Learning in Computational Intelligence. SCI, vol. 358, pp. 273–315. Springer, Heidelberg (2011)

Validating Ontologies with OOPS!

María Poveda-Villalón, Mari Carmen Suárez-Figueroa, and Asunción Gómez-Pérez

Ontology Engineering Group. Departamento de Inteligencia Artificial.
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{mpoveda,mcsuarez,asun}@fi.upm.es

Abstract. Ontology quality can be affected by the difficulties involved in ontology modelling which may imply the appearance of anomalies in ontologies. This situation leads to the need of validating ontologies, that is, assessing their quality and correctness. Ontology validation is a key activity in different ontology engineering scenarios such as development and selection. This paper contributes to the ontology validation activity by proposing a web-based tool, called OOPS!, independent of any ontology development environment, for detecting anomalies in ontologies. This tool will help developers to improve ontology quality by automatically detecting potential errors.

Keywords: ontology, pitfalls, ontology evaluation, ontology validation.

1 Introduction

The emergence of ontology development methodologies during the last decades has facilitated major progress, transforming the art of building ontologies into an engineering activity. The correct application of such methodologies benefits the ontology quality. However, such quality is not always guaranteed because developers must tackle a wide range of difficulties and handicaps when modelling ontologies [1, 2, 11, 15]. These difficulties can imply the appearance of anomalies in ontologies. Therefore, ontology evaluation, which checks the technical quality of an ontology against a frame of reference [18], plays a key role in ontology engineering projects.

Ontology evaluation, which can be divided into validation and verification [18], is a complex ontology engineering process mainly due to two reasons. The first one is its applicability in different ontology engineering scenarios, such as development and reuse, and the second one is the abundant number of approaches and metrics [16].

One approach for validating ontologies is to analyze whether the ontology is conform to ontology modelling best practices; in other words, to check whether the ontologies contain anomalies or pitfalls. In this regard, a set of common errors made by developers during the ontology modelling is described in [15]. Moreover, in [10] a classification of errors identified during the evaluation of different features such as consistency, completeness, and conciseness in ontology taxonomies is provided. Finally, in [13] authors identify an initial catalogue of common pitfalls.

In addition, several tools have been developed to alleviate the dull task of evaluating ontologies. These tools support different approaches like (a) to check the consistency of the ontology, (b) to check the compliance with the ontology language used to

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 267–281, 2012.

build the ontology or (c) to check modelling mistakes. In this context, our goal within this paper is to present an on-line tool that supports the automatic detection of pitfalls in ontologies. This tool is called OOPS! (OntOlogy Pitfall Scanner!) and represents a new option for ontology developers within the great range of ontology evaluation tools as it enlarges the list of errors detected by most recent and available works (like MoKi[1] [12] and XD Analyzer[2]). In addition, OOPS! is executed independently of the ontology development platform without configuration or installation and it also works with main web browsers (Firefox, Chrome, Safari and Internet Explorer[3]).

The remainder of this paper is structured as follows: Section 2 presents related work in ontology evaluation techniques and tools while Section 3 describes the pitfall catalogue taken as starting point in our evaluation approach. Section 4 shows OOPS! architecture and an illustrative use case. In Section 5 the user-based evaluation carried out over OOPS! is detailed. Finally, Section 6 outlines some conclusions and future steps to improve OOPS!.

2 State of the Art

In the last decades a huge amount of research on ontology evaluation has been performed. Some of these attempts have defined a generic quality evaluation framework [6, 9, 10, 17], other authors proposed to evaluate ontologies depending on the final (re)use of them [18], others have proposed quality models based on features, criteria and metrics [8, 3], and in recent times methods for pattern-based evaluation have also emerged [5, 14,]. A summary of generic guidelines and specific techniques for ontology evaluation can be found on [16].

Despite vast amounts of frameworks, criteria, and methods, ontology evaluation is still largely neglected by developers and practitioners. The result is many applications using ontologies following only minimal evaluation with an ontology editor, involving, at most, a syntax checking or reasoning test. Also, ontology practitioners could feel overwhelmed looking for the information required by ontology evaluation methods, and then, to give up the activity. That problem could stem from the time-consuming and tedious nature of evaluating the quality of an ontology.

To alleviate such a dull task technological support that automate as many steps involved in ontology evaluation as possible have emerged. In 2002 Fernández-López and Gómez-Pérez [7] developed ODEClean providing technological support to OntoClean Method [19] in the form of a plug-in for the WebODE ontology development environment. Few years later, ODEval[4] [4] was developed to help users evaluating RDF(S) and DAML+OIL concept taxonomies. Within those tools that support OWL ontologies we can find some developed as plug-ins for desktop applications as

[1] `https://moki.fbk.eu/website/index.php` (Last visit on 14-04-2012)

[2] `http://neon-toolkit.org/wiki/XDTools` (Last visit on 14-04-2012)

[3] You may experience some layout strange behaviours with Internet Explorer.

[4] Even thought the approach described in the bibliographic documentation addresses OWL ontologies the on-line application available at `http://oeg1.dia.fi.upm.es/odeval/ODEval.html` only works with RDF(S) and DAML+OIL ontologies.

XDTools plug-in for NeOn Toolkit and OntoCheck plug-in for Protégé. This kind of tools have two main disadvantages: (a) to force the user to install the ontology editor in which they are included as a plug-in and (b) to tend to be outdated, and sometimes incompatible, as long the core desktop applications evolve to new versions. Other tools rely on web based technologies as MoKi [12] that consists on a wiki-based ontology editor that incorporates ontology evaluation functionalities. In this case, although a testing user account is provided to try out the tool, an installation process is also required to set up the wiki system. Finally, command line tools like Eyeball[5] have been proposed. Eyeball is also available as Java API, what makes its use more suitable for users with technological background. In order to provided a more user-friendly version of Eyeball a graphical user interface is also provided, however it is still in an experimental phase.

As already mentioned in Section 1, different ontology evaluation tools can follow different approaches, and therefore check the ontology quality against different kind of issues. After performing an analysis of available tools we have realized that the following six dimensions[6] (Fig. 1) can be identified with respect to ontology quality:

- **Human understanding** dimension refers to whether the ontology provides enough information so that it can be understood from a human point of view. This aspect is highly related to the ontology documentation and clarity of the code.
- **Logical consistency** dimension refers to whether (a) there are logical inconsistencies or (b) there are parts of the ontology that could potentially lead to an inconsistency but they cannot be detected by a reasoner unless the ontology is populated.
- **Modelling issues** dimension refers to whether the ontology is defined using the primitives provided by ontology implementation languages in a correct way, or whether there are modelling decision that could be improved.

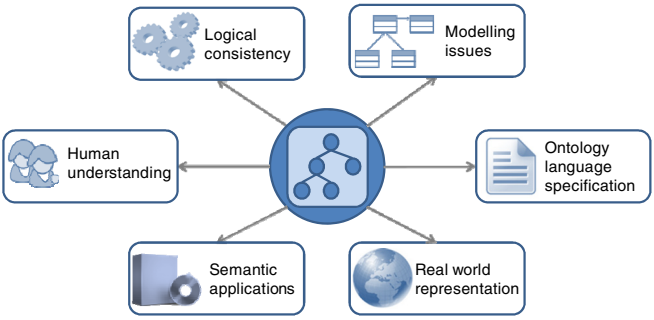


Fig. 1. Ontology Evaluation Dimensions

- **Ontology language specification** dimension refers to whether the ontology is compliant (e.g., syntax correctness) with the specifications of the ontology language used to implement the ontology.

[5] `http://jena.sourceforge.net/Eyeball/`

[6] It should be noted that this enumeration is not intended to be exhaustive and there could be more aspects to check an ontology against.

- **Real world representation** dimension refers to how accurately the ontology represents the domain intended for modelling. This dimension should be checked by humans (e.g., ontology engineers and domain experts).
- **Semantic applications** dimension refers to whether the ontology is fit for the software that uses it, for example checking availability, format compatibility, etc.

The results of the comparative analysis performed over available tools that support ontologies written in OWL (including OOPS!) with respect to the six aforementioned dimensions is shown in Table 1. This table presents the comparison according to (a) three general characteristics (whether they are IDE[7] independent, if a GUI[8] is provided or whether an installation process is needed) and (b) the ontology quality dimensions they consider, to some extent, into their algorithm. Ticks (✓) appearing in Table 1 mean that the given tool fulfils a general characteristic or it addresses a dimension; while crosses (×) mean that the tool does not fulfil the characteristic or does not provide any support for the given dimension. In this sense, we say that an ontology evaluation tool addresses a given dimension if it checks the ontology quality against at least one issue related to that dimension. For example, when a given tool checks whether the ontology contains individuals belonging to two disjoint classes, we can argue that this tool addresses the logical consistency dimension.

Table 1. Ontology evaluation tools comparison

	XD-Tools	OntoCheck	EyeBall	Moki	**OOPS!**
General Characteristics					
IDE development independent	×	×	✓	×	✓
GUI provided	✓	✓	× (experimental)	✓	✓
No installing process required	×	×	×	×	✓
Ontology Evaluation Dimensions					
Human understanding	✓	✓	×	✓	✓
Logical consistency	✓	×	✓	×	✓
Modelling issues	✓	×	×	✓	✓
Ontology language specification	✓	×	✓	×	✓
Real world representation	×	×	×	×	✓
Semantic applications	×	×	×	×	×

3 Pitfall Catalogue So Far

One of the crucial issues in ontology evaluation is the identification of anomalies or bad practices in the ontologies. As already mentioned in Section 1, different research works have been focused on establishing sets of common errors [15, 10, 11, 13].

Having the pitfall catalogue presented in [13] as starting point, we are performing a continuous process of maintenance, revision, and enlargement of such a catalogue as

[7] Integrated Development Environment.
[8] Graphical User Interface.

long as we discover new pitfalls during our research. Up to the moment of writing this paper, 5 new pitfalls have been included in the catalogue (P25-P29). Thus, the current version of the catalogue[9] consists on the 29 pitfalls shown in Table 2. Pitfalls in such a table are grouped by the quality dimensions presented in Section 2.

Table 2. Catalogue of pitfalls grouped by ontology quality dimension

Human understanding	Modelling issues
• P1. Creating polysemous elements • P2. Creating synonyms as classes • P7. Merging different concepts in the same class • P8. Missing annotations • P11. Missing domain or range in properties • P12. Missing equivalent properties • P13. Missing inverse relationships • P19. Swapping intersection and union • P20. Misusing ontology annotations • P22. Using different naming criteria in the ontology	• P2. Creating synonyms as classes • P3. Creating the relationship “is” instead of using "rdfs:subClassOf", "rdf:type" or "owl:sameAs" • P4. Creating unconnected ontology elements • P5. Defining wrong inverse relationships • P6. Including cycles in the hierarchy • P7. Merging different concepts in the same class • P10. Missing disjointness • P17. Specializing too much a hierarchy • P11. Missing domain or range in properties • P12. Missing equivalent properties • P13. Missing inverse relationships • P14. Misusing "owl:allValuesFrom" • P15. Misusing “not some” and “some not” • P18. Specifying too much the domain or the range • P19. Swapping intersection and union • P21. Using a miscellaneous class • P23. Using incorrectly ontology elements • P24. Using recursive definition • P25. Defining a relationship inverse to itself • P26. Defining inverse relationships for a symmetric one • P27. Defining wrong equivalent relationships • P28. Defining wrong symmetric relationships • P29. Defining wrong transitive relationships
Logical consistency	
• P5. Defining wrong inverse relationships • P6. Including cycles in the hierarchy • P14. Misusing "owl:allValuesFrom" • P15. Misusing “not some” and “some not” • P18. Specifying too much the domain or the range • P19. Swapping intersection and union • P27. Defining wrong equivalent relationships • P28. Defining wrong symmetric relationships • P29. Defining wrong transitive relationships	
Real world representation	
• P9. Missing basic information • P10. Missing disjointness	

As part of the maintaining process of the pitfall catalogue, our intention is to extend it also with pitfalls proposed by users. Up to now, the suggestions from users are gathered using a form where they can describe what they consider to be a pitfall. In such a form, users can also add information about (a) how this suggested pitfall could be solved, (b) how important it could be to solve the pitfall when it appears, and (c) how it could be automatically detected. After a revision process the suggested pitfalls could be included in the catalogue.

4 OOPS!

OOPS! is a web-based tool, independent of any ontology development environment, for detecting potential pitfalls that could lead to modelling errors. This tool is

[9] The official catalogue consists on the list of the pitfalls as well as their descriptions and it can be found at `http://www.oeg-upm.net/oops/catalogue.jsp`

intended to help ontology developers during the ontology validation activity [18], which can be divided into diagnosis and repair. Currently, OOPS! provides mechanisms to automatically detect as many pitfalls as possible, thus helps developers in the diagnosis activity. In the near future OOPS! will include also prescriptive methodological guidelines for repairing the detected pitfalls.

In this section we first explain the internal architecture of OOPS! and its main components (Section 4.1), followed by an exemplary use case showing how OOPS! helps ontology developers during the validation activity (Section 4.2).

4.1 How OOPS! is Internally Organized

Fig. 2 presents OOPS! underlying architecture in which it can be seen that OOPS! takes as input an ontology and the pitfall catalogue in order to produce a list of evaluation results. OOPS! is a web application based on Java EE[10], HTML[11], jQuery[12], JSP[13] and CSS[14] technologies. The web user interface consists on a single view where the user enters the URI pointing to or the RDF document describing the ontology to be analyzed. Once the ontology is parsed using the Jena API[15], it is scanned looking for pitfalls from those available in the pitfall catalogue (Section 3). During this scanning phase, the ontology elements involved in potential errors are detected; in addition, warnings regarding RDF syntax and some modelling suggestions are generated. Finally, the evaluation results are displayed by means of the web user interface showing the list of appearing pitfalls, if any, and the ontology elements affected as well as explanations describing the pitfalls.

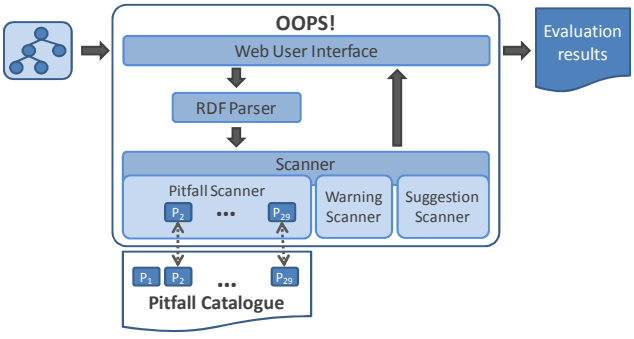


Fig. 2. OOPS! architecture

The "Pitfall Scanner" module, shown in Fig. 2, implements the automatic detection of a subset of 21 pitfalls of those included in the catalogue[16]. This subset

[10] http://www.oracle.com/technetwork/java/javaee/overview/index.html
[11] http://www.w3.org/html/wg/
[12] http://jquery.com/
[13] http://www.oracle.com/technetwork/java/javaee/jsp/index.html
[14] http://www.w3.org/Style/CSS/
[15] http://jena.sourceforge.net/
[16] http://www.oeg-upm.net/oops/catalogue.jsp

includes pitfalls related to the following dimensions: (a) human understanding (P2, P7, P8, P11, P12, P13, P19, P20, and P22); (b) logical consistency (P5, P6, P19, P27, P28, and P29); (c) real world representation (P10); and (d) modelling issues (P2, P3, P4, P5, P6, P7, P10, P11, P12, P13, P19, P21, P24, P25, P26, P27, P28, and P29). It is worth mentioning that since the catalogue consists on a list of pitfalls defined in natural language, they have to be transformed into a formal or procedural language in order to detect them automatically. Currently, this transformation is implemented in OOPS! as a Java class for each of the 21 pitfalls. In order to detect a greater range of pitfalls, developers should implement the appropriate Java class and plug it into the Pitfall Scanner" module. Up to now, OOPS! provides an on-line form[17] where users can suggest new pitfalls by describing them in natural language and attaching diagrams if needed. Once a pitfall suggestion is reviewed and accepted, it can be included in OOPS! by implementing the corresponding Java class as already mentioned.

The automatic detection of pitfalls has been approached in two different ways. On the one hand, some pitfalls (namely P3, P7, P12, P20, P21, and P22) have been automated by checking general characteristics of the ontology, for example the use of more than one naming convention to detect P22 (Using different naming criteria in the ontology). On the other hand, other pitfalls (namely P2, P4, P5, P6, P8, P10, P11, P13, P19, P24, P25, P26, P27, P28, and P29), related to the internal structure of the ontology, have been translated into patterns that indicate that a pitfall appears when a pattern is spotted. Fig. 3 shows some of the patterns used to detect pitfalls within OOPS!; for example, the pattern used to detect P5 (Defining wrong inverse relationships) consist on pairs of relationships defined as inverse but where the domain of one of them is not equal to the range of the other. Up to now, these patterns are spotted programmatically by means of a Java class; however, our aim is to transform into SPARQL[18] queries as many patterns as possible in future releases of OOPS!.

The module "Warning Scanner" identifies cases where a class or property is not defined as such by means of the corresponding OWL primitive, that is, related to the "ontology language specification" dimension presented in Section 2. It is worth mentioning that currently there is not a Java class looking for all the cases within the ontology. Instead, these warnings are spotted on running time during the execution of the "Pitfall Scanner" module so that only the classes and relationships related to the other pitfalls detection are flag up.

Finally, the module "Suggestion Scanner" looks for properties with equal domain and range axioms and proposes them as potential symmetric or transitive properties.

4.2 How OOPS! Works

OOPS! main functionality is to analyze ontologies[19] (a) via the URI in which an ontology is located or (b) via text input containing the source code of the ontology. As a

[17] http://www.oeg-upm.net/oops/submissions.jsp

[18] http://www.w3.org/TR/rdf-sparql-query/

[19] The input ontology must be implemented in OWL (http://www.w3.org/TR/2009/REC-owl2-primer-20091027/) or RDF (http://www.w3.org/TR/2004/REC-rdf-primer-20040210/).

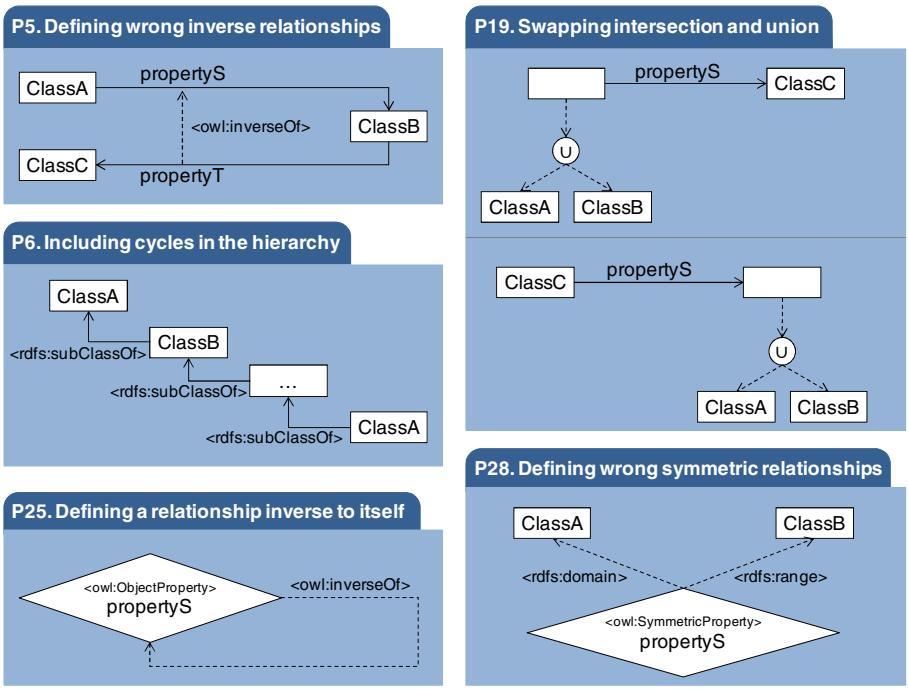


Fig. 3. Example of patterns defined to detect pitfalls

result of the analysis, OOPS! informs developers about which elements of the ontology are possibly affected by pitfalls.

Fig. 4 shows OOPS! home page[20] where a user can enter an ontology to be analyzed via URI or RDF coding. This page also presents a brief description of OOPS!. In addition, the menu in the right side of the web site contains links to (a) documentation related to OOPS! (the pitfall catalogue, a user guide, and a technical report) and (b) papers related to OOPS! and the research behind it. In addition, two different ways in which users can send their suggestion are also provided: (1) a questionnaire to send feedback after using the tool and (2) a form to suggest new pitfalls.

As result of analyzing the ontology provided by the user, OOPS! generates, as it is shown in Fig. 5[21], a new web page listing the appearing pitfalls in the ontology. This list provides information about (a) how many times a particular pitfall appears, (b) which specific ontology elements are affected by such a pitfall, and (c) a brief description about what the pitfall consist on.

It is worth mentioning that OOPS! output points to ontology elements identified as potential errors but they are not always factual errors as sometimes something can be considered a factual errors depending on different perspectives (such as the particular ontology being analyzed, its particular requirements, the characteristics of the domain intended for modelling, etc.). In this sense, there are seven pitfalls (P3, P8, P22, P25,

[20] http://www.oeg-upm.net/oops

[21] For the sake of clarity some screenshots have been reduced keeping the interesting information for each example.

P26, P28, and P29) that should be repaired when detected by OOPS! as they certainly point to modelling problems within the ontology; while the rest of pitfalls appearing in OOPS! output must be manually checked in order to discern whether they point to factual problems in the ontology being analyzed.

Fig. 4. OOPS! home page

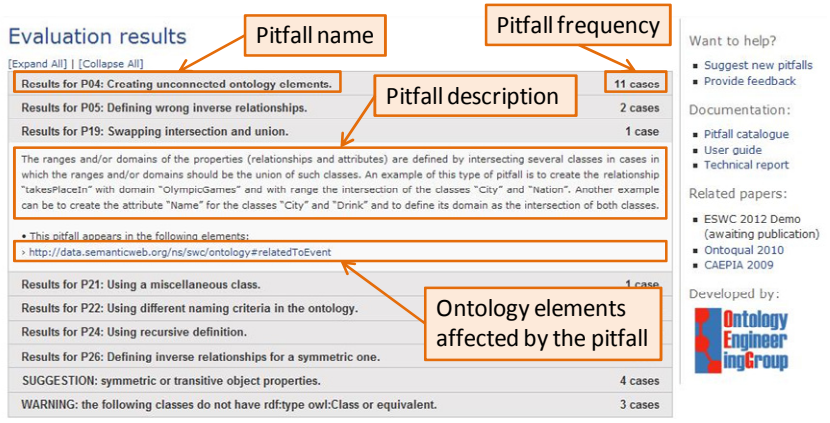


Fig. 5. Example of evaluation results web page generated by OOPS!

OOPS! shows the result for each pitfall in three different ways depending on the kind of pitfall. There are pitfalls that affect individual elements in the ontology, others affect more than one element, and there are also pitfalls that do not affect particular ontology elements but the whole ontology. In the following, an example of OOPS! execution is shown in order to clarify the different types of results a user can get from

OOPS! and how to interpret them. For illustrating the example we are using the Semantic Web Conference Ontology[22] (SWC).

After executing OOPS! with the SWC ontology, we obtain a summary of the pitfalls encountered as presented in Fig. 6 . Such a figure shows that 11 pitfalls have been detected as well as 1 suggestion and 1 warning. For the sake of simplicity, not all results will be detailed but only those contributing to an explanation of different kind of outputs or interpretations.

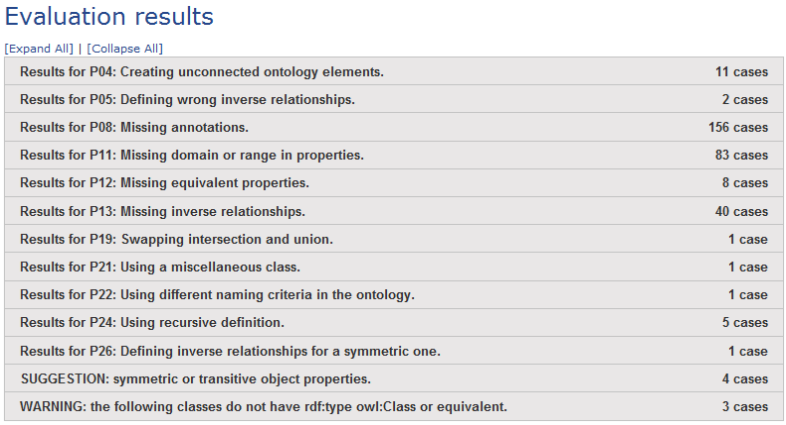


Fig. 6. Evaluation results for SWC ontology

As already mentioned, some pitfalls can affect individual ontology elements, other pitfalls can affect more than one element in the ontology, and others can affect the whole ontology. Fig. 7 shows an example[23] of a pitfall (P08. Missing annotations) that affects individual ontology elements. In this case, the output is grouped by (a) elements that have neither `rdfs:label` or `rdfs:comment` defined and (b) elements that have no `rdfs:comment`.

Fig. 8 shows an example of a pitfall (P05. Defining wrong inverse relationships) that affects more than one ontology element. In this case, when OOPS! detects a potential mistake while defining inverse relationships it provides the pair of relationships involved in such pitfall.

Fig. 9 shows a particular example of pitfall (P22. Using different naming criteria in the ontology), which affects the whole ontology. It is worth mentioning that the ontology elements shown in Fig. 9 represent just arbitrary example as P22 points to the fact of having different naming criteria along the ontology instead of between particular elements.

---

[22] Official URI is `http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf`. As the results shown in this paper may be affected by possible updates on the original ontology there is a copy of the ontology code used in this example on 18-04-2012 that can be found at `http://www.oeg-upm.net/files/mpoveda/EKAW2012/swc_2009-05-09.rdf`

[23] As it is shown in the top part of the example there have been found 156 cases of this pitfall, however just an excerpt of the results is shown due to space constraints. This case may also apply to further examples.

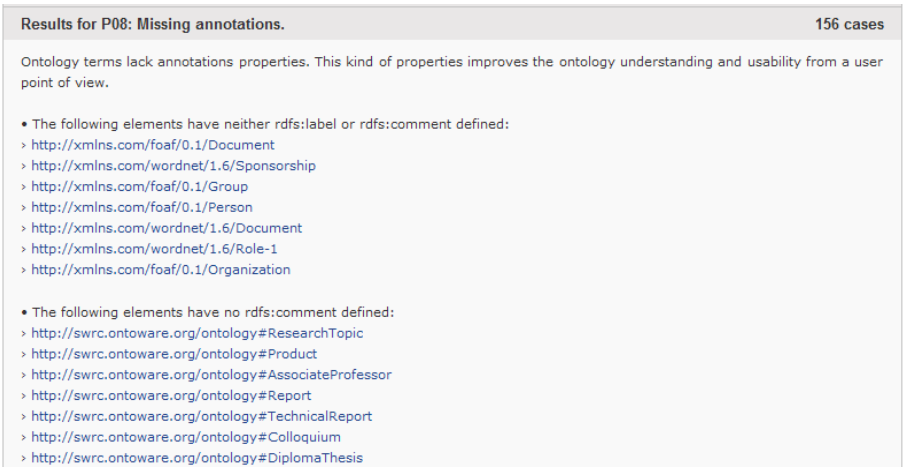


Fig. 7. Excerpt of an example of evaluation results for P08: Missing annotations

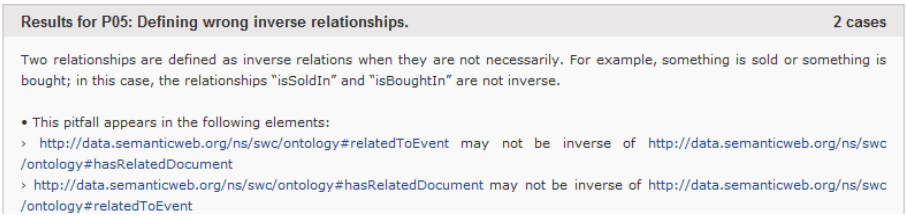


Fig. 8. Example of evaluation results for P05: Defining wrong inverse relationships

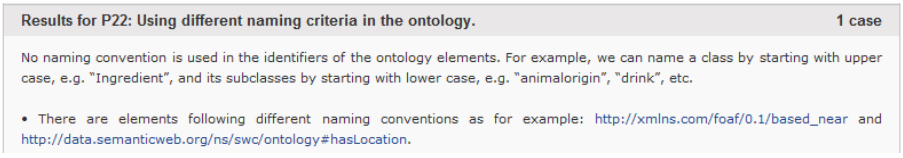


Fig. 9. Example of evaluation results for P22: Using different naming criteria in the ontology

5 User-Based Evaluation of OOPS!

OOPS! has been used in different research and educational projects with positive feedback and interesting comments from the developers involved in each case. In this section we briefly summarize a set of such cases, presenting qualitative results[24].

The first case we can mention is the use of OOPS! in the context of two Spanish research projects called *mIO!*[25] and *Buscamedia*[26]. Both projects involved the development of two ontologies about user context in mobile environments and multimedia

[24] Quantitative results are not provided because to test the same real case using the proposed tool and without the tool was not feasible due to the effort needed.

[25] `http://www.cenitmio.es/`

[26] `http://www.cenitbuscamedia.es/`

information objects respectively. In both projects the validation activity was carried out using OOPS!. After the diagnosis phase, the ontologies were repaired accordingly to OOPS! output. It is worth mentioning that by the time when the validation activities were carried out (September 2011) OOPS! did not provide a graphical user interface, but it was provided to ontology developers involved in the projects as a .jar file and its output was given in a .txt file.

A total of seven ontology developers were involved in the ontology validation activities within mIO! and Buscamedia use cases. Such developers provided positive feedback about (a) OOPS! usefulness, (b) the advantages of being IDE independent and (c) coverage of pitfalls detected by OOPS! in comparison with other tools. They also provided very valuable feedback about aspects that could be improved in OOPS! as for example (a) providing a graphical user interface, (b) providing more information about what the pitfalls consist on, and (c) considering the imported ontologies during the analysis. All these comments were taking into account and implemented in subsequent releases of OOPS!. Other suggestions as (a) to allow the evaluation of subsets of pitfalls and (b) to provide some recommendations to repair the pitfalls found within the ontology are currently considered as future work to be included in next releases.

The second case refers to a controlled experiment to test the benefits of using OOPS! during the ontology validation activity that was carried out with master students. This experiment was performed during the ATHENS course that took place in November 2011 at *Universidad Politécnica de Madrid*. Twelve master students working in pairs executed the experiment. Before the experiment, students were provided with (a) some explanations about OOPS! and ontology evaluation concepts, (b) the detailed tasks to be performed during the experiment, and (c) two different ontologies to be evaluated. After the experiment, we gathered students' feedback using questionnaires. Most of the students considered that OOPS! was very useful to evaluate the ontologies at hand and that its output shows clearly what are the problems detected and in which elements. Again in this experiment, main proposed feature to be added in OOPS! was to include guidelines about how to solve pitfalls. In addition, some of the students commented that it could be useful (a) to associate colours to the output indicating how critical the pitfalls are, like error and warnings recognition in many software IDEs and (b) to provide a way to see the lines of the file that the error considered is originated from.

Finally, we announced the release of the tool thorough several mailing lists[27] related to the Semantic Web so that all the people interested in analyzing their ontologies can use OOPS! and send feedback after that. Up to now we have received four feedback responses from users not related to any project or any controlled experiment. This feedback shows that even though all of the users think that OOPS! is very useful, three of them will always use it within their ontology developments or recommend it to a colleague while one of them will do sometimes. Also some strengths of the tool were explicitly pointed out by users as[28]: "*easy to use*", "*no installation required*", "*quick results*" and "*good to detect low level problems in ontologies*". However, the

[27] For example `http://lists.w3.org/Archives/Public/semantic-web/2012Mar/0064.html`

[28] The following comments have been taken literally from the feedback questionnaires.

richest side of this feedback is the set of proposals to improve the tool. The most important feedback in this regard refers to (a) show which pitfalls do not appear in the ontology, (b) include reasoning processes so that OOPS! would not complain when a property inherits domain/range from its superproperty, and (c) allow the evaluation of subsets of pitfalls.

Apart from the feedback received through the web questionnaire, we have also received comments and questions about OOPS! by email, what reveals users willingness to adopt this type of tools within their ontology developments. Within these feedback emails users also pointed out the need of developing systems like OOPS! in the context of ontological engineering. In addition, the following improvements for our tool were received: (a) to discard pitfalls involving terms properly marked as DEPRECATED following the OWL 2 deprecation pattern, (b) to take into account the different namespaces used within the ontology, (c) to look for URI misuse as using the same URI as two types of ontology elements, and (d) to look for non-standard characters in natural language annotations.

6 Conclusions and Future Work

Ontology evaluation is a key ontology engineering activity that can be performed following a variety of approaches and using different tools. In this paper we present (a) an evaluation approach based on the detection of anomalies in ontologies and (b) an on-line tool called OOPS! that automatically detects a subset of pitfalls from those gathered in a pitfall catalogue.

OOPS! represents a step forward within ontology evaluation tools as (a) it enlarges the list of errors detected by most recent and available works (e.g. MoKi [12] and XD Analyzer), (b) it is fully independent of any ontology development environment, and (c) it works with main web browsers (Firefox, Chrome, Safari, and Internet Explorer).

OOPS! has been tested in different settings (research and educational projects) with positive feedback from the ontology developers who evaluated their ontologies. Such ontology developers provided also interesting suggestions to improve the tool. Both feedback and suggestions have been provided via the feedback questionnaire available in OOPS! website. Apart from particular projects, OOPS! has been already used by other ontology developers who belong to different organizations (such as AtoS, Tecnalia, *Departament Arquitectura La Salle* at *Universitat Ramon Llull*, and Human Mobility and Technology Laboratory at CICtourGUNE). In fact, OOPS! is freely available to users on the Web. In addition, OOPS! is currently being tested by Ontology Engineering Group[29] members in order to debug it and extend its functionality.

As part of the continuous process of improving OOPS!, currently we are working in the improvement of the rules applied to automatically detect pitfalls to make them more accurate. In addition, we are also working on the maintenance of the pitfall catalogue. For this purpose, OOPS! web site includes a form to suggest new pitfalls, what allows extending the catalogue in a collaborative way. In addition, as long as we discover new pitfalls during our research, they will be included in the current catalogue.

[29] `http://www.oeg-upm.net/`

New releases of OOPS! could include the detection of both new pitfalls proposed by users and new errors identified by us. In this regard, more ambitious plans are (a) the development of a formal language to define pitfalls so that a user can define in a formal way the pitfalls to be detected, and (b) the implementation of mechanisms so that OOPS! can automatically interpret and process the defined pitfalls without encoding them manually.

Based on the feedback we have already received, we have planned to improve OOPS by allowing the validation of groups of pitfalls the user is interested in. To do this, we are going to classify pitfalls in different categories according to the ontology quality criteria identified in [9] and [10]. This new feature will provide more flexibility to the ontology validation, since it will allow users to diagnose their ontologies just with respect to the dimensions or pitfalls they are interested in.

Regarding increasing OOPS! features to help the user in the activity of repairing the detected pitfalls, our plan is to provide more detailed descriptions about the pitfalls and some prescriptive methodological guidelines about how to solve them. Our plans also include associating priority levels to each pitfall according to the different types of consequences they can convey when appearing in an ontology. This feature will be useful to prioritize actions to be taken during the repairing task.

Finally, future plans also include making REST services available in order to allow other developments to use and integrate the pitfall scanner functionalities within their applications.

Acknowledgments. This work has been partially supported by the Spanish projects *BabelData* (TIN2010-17550) and *BuscaMedia* (CENIT 2009-1026).

References

1. Aguado de Cea, G., Gómez-Pérez, A., Montiel-Ponsoda, E., Suárez-Figueroa, M.C.: Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 32–47. Springer, Heidelberg (2008)
2. Blomqvist, E., Gangemi, A., Presutti, V.: Experiments on Pattern-based Ontology Design. In: Proceedings of K-CAP 2009, pp. 41–48 (2009)
3. Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P.: A Semiotic Metrics Suite for Assessing the Quality of Ontologies. Data and Knowledge Engineering 55(1), 84–102 (2005)
4. Corcho, O., Gómez-Pérez, A., González-Cabero, R., Suárez-Figueroa, M.C.: ODEval: a Tool for Evaluating RDF(S), DAML+OIL, and OWL Concept Taxonomies. In: 1st IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI 2004), Toulouse, France, August 22-27 (2004)
5. Djedidi, R., Aufaure, M.-A.: *ONTO-eVOaL* an ontology evolution approach guided by pattern modeling and quality evaluation. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 286–305. Springer, Heidelberg (2010)
6. Duque-Ramos, A., López, U., Fernández-Breis, J.T., Stevens, R.: A SQUaRE-based Quality Evaluation Framework for Ontologies. In: OntoQual 2010 - Workshop on Ontology Quality at the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010). CEUR Workshop Proceedings, Lisbon, Portugal, pp. 13–24 (2010) ISBN: ISSN 1613-0073

7. Fernández-López, M., Gómez-Pérez, A.: The integration of OntoClean in WebODE OntoClean method. In: Proceedings of the Evaluation of Ontology based Tools Workshop EON 2002 at 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002), Spain (2002)
8. Flemming, A.: Assessing the quality of a Linked Data source. Proposal, http://www2.informatik.hu-berlin.de/~flemming/Proposal.pdf
9. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Modelling Ontology Evaluation and Validation. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 140–154. Springer, Heidelberg (2006)
10. Gómez-Pérez, A.: Ontology Evaluation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. International Handbooks on Information Systems, pp. 251–274. Springer (2004)
11. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology.Technical Report SMI-2001-0880, Standford Medical Informatics (2001)
12. Pammer, V.: PhD Thesis: Automatic Support for Ontology Evaluation Review of Entailed Statements and Assertional Effects for OWL Ontologies. Engineering Sciences. Graz University of Technology
13. Poveda, M., Suárez-Figueroa, M.C., Gómez-Pérez, A.: A Double Classification of Common Pitfalls in Ontologies. In: OntoQual 2010 - Workshop on Ontology Quality at the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010). CEUR Workshop Proceedings, Lisbon, Portugal, pp. 1–12 (2010) ISBN: ISSN 1613-0073
14. Presutti, V., Gangemi, A., David, S., Aguado, G., Suárez-Figueroa, M.C., Montiel-Ponsoda, E., Poveda, M.: NeOn D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. NeOn project, FP6-27595 (2008)
15. Rector, A.L., Drummond, N., Horridge, M., Rogers, J.D., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 63–81. Springer, Heidelberg (2004)
16. Sabou, M., Fernandez, M.: Ontology (Network) Evaluation. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (eds.) Ontology Engineering in a Networked World, pp. 193–212. Springer (2012) ISBN 978-3-642-24793-4
17. Strasunskas, D., Tomassen, S.L.: The role of ontology in enhancing semantic searches: the EvOQS framework and its initial validation. Int. J. Knowledge and Learning 4(4), 398–414
18. Suárez-Figueroa, M.C.: PhD Thesis: NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse. Universidad Politécnica de Madrid, Spain (June 2010)
19. Welty, C., Guarino, N.: Supporting Ontological Analysis of Taxonomic Relationships. In: Data and Knowledge Engineering (September 2001)

Towards the Semantic Web – Incentivizing Semantic Annotation Creation Process

Szymon Łazaruk[1], Monika Kaczmarek[1], Jakub Dzikowski[1], Oksana Tokarchuk[2], and Witold Abramowicz[1]

[1] Department of Information Systems,
Faculty of Informatics and Electronic Economy,
Poznan University of Economics
al. Niepodleglosci 10, 61-875 Poznan, Poland
[2] School of Economics and Management
Free Univerisity of Bolzano
Piazzetta dell'Università, 1
39031 Brunico, Italy

Abstract. One of the main obstacles hampering the adoption of semantic technologies, is the lack of interest of users to create semantic content. In this paper, we focus on the incentives that may be applied and embedded within an application, in order to motivate users to create semantically annotated content. As an example, we show a semantically-enhanced Social Web Recommendation application, called *Taste It! Try It!*, integrated with a Linked Data source and Social Network. We also discuss the findings from the experiments run with 140 users.

Keywords: Linked data application, semantic annotations, semantic content creation tool.

1 Introduction

The Semantic Web is to enable machines to understand the meaning of information on the WWW by inserting machine-readable meta-data, i.e., semantic annotations, about the Web content and information on how they are related to each other [Berners-Lee et al., 2001]. A semantic annotation is machine processable, if it is explicit, formal, and unambiguous and this goal is usually reached by using ontologies [Uschold and Grüninger, 1996]. However, although the idea of the Semantic Web was coined in 2001, it has so far not been fully realized [Shadbolt et al., 2006] [Siorpaes and Simperl, 2010]. One reason for this is the lack of critical mass of the semantic content, i.e., real-world ontologies and RDF data. Although large efforts have been invested in order to automate the process of creating and managing the semantic content, gained experiences show that specific aspects of creating the semantic content still rely on a considerable amount of manual effort [Siorpaes and Simperl, 2010]. However, the interest of users to contribute to the creation of the semantic content is rather low, due to e.g., [Siorpaes and Simperl, 2010]:

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 282–291, 2012.

- a high barrier of entry - creation of semantic annotations requires specific skills and expertise on knowledge representation relevant topics;
- lack of incentives - most of the semantic applications are difficult to use and lack built-in incentives;
- lack of clear benefits - the benefits of using semantic content are often decoupled from the effort of creating the semantic content.

The above constitutes a motivation for the development, within the INSEMTIVES project[1], of a *Taste It! Try It!*– a semantically-enhanced Social Web Recommendation application, focusing on the creation of semantically annotated restaurants' reviews using concepts from DBpedia. The mentioned application is not only consuming the Linked Data (while creating the reviews), but also produces additional semantic annotations (about the reviewed entities) using either the mobile or WWW interface.

The goal of the paper is to show how faceted based approach along with a set of incentives mechanisms can support and encourage users to generate semantic annotations of reviews that may be used within recommendations systems. The *Taste It! Try It!* application has been equipped with the following mechanisms:

- Usability design related incentives – user-friendliness and easiness of creation of semantic annotation – addressing the high barrier of entry problem;
- Sociability design – integration of the application with the Facebook portal and usage of badges and points to award users for the activities – dealing with the lack of incentives;
- Additional functionalities being enabled by created semantic annotations – addressing the lack of clear benefits problem.

As *Taste It! Try It!* is a real-world application and during the performed experiments it has been used by 140 users, the embedded incentives mechanisms have been validated and their impact tested.

In order to meet the above mentioned goal, the paper is structured as follows. We start with a short summary of the related work and position our application towards the work of others. Then, the vision of the tool, along with its architecture is shortly presented. Next, we focus on the incentives mechanisms embedded in the system. Then, findings from the conducted experiments are shortly discussed. The paper concludes with final remarks.

2 Related Work

Semantic Web technologies have been introduced almost a decade ago, and yet, their real-life impact has been considerably limited for first few years.

The realization of the Semantic Web vision requires the availability of the critical mass of semantic annotations. As the fully automated creation of semantic annotations is an unsolved challenge, on the one hand, various Semantic Web annotations platforms providing various facilities as the storage of annotations and

[1] http://insemtives.eu

ontologies, user interfaces, access APIs were developed [Reeve and Han, 2005], and on the other, the researchers focused on the incentive schemas for the Semantic Web [Siorpaes and Hepp, 2008]. And thus, many researchers adopted Luis von Ahn's games with the purpose paradigm [Ahn, 2006] focusing on the challenges involved in building the Semantic Web that are easy for a human but hard for a computer. Data produced in the games are then analysed in order to create a Semantic Web representation such as RDF. One of the most prominent examples of games with the purpose in the Semantic Web settings is the OntoGame [Siorpaes and Hepp, 2007] – a project and series of games that aim at weaving the Semantic Web behind on-line, multi-player game scenarios, in order to create proper incentives for humans to contribute [Siorpaes and Hepp, 2007]. The first game developed was the OntoPronto. A more recent example is SeaFish [Thaler et al., 2011b] or SpotTheLink [Thaler et al., 2011a].

An example of a domain where engaging a number of users in the creation of semantic content is of utmost importance, is the area of recommendation and review systems. Recommender Systems (RS) are information search tools that have been proposed to cope with the information-overload problem, i.e, the typical state of a consumer, having too much information to make a decision [Adomavicius and Tuzhilin, 2005, Burke, 2007].

As on-line reviews are increasingly becoming the de-facto standard for measuring the quality of various goods and services, their sheer volume is rapidly increasing, thus processing and extracting all meaningful information manually in order to make an educated purchase becomes impossible [Blair-Goldensohn et al., 2008]. As a result, there has been a trend towards systems automatically summarizing opinions from a set of reviews and displaying them in an easy to process manner [Blair-Goldensohn et al., 2008] [Beineke et al., 2004] [Hu and Liu, 2004] as well as attempts to semantically annotate reviews. would allow a recommendation system to complete the incomplete information through inferences, automatically summarize the submitted reviews, dynamically contextualize user preferences and opinions, and finally, to contribute to a better user experience (personalized search and recommendation process) [Adomavicius and Tuzhilin, 2005] [Peis et al., 2008]. However, in order to do that, a critical mass of reviews (of semantically annotated reviews) needs to be reached.

In order to benefit from the network effects and positive externalities, end-user semantic content authoring technology needs to offer an appropriate set of incentives in order to stimulate and reward users' participation, generating in massive production of useful semantic content. When it comes to the conditions under which a person will actively contribute towards the completion of a task, the literature distinguishes between external (socially derived) and internal motivations (individually-based motivations) [Batson et al., 2002]; or intrinsic motivations that is directly connected to the act of performing a task, and extrinsic motivation that is, unrelated to the nature of the task such as a monetary recompense [Bock et al., 2005]. For instance, the Wikipedia relies on the following non-monetary incentives [Kuznetsov, 2006]: reciprocity – contributors receive a benefit in return; community – people contributing to Wikipedia feel needed

and have a sense of common purpose; reputation – people create their identities to be respected, trusted, and appreciated by peers; and finally – autonomy – contributing people enjoy the freedom of independent decisions.

In the last few years, researchers from both computer and social sciences have investigated the grounds of users motivations and human willingness to provided semantic content. For instance, when it comes to motivating users to contribute to the Semantic Web, [Siorpaes and Hepp, 2008] distinguish four dimensions: altruism – humans will contribute to the Semantic Web vision because it is the right thing to do, good for a world; monetary incentives – paying for contributions (e.g., using Mechanical Turk); immediate benefits – in order to use a search engine, a user needs to contribute to the annotations; non-monetary incentives – fun and pleasure, badges etc. Whereas [Cuel et al., 2011] distinguished nine categories encompassing: reciprocity and expectancy, reputation, competition, conformity to a group, altruism, self-esteem and learning, fun and personal enjoyment, implicit promise of future monetary rewards and finally, money. We follow this approach.

Following [Siorpaes and Hepp, 2008], 10 issues need to be addressed when hiding the semantic-content creation behind on-line games mechanisms: identifying tasks in semantic content creation; designing game scenarios, designing a usable and attractive interface, identifying reusable bodies of knowledge, preventing cheating, avoiding typical pitfalls, fostering user participation, deriving semantic data, efficient distribution of labour and finally scalability and performance. The *Taste It! Try It!* application may be treated to some extent as an example of such a game, and all of the above mentioned elements needed to be considered while designing the solution.

The *Taste It! Try It!* application benefits from the already developed semantic technologies and tools, and offers an added value through their integration and usage in order to, on the one hand, contribute to the Linked Data by producing semantic annotations, and on the other, to offer personalized advanced discovery and clustering possibilities. For the needs of the *Taste It! Try It!* application, a distinct disambiguation solution has been designed, adjusted to the specific needs of a mobile device. Whereas in most of the semantic games, the object to be annotated by users is known and the challenge is to reach a consensus on the annotations, in the *Taste It! Try It!* application, we go one step further, as we do not known which object is being reviewed by a user and once the disambiguation process is performed, still a consensus needs to be reached in case of contradictory information being reported. All of these features together with the applied incentives mechanisms, make the *Taste It! Try It!* application a distinct solution.

3 The Incentives Design within the Tool

Taste It! Try It! has been designed as a Web 2.0 application (integrated with the Facebook portal) supporting the creation of semantic annotations describing various places and locations. It is targeted at end-users, among which two groups may be distinguished: data producers (contributors) – users providing

reviews of places, i.e., people creating semantically annotated reviews, and data consumers (beneficiaries) – users interested in the content produced by the application, i.e., people looking for opinions about various places. Therefore, on the one hand, *Taste It! Try It!* enables data producers to contribute to a semantic content creation process using their mobile devices[2] or a WWW interface, and on the other, provides data consumers with personalized, semantic, context-aware recommendation process (i.e., offer a personalized semantic search mechanism).

Users of Taste It! Try It! express their opinions and create a review by providing values to selected features suggested by the application. These features may be roughly divided into basic and obligatory information such as: name of the place being reviewed;, type of location, GPS location (this information is required for the needs of disambiguation process); Menu information such as: cuisine type and best dishes/drinks served - values of those fields are suggested from DBpedia; and finally various other features such as e.g., atmosphere, evaluated in a quantitative manner by using star ratings.

The created review is uploaded to a *Taste It! Try It!* server and in the background, the semantic representation in the form of RDF triples is created. The *Taste It! Try It!* application is integrated with the INSEMTIVES platform – a tool created by the INSEMTIVES consortium[3] on top of the OWLIM semantic repository[4]. Among others it consists of a SPARQL endpoint, that is used to store and retrieve RDF triples. A part of semantic data is retrived and cached from DBpedia via the DBpedia SPARQL endpoint. Incentives models and methods within the *Taste It! Try It!* application may be grouped into three dimensions. The first one is the usability of the tool - e.g., ease of creation of the semantic data, background processing of semantics. The second is social aspect - keeping a user entertained (integration with the Facebook portal, awarding users with badges and points for different tasks). And finally, gaining additional benefits - by using an application a user is obtaining an access to a semantic and context-aware personalized recommendation process (by taking advantage of the semantic multi-layered clustering approach).

The first mentioned layer of incentives is related to the design of the application and its interface. This refers to: controllability (the user should be able to control the sequence of interactions with the tool); self-descriptiveness (the tool should be clearly understandable); error tolerance (the tool should allow the user to edit and re-edit the provided values); expectation conformity (the interactions logic is consistent throughout the entire application); suitability for task; individualization (some basic personalization mechanisms are envisioned such as changing the language, etc.).

Apart of the above mentioned aspects, while developing the application, our main motivation was to hide from users the complexity of semantics being the backbone of the application. Users do not interact directly with Semantic Web languages or technologies e.g., *SPARQL* endpoint. The semantic annotations

[2] The application is developed to work with the Android system.

[3] http://insemtives.eu/

[4] http://www.ontotext.com/owlim

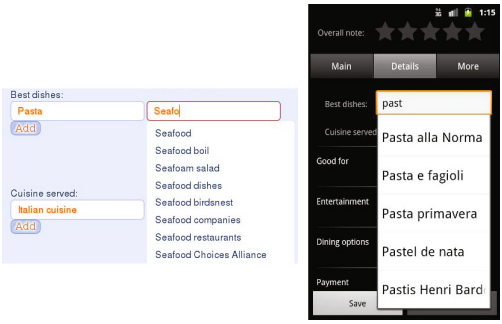


Fig. 1. Autocompletion mechanism

that are created are template-based annotations (faceted-based). Users while filling in some aspects of review (e.g., category of restaurant, type of cuisine, food and drinks served), are pointing to the concepts from DBpedia taking advantage of an auto-completion mechanism suggesting possible tags to be used (see figure 1). As we are following the faceted-based approach to the review creation, we can benefit from the additional knowledge in order to disambiguate and limit the potential tags (concepts from DBpedia) to be presented to users as an option to choose from. Therefore, for the needs of the *Taste It! Try It!* application, a disambiguation solution has been designed as follows – once a user starts to type in the first characters of the tag for the selected feature, the disambiguation takes place in the following steps:

- an appropriate SPARQL query is created limited to the branch of interest (e.g., in case a user is providing a tag to *the best dishes* feature, only the concepts related to Food and drinks are suggested),
- the obtained result is filtered using the preferred language of the user as well as the first characters typed in and as a result the list of concepts with their labels is retrieved from DBpedia,
- the proposed list of suggestions is sorted and presented to users.

The second main incentive mechanism belongs to the sociability design aspects and manifests itself through a set of non-monetary incentives that are frequently used in games, i.e., badges and points. Gathered points allow users to track their progress and measure achievements. Points also allow for social comparison between users (such as ranking of users, etc.). Badges show the status of a user as well as achievements.

The third dimension of the incentives addresses the problem of lack of clear benefits as it given users of an application an access to additional mechanisms encompassing personalized search mechanism.

Thus, in fact while searching, the personalized recommendation exploits both the knowledge base – information gathered by the *Taste It! Try It!* application and DBpedia (content-based approach), and the similarities between users (collaborative-filtering approach).

Fig. 2. *Taste It! Try It!*– user profile visualisation and friends ranking

4 Experiment

The Taste It! Try It! application has been evaluated running an experiment in December 2011 with approximately 140 participants. The aim of the experiment was twofold. On one hand, it was aimed at testing the functionalities of the tools and gathering semantic annotations. On the other hand, our intent was to assess the embedded in the tool usability and the sociability aspects. In particular, we wanted to measure the effect of different combinations of incentives.

From the incentive's point of view our interest was to check whether assigning badges or points has an effect on the general tendency to contribute to the application. Moreover, we wanted to check how information on relative standing of the user with regards to other community users (in terms of points and badges) affects their willingness to contribute to the application. While all users in the study knew the number of points that were assigned for each action, we differed information about points available to single users and assignment of badges. The application points mechanism was designed in the way to encourage users to create semantic annotations. In general, a user may obtain max of 100 application points for each review submitted – for each semantic concept added, a user was awarded with 6 points, whereas for each non-semantic feature only 1 to 2 points.

Group 1 (G1) was the baseline treatment and users in G1 had only information about the number of application points they earned. In Group 2 (G2), users earned application points and we provided information about the performance of the user who is just below or above him/her in the number of earned points (neighbourhood's performance). This information was presented without indicating the identity of these users. In Group 3 (G3), in addition to application points users could earn badges. In Group 4 (G4), users were awarded with points and badges and, in addition, information about the neighbourhoods performances (the one below or above her rate) was provided. In Group 5 (G5), users earned points and badges, and, in addition, information about the ratings of the whole group was provided. With this experiment we wanted to check whether information on the performance of peers would have an effect on the performance of users and whether badges serve as an incentive.

The design of the experiment was as follows. First, the participating students, who were presented with an option to obtain additional points from the lab part of classes, were granted an access to the application and were able to use their mobile and the Web application on Facebook. During the registration process, students were randomly assigned to one of five groups. During the experiment, an extensive logging procedure has been used, and all interactions with the application were stored. At the end of the testing period, all users were asked to answer a set of questions.

On average students participated in the experiment created 13 reviews and earned 846 application points. During the experiment, the participants created 2274 reviews on about 900 unique restaurants. The reviewed venue are located mainly in the Wielkpolska region of Poland. While filling in the review the participants used 5667 concepts from DBpedia to annotate cuisine and dishes features. As a result 14 840 RDF triples were created describing various restaurants.

Table 1 provides results of Man Whitney test of the equality of distributions of individual scores obtained by users for semantic annotations. The table shows that in G4 and G5 students provided significantly more semantic annotations compared to the baseline group G1. No other treatment group presented significant differences compared to the baseline treatment.

Table 1. Mann Whitney test statistics and associated probability on the score obtained from semantic annotations in treatment groups

	Group 2	**Group 3**	**Group 4**	**Group 5**
Group 1	Z=-1.540 (p=0.1235)	Z=-1.351 (p=0.1767)	**Z=-2.196 (p=0.0281)**	**Z=-3.206 (p=0.001)**
Group 2		Z=-1.540 (p=0.1235)	Z=0.322 (p=0.7556)	Z=-0.544 (p=0.5864)
Group 3			Z=-1.213 (p=0.2251)	Z=-0,631 (p=0.5275)
Group 4				Z=1.546 (p=0.1221)

The results in the table suggest that providing badges together with information on performance of peers stimulate performance of users. Although the incentives provided in the experiment were of general nature (points and badges), the structure of the incentives, mainly the fact that by providing semantic annotations users could get more points, influenced behavior of users with regards to semantic annotations.

5 Conclusions

Semantically structured information should lead to a better user experience (personalized and more precise search and other functionalities offered by IT tools). However, in order to achieve that, a critical mass of semantically structured data

needs to be created. The *Taste It! Try It!* application, presented in this paper, is a semantic content creation tool for mobile devices. It is to support users in the process of creation of semantically annotated reviews of various venues. It uses DBpedia as a source of data and is integrated with the Facebook portal. On the example of *Taste It! Try It!* application we show that non-monetary, easily built incentives can help to significantly improve creation of semantic annotations.

In the experiment we studied the effect of assignment of badges and the effect of the presentation of information on the willingness of users to add reviews and create semantic annotations. We find that the most effective tool for motivating users to contribute is a combination of badges and presentation of information on relative performance of peers. Both types of information about peers studied in the paper have an effect. The relative standing of the individual in the ranking of the group and information on users that are performing just above and just below them both have significant effect, if applied together with badges.

After performed evaluation and testing, we argue that the developed application is showing that it is possible to create a user friendly application for a mobile device producing semantically annotated reviews and make this process as far as possible invisible for users; and that the applied incentives mechanisms can ensure appropriate quality and quantity of information included in the reviews and encourage users to more active behaviour.

References

[Adomavicius and Tuzhilin, 2005] Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. on Knowl. and Data Eng. 17, 734–749 (2005)

[Ahn, 2006] Ahn, L.v.: Games with a purpose. Computer 39(6), 92–94 (2006)

[Batson et al., 2002] Batson, C.D., Ahmad, N., Tsang, J.: Four motives for community involvement. Journal of Social Issues 58(3), 429–445 (2002)

[Beineke et al., 2004] Beineke, P., Hastie, T., Manning, C., Vaithyanathan, S.: An exploration of sentiment summarization oes 3-page paper. In: Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text Theories and Applications (2004)

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American 284(5), 34–43 (2001)

[Blair-Goldensohn et al., 2008] Blair-Goldensohn, S., Neylon, T., Hannan, K., Reis, G.A., Mcdonald, R., Reynar, J.: Building a sentiment summarizer for local service reviews. In: NLP in the Information Explosion Era (2008)

[Bock et al., 2005] Bock, G., Zmud, R., Kim, Y., Lee, J.: Behavioral intention formation in knowledge sharing: examining the roles of extrinsic motivators, social-psychological forces, and organizational climate. MIS Quarterly 29, 87–111 (2005)

[Burke, 2007] Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007)

[Cuel et al., 2011] Cuel, R., Morozova, O., Rohde, M., Simperl, E., Siorpaes, K., Tokarchuk, O., Wiedenhoefer, T., Yetim, F., Zamarian, M.: Motivation mechanisms for participation in human-driven semantic content creation. Int. J. Knowl. Eng. Data Min. 1(4), 331–349 (2011)

[Hu and Liu, 2004] Hu, M., Liu, B.: Mining opinion features in customer reviews. In: Proceedings of the 19th National Conference on Artifical intelligence, AAAI 2004, pp. 755–760. AAAI Press (2004)

[Kuznetsov, 2006] Kuznetsov, S.: Motivations of contributors to wikipedia. SIGCAS Comput. Soc. 36(2) (2006)

[Peis et al., 2008] Peis, E., Morales-Del-Castillo, J.M., Delgado-López, J.A.: Semantic recommender systems - analysis of the state of the topic (2008)

[Reeve and Han, 2005] Reeve, L., Han, H.: Survey of semantic annotation platforms. In: Proceedings of the 2005 ACM Symposium on Applied Computing, SAC 2005, pp. 1634–1638. ACM, New York (2005)

[Shadbolt et al., 2006] Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems 21, 96–101 (2006)

[Siorpaes and Hepp, 2007] Siorpaes, K., Hepp, M.: OntoGame: Towards Overcoming the Incentive Bottleneck in Ontology Building. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2007, Part II. LNCS, vol. 4806, pp. 1222–1232. Springer, Heidelberg (2007)

[Siorpaes and Hepp, 2008] Siorpaes, K., Hepp, M.: Games with a purpose for the semantic web. IEEE Intelligent Systems 23(3), 50–60 (2008)

[Siorpaes and Simperl, 2010] Siorpaes, K., Simperl, E.: Human intelligence in the process of semantic content creation. World Wide Web 13(1), 33–59 (2010)

[Thaler et al., 2011a] Thaler, S., Simperl, E.P.B., Siorpaes, K.: Spotthelink: A game for ontology alignment. In: Maier, R. (ed.) Wissensmanagement. LNI, vol. 182, pp. 246–253. GI (2011a)

[Thaler et al., 2011b] Thaler, S., Siorpaes, K., Mear, D., Simperl, E., Goodman, C.: SeaFish: A Game for Collaborative and Visual Image Annotation and Interlinking. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 466–470. Springer, Heidelberg (2011b)

[Uschold and Grüninger, 1996] Uschold, M., Grüninger, M.: Ontologies: principles, methods, and applications. Knowledge Engineering Review 11(2), 93–155 (1996)

Evaluating Wiki-Enhanced Ontology Authoring

Chiara Di Francescomarino, Chiara Ghidini, and Marco Rospocher

FBK—IRST, Trento, Italy
{dfmchiara,ghidini,rospocher}@fbk.eu

Abstract. It is nowadays well-established that the construction of quality domain ontologies benefits by the involvement in the modelling process of more actors, possibly having different roles and skills. To be effective, the collaboration between these actors has to be fostered, enabling each of them to actively and readily participate to the development of the ontology, favouring as much as possible the direct involvement of the domain experts in the authoring activities. Recent works have shown that ontology modelling tools based on wikis' paradigm and technology could contribute in meeting these collaborative requirements.

This paper investigates the effectiveness of wiki-enhanced approaches for collaborative ontology authoring in supporting the team work carried out by domain experts and knowledge engineers.

1 Introduction

It is nowadays well-established that crafting ontologies has become a teamwork activity, as it requires a range of knowledge and skills hardly findable all together in a single person. For this reason collaborative aspects in ontology modelling have been widely investigated, and several works to support and enhance collaboration in this context have been presented (see e.g. [1,2,3,4]). The requirements and features that have emerged from these studies highlight the need to support collaboration in an articulated way: from supporting the collaboration between who understands the domain to be represented (the Domain Expert, or DE) and who has proper expertise in ontology modelling (the Knowledge Engineer, or KE), to supporting communication, discussion, and decision making between (geographically) distributed teams of ontology contributors.

The requirement of effectively involving domain experts, making them able not only to provide domain knowledge to knowledge engineers but also to directly *author* ontologies together with knowledge engineers, is increasingly recognised in a number of works (see e.g., [4,5,3]) as a crucial step to make the construction of ontologies more agile and apt to the needs of e.g., a business enterprise. The requirement of supporting the collaboration between the individuals of the whole modelling team, independently of their role, is similarly important and well recognised (see e.g., [1,2,3]). Indeed, it is not rare the situation where the modelling team is geographically distributed and/or users may not be able to participate to physical meetings. This requires to enable the awareness of the user on the evolution of the modelling artefacts, to support the coordination of the modelling effort within the team, as well as to favour the communication of the modelling choices and decisions made among the modelling actors.

These different collaborative modelling aspects may benefit from the availability of tools for ontology authoring based on *wikis*. Nowadays, wikis are among the most

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 292–301, 2012.

popular technologies for collaborative and distributed content authoring: people from all over the world can access simultaneously the same version of the content, and any change is immediately available to all the users. Although wikis were originally envisioned for editing of unstructured content, like text (e.g. Wikipedia), recent works ([6,7,8]) have shown their applicability for the (collaborative) authoring of structured content, including ontologies. Indeed, wikis offer a flexible, yet robust, platform, which can be exploited to favour the kind of collaboration needed for ontology authoring:

- wikis editing interfaces can be easily customized, even at user (or group of users) level, so to provide mechanisms to access and author the ontology compatibly to the skills and role of the team members. Thus, for example, simplified and guiding interfaces may be provided to DEs to edit (a limited part of) the ontology, while more powerful interfaces may be offered to KEs. This approach, called *multi-mode access* in [8], aims at promoting the participation of DEs to the authoring of the ontology, thus favouring a more balanced authoring contribution by DEs and KEs.
- wikis are by nature collaborative editing platforms. Therefore, functionalities for discussing ideas and choices, for commenting decisions, for notifying (and tracking) changes and revisions, are in the core-set of most wiki systems, or can be easily added to them. The advantages of having these functionalities tightly integrated in the modelling tool are many, from the possibilities to link discussions to specific entity of the ontology, to avoid users learning yet another tool.

In this paper we investigate the effectiveness of using wiki-enhanced tools to support collaborative ontology authoring. The investigation has been conducted using MoKi [8], a wiki-based ontology authoring tool employing multi-mode accesses to the ontological content for DEs and KEs, and described in Section 2. An empirical evaluation with real DEs and KEs has been performed according to the methodology proposed in [9] and described in Section 3. It had the aim of understanding more in detail whether wiki-enhanced collaborative modelling tools are effective in (i) making DEs more active in the authoring of ontologies, and (ii) supporting the collaboration during modelling. The results of this evaluation, illustrated in Section 4, show that wiki-enhanced tools are effective in supporting the active involvement of DEs in the modelling activities and in reducing the overall effort spent by team members in interacting. To the best of our knowledge, the evaluation of MoKi performed in this paper provides a first rigorous empirical evaluation of the support provided by wiki-based tools to favour collaboration in ontology modelling. Thus, we believe that the findings illustrated in this paper highlight the potential and criticality of using wiki-based modelling tools in collaborative ontology engineering.

2 MoKi

MoKi[1] is a collaborative MediaWiki-based [10] tool for modelling ontological and procedural knowledge in an integrated manner[2]. MoKi is grounded on three main pillars, which we briefly illustrate with the help of Figure 1:

[1] http://moki.fbk.eu

[2] Though MoKi allows to model both ontological and procedural knowledge, here we will limit our description only to the features for building ontologies.

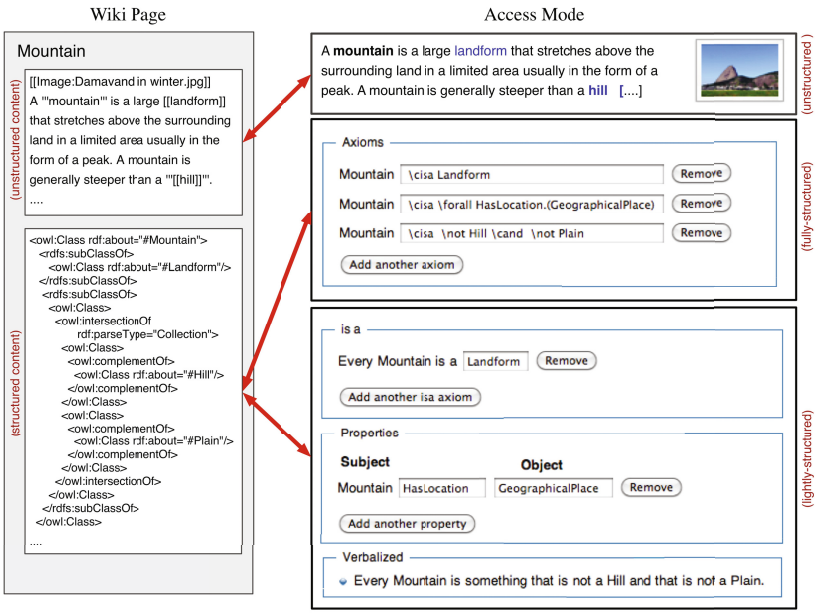


Fig. 1. A page and the access modes in MoKi

- each basic entity of the ontology (i.e., concepts, object and datatype properties, and individuals) is associated to a wiki page. For instance, the concept Mountain in Figure 1 is associated to a wiki page which contains its description;
- each wiki page describes an entity by means of both unstructured (e.g., free text, images) and structured (e.g. OWL axioms) content;
- a multi-mode access to the page content is provided to support easy usage by users with different skills and competencies. Figure 1 shows three different access modes, for accessing the unstructured and structured content of the wiki page.

A comprehensive description of MoKi is presented in [8]. In what follows, we focus only on the collaborative features that MoKi offers to the users of the modelling team.

Multi-mode access. Users can access the knowledge contained in each MoKi page using three different access modes, one for accessing the unstructured content (*unstructured access mode*), and two for accessing the structured content (*fully-structured* and *lightly-structured access mode*), as depicted in Figure 1:

- the unstructured access mode allows the user to edit/view the unstructured part of a MoKi page. Editing/viewing occurs in the standard MediaWiki way;
- the fully-structured access mode allows the user to edit/view the structured part of a MoKi page using the full OWL 2 expressivity[3] and is meant to be used by KEs to author the formal statements describing the entity associated to the wiki page;
- the lightly-structured access mode enables users to edit/view the content of the structured part of the MoKi page in a simplified way. This access mode, consists

[3] We adopt the syntax of *latex2owl*: https://dkm.fbk.eu/index.php/Latex2owl

of a form meant to be used by DEs, and contains statements that correspond to all the axioms in the fully-structured access mode. In the upper part the user can view and edit simple statements which can be easily converted to/from OWL statements. An example is the uppermost statement "Every Mountain is a Landform" in the lightly-structured access mode of Figure 1. The bottom part of the form provides a verbal description (automatically obtained via the OWL 2 Verbalizer [11]) of those OWL statements which cannot be intuitively translated/edited as simple statements in the upper part of the page. The purpose of this verbal description is to give the DE a flavour of the complex statements that the KE has formalized. If doubtful about some of the statements, the DE can mark them and ask for a clarification using e.g., the Discussion mechanism.

Discussion. MoKi exploits the MediaWiki discussion mechanism to enable DEs and KEs commenting on modelling choices, and debating on modelling options in order to converge to a shared formalization. Discussions are possible on single ontology entities or on (a part of) the whole model. Comments in the discussion pages are organized in threads, with details on the user and date/time associated to each comment.

Watchlist. The MediaWiki *watchlist* functionality allows MoKi users to be notified (with messages and email alerts) of changes performed on pages (and, thus, ontology entities) they are monitoring. Each user can autonomously manage his/her watchlist. By default, pages created/edited/discussed by a user are automatically added to the user's watchlist.

Notification. MoKi provides a notification mechanism to inform users about ontology changes that are relevant for them. Notifications are automatically sent in case changes to pages in the users' watchlist occur. Users can also send specific notifications, soliciting a confirmation or revision on some aspects of the ontology from particular users. Users receive notification by email, or the first time they connect back to MoKi.

History and Revision. Any change and comment added on a specific ontology entity is tracked in MoKi, thanks to the MediaWiki revision mechanism. Some specific functionalities are additionally provided (e.g. browsing last changes performed, newly created ontology entities, specific user contributions, recent/new discussions, most active users) to enable users being aware of the activities happening on the ontology.

3 The Empirical Evaluation: Experiment Design

This section presents the design of the empirical study carried out to evaluate the support provided by the MoKi collaborative authoring features to the process of ontology modelling, by following the methodology proposed by Wohlin [9] for the evaluation of software engineering experimentations[4].

[4] For lack of space we focus here on the main aspects of the evaluation that we performed. A more complete description can be found in [12].

3.1 Goal of the Study and Research Questions

In the study we investigate whether wiki-enhanced collaborative authoring tools *effectively* support the collaboration between DEs and KEs working together for the ontology construction, as well as whether the wiki-based collaborative features *have an impact* on the overall collaborative modelling process carried out by DEs and KEs.

The *goal* of the study is hence to consider two approaches (with and without the support of wiki-enhanced collaborative authoring features) with the purpose of evaluating the resulting processes of collaborative ontology construction.

The *quality focus* of the evaluation is related to the effectiveness of the support provided by the features to the collaborative modelling process. The research questions we are interested in investigating are hence the following:

Effectiveness of the wiki-enhanced collaborative features for ontology modelling

RQ1. Do the wiki-enhanced collaborative authoring features improve the involvement (and productivity) of DEs in editing activities?

RQ2. Do the wiki-enhanced collaborative authoring features reduce the effort required to team members to interact?

RQ3. Do the users perceive the support provided by the wiki-enhanced collaborative authoring features as effective?

RQ1 deals with the effectiveness of the wiki-enhanced collaborative authoring features in augmenting the involvement of DEs in the ontology authoring process, while **RQ2** regards the effectiveness of these features in reducing the effort required for the interaction among the team members, a key factor to enable the collaboration. Instead, **RQ3** deals with the subjective perception of the users about the effectiveness of the support provided by the wiki-enhanced collaborative authoring features. Though subjective, real users' opinion is absolutely relevant when evaluating features aiming at supporting them in their work.

For each of the above research questions, RQx, a null hypothesis, Hx_0, and an alternative hypothesis Hx are formulated. Hx_0 states a negative answer to RQx, while Hx states the positive answer to RQx. For example, if we consider $RQ2$ we have the null hypothesis $H2_0$ stating that "wiki-enhanced authoring features do not reduce the effort required to team members to interact" and the alternative hypothesis $H2$ stating that "wiki-enhanced authoring features reduce the effort required to team members to interact". The rejection of the null hypothesis implies the acceptance of the alternative one and hence the positive answer to the corresponding research question.

To answer the research questions, we asked 4 teams, composed of both DEs and KEs, to model an ontology describing a specific domain with and without the support of wiki-based collaborative authoring features. To this purpose, in the study, we used the MoKi tool. Indeed, besides the possibility to interact through chats and emails, team members were provided with two versions of the MoKi tool: with (CMoKi) and without (NCMoKi) the support of the wiki-enhanced authoring functionalities described in Section 2. The choice of relying on two different versions of the same tool, rather than comparing MoKi with a tool without collaborative authoring features, allowed us to

focus on the specific aspects we are interested to evaluate, avoiding the influence of additional factors on the results. We then compared and analysed the resulting processes and ontologies, by focusing on the above-mentioned aspects.

3.2 Subjects, Design and Material

The study involved 12 *subjects* (8 DEs and 4 KEs), organized in 4 teams (T_A, T_B, T_C and T_D), each including two DEs and one KE. In detail, the 8 DEs are pedagogists and psychologists employed in a publishing house[5] specialized in educational books, while the 4 KEs are experts in knowledge engineering working at FBK[6]. To match the background of the DEs, two domains from the pedagogical field were used for the experiment: one related to two cognitive abilities, namely *Attention and Concentration* (*AC*), and the other to motivational and emotional aspects of the learning process, namely *Motivation and Emotion* (*ME*).

Subjects carried out the ontology modelling tasks on the two domains in two different laboratory sessions, hereafter identified with *L1* and *L2*, held one in the morning and one in the afternoon, using NCMoKi and CMoKi (hereafter also called *treatments*) according to a balanced design [9], which allows to maximize the data, while controlling learning effects. Table 1 reports the details of the balanced schema we adopted in the study.

Table 1. Study balanced design

	$L1$ (morning session)		$L2$ (afternoon session)	
	NCMoKi	**CMoKi**	**NCMoKi**	**CMoKi**
T_A	*AC*			*ME*
T_B		*AC*	*ME*	
T_C	*ME*			*AC*
T_D		*ME*	*AC*	

Each team worked with *each* of the treatments and *each* of the domains exactly *once* (e.g., T_A formalized *AC* with NCMoKi and *ME* with CMoKi). The order in which treatments were used is balanced (e.g., NCMoKi is used in the morning session by T_A and T_C, and in the afternoon session by T_B and T_D; the contrary happens for CMoKi). This way, the learning effect that could arise due to the formalization of the same domain with both treatments and the effect of the order in which treatments are used by the teams are limited.

To carry out the experiment each participant was provided with: (i) a pre-questionnaire collecting information about her background and high-level competencies; (ii) an email account; (iii) a MoKi account; (iv) a final questionnaire to collect her subjective perception about the tool. Moreover, each DE received a description of the domain to be formalized, which, given the strict time constraints of the study, was intended to be used only as starting point to help DEs in focusing on the domain[7].

[5] Centro Edizioni Erickson (http://www.erickson.it/)

[6] www.fbk.eu

[7] Questionnaires provided to subjects are available at https://dkm.fbk.eu/index.php/MoKi_Evaluation_Resources

Before the experiment execution, subjects were trained on MoKi (both the NCMoKi and the CMoKi version) with a theoretical description, and a hands-on session to increase their familiarity with the tool. At the beginning of the first laboratory, subjects were asked to fill the pre-questionnaire and, at the end of the second laboratory, a final questionnaire. Each laboratory session was divided into the five phases reported in Table 2: **PH1**–**PH4** aiming to simulate the asynchronous collaboration between DEs

Table 2. Laboratory Division in Phases

Role	**PH1** ($\approx$ 25 min)	**PH2** ($\approx$ 25 min)	**PH3** ($\approx$ 15 min)	**PH4** ($\approx$ 15 min)	**PH5** ($\approx$ 40 min)
DEs	•		•		•
KEs		•		•	•

and KEs (i.e., either only DEs or only KEs were working on the ontology); and **PH5** investigating the process of synchronous collaborative modelling between them (i.e., both DEs and KEs were working simultaneously on the ontology).

3.3 Variables

In order to accept (or reject) the alternative hypothesis associated to each of the research questions listed in Subsection 3.1, we evaluate the effect of the wiki-enhanced collaborative features, that is of the treatment, on the factors investigated in the hypotheses (e.g., the effort required to team members to interact, in case of $H2$). Table 3 details the variables considered in the study to evaluate such an effect on each factor.

Table 3. Summary of the variables considered in the study

RQ	Alt. hp	Factor	Variable	Unit/Scale	Description
RQ1	$H1$	involvement (and productivity)	AXN	integer	average number of edited axioms
			$EDOPN$	integer	number of editing operations
RQ2	$H2$	effort to interact	$CONVL$	integer	conversation # of characters
RQ3	$H3$	users' subjective perception	$OEss$	$[0, 4]$	perceived effectiveness
			$FEoU$	$[0, 4]$	perceived ease of use
			$AWEss, CommEss, CoordEss$ $DMEss$	$[0, 4]$	perceived effectiveness for collaborative aspects (namely awareness, communication, coordination and decision making)

In detail, for evaluating the effectiveness of the wiki-enhanced collaborative authoring features in increasing the involvement of DEs in modelling activities (**RQ1**), we chose to look at both the modelling product (by considering the average number of axioms AXN of the final ontology edited by DEs) and at the modelling process (by computing the number of editing operations $EDOPN$ performed by DEs). We believe in fact that these two variables represent a good indicator of the active involvement of DEs in the ontology authoring. AXN is computed by manually inspecting the final ontology and taking into account all the concept, individual and property creation axioms, the "is-a" axioms, as well as more complex axioms e.g., restrictions or unions. $EDOPN$, instead, is computed by looking at the editing operations on ontology elements contained in the final ontology collected in the MediaWiki database.

To evaluate the effort required to team members to interact (**RQ2**), due to the difficulty in comparing the number of asynchronous messages (e.g., number of email or MoKi discussion messages) with the one of synchronous messages (e.g., number of chat

messages), we computed the overall length of exchanged messages (by e-mail, chat or discussion) in terms of number of characters ($CONVL$). In our opinion, this variable, though disregarding the effort required to *create* the message content, provides a reasonable approximation of the quantity and complexity of information exchanged for interacting.

RQ3 (i.e., the users' subjective perception) is evaluated by taking into account the subjects' perception about the overall effectiveness of the MoKi functionalities ($OEss$), their ease of use ($FEoU$), as well as the support they provide to each of the five levels of interaction investigated in the field of CSWC (Computer-Supported Cooperative Work), i.e., Awareness, Communication, Coordination, Decision Making and Community Building ($AWEss$, $CommEss$, $CoordEss$, $DMEss$ and $TBEss$). All the subjective evaluations, collected from the final questionnaire filled by the subjects, are expressed on a 5-point Likert scale.

4 The Empirical Evaluation: Experiment Results

In this section we describe the results of the statistical analyses performed on the collected data, including the objective ones related to the process and ontology analysis as well as the subjective users' evaluations. We analysed the influence of the treatments on each of the variables reported in Table 3. Since each of the subjects performed the task both with NCMoKi and with CMoKi, for the objective data (**RQ1** and **RQ2**) we applied a paired statistical test. Specifically, taking into account the type of data, we resorted to the paired Wilcoxon test [9]. Instead, for the analysis of the subjective evaluation (**RQ3**), we used a non-parametric test for independence, the Chi-squared [9]. All the analyses are performed with a level of confidence of 95% (p-value < 0.05), i.e., there is only a 5% of probability that the results are obtained by chance.

4.1 Data Analysis

RQ1. Table 4 (top) provides the descriptive statistics for the variables AXN and $EDOPN$, for both DEs and KEs. The table shows that the wiki-enhanced collaborative authoring features determine an increase in the number of ontology axioms in the final ontology formalized by DEs, as well as of the editing operations they performed: the number of axioms in the final ontology is on average 3.11 with NCMoKi and 4.43 with CMoKi, while the number of operations is on average 21.63 versus 29.13, respectively. In other terms, the CMoKi approach makes the DEs more active in authoring the ontology and hence also more productive in terms of number of axioms formalized. At

Table 4. Descriptive statistics and p-values related to hypotheses $H1$ and $H2$

Alt. hp	Role	Variable	Mean		Median		Std. Dev.		p-value
			NCMoKi	CMoKi	NCMoKi	CMoKi	NCMoKi	CMoKi	
$H1$	DEs	AXN	3.11	4.43	0.25	2	5.44	5.03	**0.03**
		$EDOPN$	21.63	29.13	27	30	9.29	16.17	**0.038**
	KEs	AXN	2.36	1.32	2.25	0.75	1.13	1.07	**0.025**
		$EDOPN$	31	11.75	18.5	14	13.98	5.12	**0.049**
$H2$	DEs & KEs	$CONVL$	3919.25	3319.92	3786.5	3372	1207.14	420.49	**0.046**

the same time, the table shows a reduction of the editing activities for KEs: the average number decreases from 31 to 11.75. A possible explanation for this result is a shift of the KEs work towards other types of activities, like formalization checks and DEs support (e.g. providing more fine-grained formalizations). These trends are statistically confirmed by the Wilcoxon test. Hence, we can reject the null hypothesis $H1_0$ and accept the alternative one $H1$: the collaborative wiki-based authoring features increase the involvement of DEs in the ontology authoring.

RQ2. Table 4 (last row) reports the descriptive statistics related to the variable $CONVL$, which we used for evaluating the effort spent by the members of the team in interaction. In case of the NCMoKi approach, the overall length of messages (in terms of number of characters) exchanged among team members is higher than in the CMoKi approach: on average 3919 characters with NCMoKi and 3319 with CMoKi. The result is statistically confirmed. We can hence reject $H2_0$ and confirm that the wiki-enhanced collaborative authoring features reduce the effort required by team members to interact (and hence collaborate).

RQ3. Table 5 reports the descriptive statistics of the subjective evaluation provided by users about the overall effectiveness of the collaborative authoring features, the average ease of use of the MoKi collaborative authoring functionalities and the effectiveness of these features in addressing the five CSWC interaction levels. Overall, the subjects provided a positive evaluation about all the different aspects related to the effectiveness of

Table 5. Descriptive statistics and p-values related to hypothesis $H3$

Variable	Investigated Factor	Negative	Positive	Median	p-value
$OEss$	overall effectiveness	0	8	3	**0.004678**
$FEoU$	ease of use	0	12	2.56	**0.000532**
$AWEss$	effectiveness for awareness	0	10	3	**0.001565**
$CmmEss$	effectiveness for communication	1	8	3	**0,01963**
$CoordEss$	effectiveness for coordination	1	5	2	0.1025
$DMEss$	effectiveness for decision making	0	9	3	**0.0027**
$TBEss$	effectiveness for team building	3	6	2.5	0.3173

the collaborative authoring features (the positive evaluations are always more than the negative ones, which, in turn, are almost always null). Except for coordination and team building, the positive evaluation of the users is also corroborated by statistical significance. Relying on the positive perception by users about the overall effectiveness and ease of use of the features, as well as on their positive evaluation about the effectiveness in supporting three out of five interaction aspects, we can reject the null hypothesis $H3_0$ and confirm the alternative one $H3$.

5 Conclusion

The paper, through a rigorous empirical evaluation, shows that wiki-enhanced tools for ontology authoring, by actively involving domain experts in the authoring process and supporting the interaction of modellers with other team members, effectively support the process of collaborative ontology authoring. Starting from results and feedbacks

obtained in this analysis, we plan, in the future, to further investigate how the support provided by wiki-based authoring features can be improved for specific interaction levels (e.g., decision making), as well as how users can be guided (e.g., by means of good practices) in the process of collaborative modelling so as to improve both the effective collaboration and the resulting ontology.

Acknowledgement. We thank Dr. G. Vaschetto and Dr. S. Cramerotti for their support in arranging the evaluation, as well as the members of FBK and Edizioni Erickson who took part to the experiment for their availability and the valuable feedback provided.

References

1. Palma, R., Corcho, O., Gómez-Pérez, A., Haase, P.: A holistic approach to collaborative ontology development based on change management. Web Semantics: Science, Services and Agents on the World Wide Web 9(3) (2011)
2. Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D.: OntoEdit: Collaborative Ontology Development for the Semantic Web. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 221–235. Springer, Heidelberg (2002)
3. Tudorache, T., Falconer, S., Noy, N.F., Nyulas, C., Üstün, T.B., Storey, M.-A., Musen, M.A.: Ontology Development for the Masses: Creating ICD-11 in WebProtégé. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 74–89. Springer, Heidelberg (2010)
4. Dimitrova, V., Denaux, R., Hart, G., Dolbear, C., Holt, I., Cohn, A.G.: Involving Domain Experts in Authoring OWL Ontologies. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 1–16. Springer, Heidelberg (2008)
5. Bagni, D., Cappella, M., Pazienza, M.T., Stellato, A.: CONGAS: A *CO*llaborative Ontology Development Framework Based on *N*amed *GrAphS*. In: Serra, R., Cucchiara, R. (eds.) AI*IA 2009. LNCS, vol. 5883, pp. 42–51. Springer, Heidelberg (2009)
6. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic wikipedia. Journal of Web Semantics 5, 251–261 (2007)
7. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
8. Ghidini, C., Rospocher, M., Serafini, L.: Conceptual modeling in wikis: a reference architecture and a tool. In: The Fourth International Conference on Information, Process, and Knowledge Management (eKNOW 2012), Valencia, Spain, pp. 128–135 (2012)
9. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering: an introduction. Kluwer Academic Publishers (2000)
10. Wikimedia Foundation: Mediawiki, http://www.mediawiki.org (accessed on November 6, 2011)
11. Kaljurand, K., Fuchs, N.E.: Verbalizing owl in attempto controlled english. In: Proceedings of Third International Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, vol. 258 (2007)
12. Ghidini, C., Rospocher, M., Di Francescomarino, C.: Theoretical and empirical evaluation of wiki-enhanced ontology authoring. Technical report, FBK-IRST, Italy (2012), https://dkm.fbk.eu/index.php/MoKi_Evaluation_Resources

SlideWiki: Elicitation and Sharing of Corporate Knowledge Using Presentations

Ali Khalili[1], Sören Auer[2], Darya Tarasowa[1], and Ivan Ermilov[1]

[1] Universität Leipzig, IfI/BIS/AKSW, Johannisgasse 26, 04103 Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org
[2] Technische Universität Chemnitz, Informatik/ISST, D-09107 Chemnitz, Germany
soeren.auer@informatik.tu-chemnitz.de

Abstract. Presentations play a crucial role in knowledge management within organizations, in particular to facilitate organizational learning and innovation. Much of the corporate strategy, direction and accumulated knowledge within organizations is encapsulated in presentations. In this paper, we investigate the limitations of current presentation tools for semi-structured knowledge representation and sharing within organizations. We address challenges such as collaborative creation of presentations, tracking changes within them, sharing and reusing existing presentations. Then we present SlideWiki as a crowd-sourcing platform for the elicitation and sharing of corporate knowledge using presentations. With SlideWiki users can author, collaborate and arrange slides in organizational presentations by employing Web 2.0 strategies. Presentations can be organized hierarchically, so as to structure them reasonably according to their content. According to the wiki paradigm, all content in SlideWiki (i.e. slides, decks, themes, diagrams) are versioned and users can fork and merge presentations the same way as modern social coding platforms allow. Moreover, SlideWiki supports social networking activities such as following and discussing presentations for effective knowledge management. The article also comprises an evaluation of our SlideWiki implementation involving real users.

1 Introduction

In medium and large enterprises and organizations presentations are a crucial element of the corporate knowledge exchange. Such organizations are mostly hierarchically organized and communication and knowledge flows usually accompany corporate hierarchies. In addition to sending emails and documents, meetings where presentations are shown to co-workers, subordinates and superiors are one of the most important knowledge exchange functions. Research conducted by the Annenberg School of Communications at UCLA and the University of Minnesota's Training & Development Research Center show that executives on average spend 40-50% of their working hours in meetings.[1] They spend a remarkable amount of time collecting their required materials and creating new presentations.

[1] http://www.shirleyfinelee.com/MgmtStats

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 302–316, 2012.

The challenges with current organizational presentations can be roughly divided into the following categories:

- *Sharing and reuse of presentations.* Much of the corporate strategy, direction and accumulated knowledge is encapsulated in presentation files; yet this knowledge is effectively lost because slides are inaccessible and rarely shared. Furthermore offline presentations are hard to locate. Thereby executives usually spend their time creating new slides instead of re-using existing material.
- *Collaborative creation of presentations.* Executives in different departments or countries often unknowingly duplicate their efforts, wasting time and money. To collaboratively create a presentation, the members need to manually download and merge the presentations.
- *Following/discussing presentations.* Finding the most up-to-date presentation is difficult and time-consuming, therefore costly. Furthermore, discussing the content of presentations in face-to-face meetings or email discussions is not efficient within organizations.
- *Tracking/handling changes in presentations.* Tracking and handling changes that occur within different presentations is a time-consuming task which needs opening all offline presentations and manually comparing their content. Additionally there are hundreds of slide copies to change when an original is modified. This cascading change costs a fortune each time.
- *Handling heterogeneous presentation formats.* Presentations can be created in different formats (e.g. Office Open XML, Flash-based, HTML-based or LaTeX-based presentations) thereby integration and reuse of them will be a cumbersome task for organization members.
- *Ineffective skills management and training.* Medium and large enterprises are obliged by law to provide means for training and qualification to their employees. This is usually performed by seminars, where training material is prepared in the form of presentations. However, it is usually not possible to provide engaging bi-directional and interactive means of knowledge exchange, where employees contribute to the training material.
- *Preserving organization identity.* Having a consistent template and theme including the logo and brand message of organization is of great significance in shaping the organization identity. With offline presentations it is difficult to persistently manage and sustain specific organization templates. Everyone needs to take care of templates and themes individually and managing the changes takes a remarkable amount of time.

In this paper, we investigate the above mentioned limitations of current presentation tools for semi-structured knowledge representation and sharing within organizations. We present an application called *SlideWiki* as a crowdsourcing platform for the elicitation and sharing of knowledge using presentations. With SlideWiki users can author, collaborate and arrange slides in organizational presentations by employing Web 2.0 strategies. Presentations can be organized hierarchically, so as to structure them reasonably according to their content. According to the wiki paradigm, all content in SlideWiki (i.e. slides, decks, themes,

diagrams) are versioned and users can fork and merge presentations the same way as modern social coding platforms (e.g. Github, Bitbucket) allow. Moreover, SlideWiki supports social networking activities such as following and discussing presentations for effective knowledge management.

This article is structured as follows: We first discuss the applicability of presentations as a tool for knowledge management in Section 2. In Section 3 we describe SlideWiki concept for elicitation and sharing of corporate knowledge. We discuss our implementation including crucial functionality such as authoring, versioning, Linked Data interface and search in Section 4. An evaluation using synthetic benchmarking as well as involving real users is provided in Section 5. We review related work in Section 6 and conclude with an outlook on future work in Section 7.

2 Presentation as a Tool for Knowledge Management

Presentations play a crucial role in knowledge management within organizations, in particular to facilitate organizational learning and innovation. Presentations are one of the commonly used communication channels through which organization members elicit and share their knowledge. They typically have a different purpose than normal written documents. Presentations transfer fewer words per minute in the verbal channel, but also convey nonverbal information known for its impact on credibility [8]. They do not focus on the detailed information (the *what*), but what this information means to the audience in view of the presentation's purpose (the *so what*). As a tool for knowledge management within organizations, presentations can be applied to the following areas:

Developing a shared mental model within organization. In organizational learning, learning occurs through shared insights and mental models. In this process, organizations obtain the knowledge that is located in the minds of their members or in the epistemological artifacts (maps, memories, policies, strategies and programs) and integrates it with the organizational environment [19]. This shared mental model (a.k.a. *organizational memory*) is the accumulated body of data, information, and knowledge created in the course of an individual organization's existence.

Combining presentations with social approaches for crowdsourcing. Presentations when combined with *crowdsourcing* and *collaborative social approaches* can help organizations to cultivate innovation by collecting and expressing the individual's ideas within organizational social structures. As discussed in [4], there are different types of social structures living in the context of organizations. Work groups, project teams, strategic communities, learning communities, communities of practice, informal networks, etc. to mention some. These social structures make use of presentations frequently to present and discuss their internal ideas. Therefore, creating an integrated collaborative platform for authoring and sharing presentations will result in exchanging knowledge within and cross these social structures (even supporting inter-organizational knowledge transfer).

As a driver for organizational innovation. Presentations are an important driver of organizational innovation particularly when they are exchanged between social connections that cross functional and organizational boundaries. As discussed in [9], improvising is a structured process of innovation that involves responding to changing situations with resources at hand by creating a production and adapting it continuously. Presentation tools enable the creation of so called *Structural Referents* – a representation one develops about a structure. Structural referents support the communities to collaborate on individual's ideas and foster the potential ideas in alignment with the organizational goals. *Ghost Sliding* is a process introduced in [9] which utilizes presentation slides as structural referents for collaborative knowledge management. Ghost sliding is an iterative process where consultants draw up quick, rough representations of each slide and discuss them with clients to develop consensus on what statements are going to be included in the final presentation and what data needs to be collected to support those statements. The rationale for ghost-sliding is that by developing explicit representations of what a consultant is striving for, the consultant could discuss the hypotheses with others and be more efficient about what kind of data to look for.

As a media for knowledge exchange and training. As reported in [7], PowerPoint presentations are the most used (75.4 %) for developing e-learning content within organizations. Presentations contain visualized learning materials which improve the training of organization members having different levels of knowledge. Enabling users to contribute to this training materials makes it possible to provide engaging bi-directional and interactive means of knowledge exchange.

3 SlideWiki Concept

SlideWiki is a crowdsourcing platform for elicitation and sharing of corporate knowledge using presentations. It exploits the wisdom, creativity and productivity of the crowd for the collaborative creation of structured presentations. Figure 1 shows the SlideWiki ecosystem for supporting organizational knowledge management. SlideWiki provides a collaborative environment to resolve the challenges discussed in Section 1. It enables knowledge communities to contribute to dynamic parts of organizational memory which is encapsulated in presentations. The dynamic view of the structure of organizational memory [6] takes into account the social nature of memory. Rather than viewing memory as knowledge stored in a collection of retention bins, the emphasis is on memory as continually constructed and reconstructed by humans interacting with each other and their organizational environment.

In SlideWiki, users from different knowledge communities crossing the organization and functional boundaries can collaboratively create structured online presentations. Users can assign tags and categories for structuring the presentations. The created presentations can be shared and reused to build new synergetic presentations. Users can also track and manage changes occurring within presentations using a revisioning system. Additionally, SlideWiki includes an e-learning

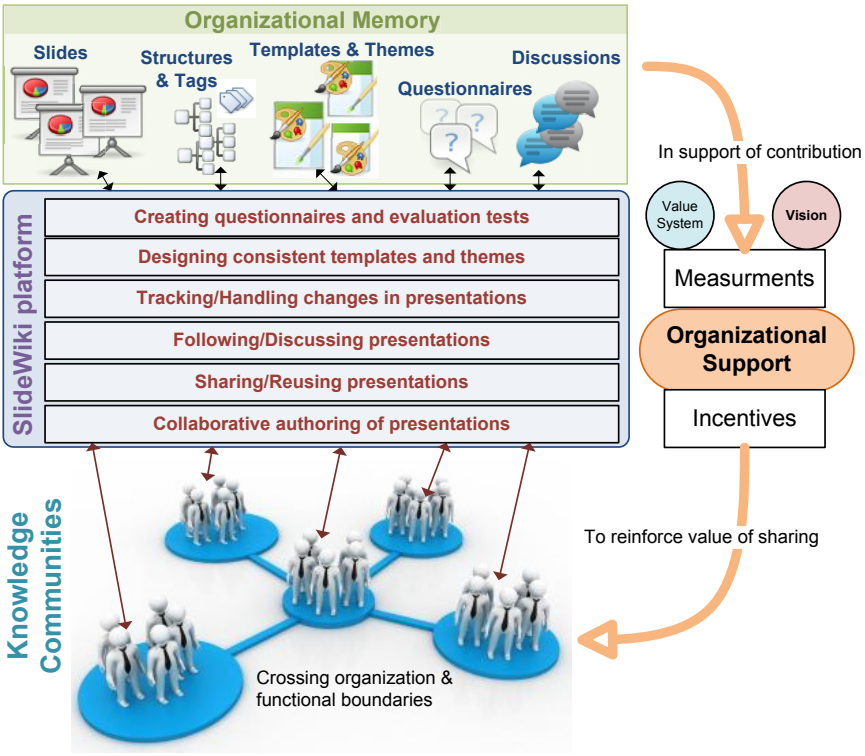


Fig. 1. SlideWiki ecosystem for organizational knowledge sharing

component which deals with questionnaires created for each presentation slide. Questionnaires together with the evaluation tests facilitate the training of users within organizations. With regard to preserving the organization identity and branding, SlideWiki supports creating and sharing of templates and themes. Apart from the contribution on authoring of presentation content, SlideWiki also supports social networking activities such as following presentation decks, slides and users as well as discussing the created content.

In order to effectively integrate and employ SlideWiki within an organization, having an organizational evaluation and support system seems to be crucial. Measuring the quality of user contributions as well as providing some incentives to reinforce the value of user contributions are required elements of a comprehensive knowledge management framework [10]. Organizations need to frequently monitor the organizational memory created by presentations in order to link and align the user contributions with the organization's goals, visions, management, value system, and infrastructure.

SlideWiki as a social Web 2.0 software for organizational knowledge management is developed according to the PLANT SEEDS Framework [5]. The PLANT SEEDS framework is a set of high-level design considerations targeted at enhancing social computing success by increasing the chances of rapidly

growing productive and self-sustaining communities that deliver business value. The PLANT SEEDS framework consists of 10 design considerations from which it gets its name: *Purpose*, *Liberty*, *Authorship*, *Nurturing*, *Tipping Point*, *Structure*, *Ease of Use*, *Ecosystem*, *Discoverability*, *Seeding*.

The first five design considerations (PLANT) are associated with defining and growing the community, and facilitating productive and valuable participation. The second five considerations (SEEDS) address the corresponding information system's design. In the sequel, we briefly describe each aspect of the PLANT SEEDS framework and discuss them in relation to our SlideWiki implementation.

Purpose. Purpose is the foundational design consideration. The purpose is the cause around which the community will rally. It is the "What's in it for me?" that will motivate the community to participate. SlideWiki helps users to manage their knowledge using presentations. For this purpose, it enables users to share, reuse and collaboratively create interactive online presentations.

Liberty. Liberty is about social participation governance – that is, how to guide participant behaviors toward productive interactions and away from undesirable behaviors. As shown in Figure 1, organizational leadership and support should be considered when implementing SlideWiki system within an organization.

Authorship. Promoting authorship is a critical aspect of an enterprise social solution's success. SlideWiki follows the wiki paradigm hence allowing all users to write and edit presentations. The SlideWiki implementation complies with our WikiApp development model [2] for engineering of domain-specific wiki applications.

Nurturing. Community productivity depends on sharing and reuse. When deploying SlideWiki, leadership involvement is critical. Leaders must not only care, they must also participate visibly in the collaborative creation of presentations.

Tipping Point. The tipping point is the critical mass level of community participation required for the system to experience viral growth, achieve the network effect and become self-sustaining. In order to reach this point when deploying SlideWiki, a set of initial presentation content and early adopters should be provided by the organization. These initial content act as examples for other users to help them contribute to system.

Structure. To minimize the curve to productivity, SlideWiki employs a set of structures that facilitate participation relevant to the related purpose. Structures involved predefine themes and transitions as well as slide templates which can be edited and reused by users.

Ease of Use. Ease of use is a common mantra for all applications, but its importance for social solution adoption is uniquely grand. SlideWiki utilizes several modern and user-friendly interfaces to increase the usability of system. Inline content authoring UI, progressive Ajax-based load of presentations are two examples (c.f Section 4).

Ecosystem. An ecosystem is the converse of a social island, where the social solution is tied into existing enterprise systems and work practices. Since SlideWiki supports importing/exporting different file formats (PowerPoint, PDF, HTML, SVG, etc.), it can be easily integrated into existing enterprise systems thereby providing a high level of interoperability.

Discoverability. The social computing tenets of "architecture of participation" and everyone's a potential author is leading to an explosion of content on the public Web. This is also happening in enterprises. This explosion creates a content overload challenge that demands a new and more-comprehensive approach to information discoverability. Better discoverability is required to deliver a good user experience. SlideWiki's search features enables participants to view a set of results that may address their needs or provide an entry point for further discoverability. It also includes the ability to organize or reduce often huge search result sets, with capabilities like sorting and filtering, for more-efficient discoverability. Furthermore, by supporting social feedback technologies – such as rating, ranking, voting, investing, commentary, badging and tagging – SlideWiki can substantially add value to discoverability. SlideWiki also employs *subscription* technology, such as email feedback, RSS and Atom, which allows participants to determine what information they want to be alerted to or have pushed to them.

Seeding. The critical aspect here is that, prior to the tipping point, the social solution environment must contain enough base content and initial participation to encourage participant active content contribution and to catalyze the community to critical mass. By providing import from other presentation formats, organizations can easily convert and inject a huge amount of their previously created content into SlideWiki system.

4 SlideWiki Implementation

The SlideWiki application makes extensive use of the model-view-controller (MVC) architecture pattern. The MVC architecture enables the decoupling of the user interface, program logic and database controllers and thus allows developers to maintain each of these components separately. As shown in Figure 2, the implementation comprises the main components authoring, change management, import/export, frontend, social networking, linked data interface, search, e-learning and styling. We briefly walk-through these components in the sequel.

Authoring. SlideWiki employs an inline HTML5 based WYSIWYG (What-You-See-Is-What-You-Get) text editor for authoring the presentation slides. Using this approach, users will see the slideshow output at the same time as they are authoring their slides. The editor is implemented based on ALOHA editor[2] extended with some additional features such as image manager, source manager, equation editor. The inline editor uses SVG images for drawing shapes on slide

[2] http://aloha-editor.org/

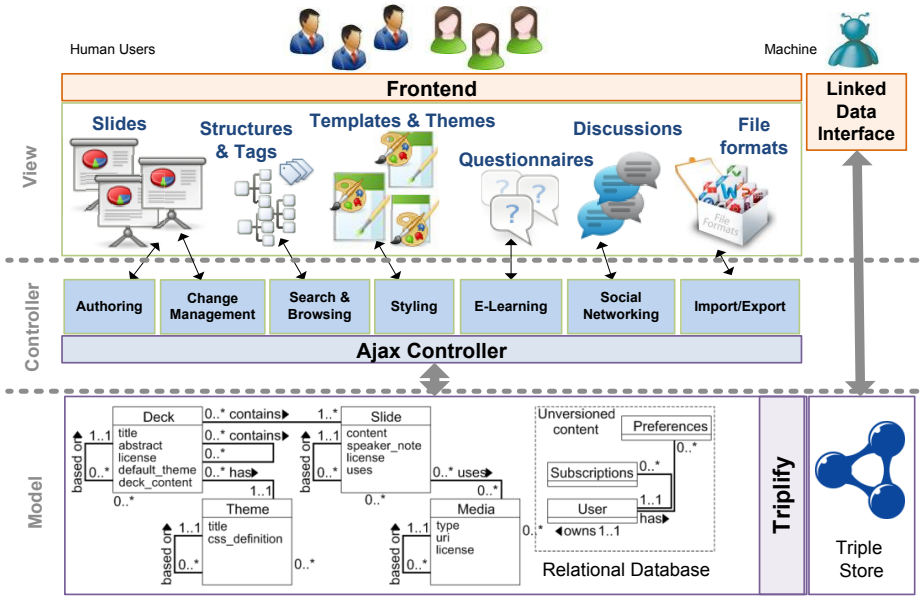


Fig. 2. Bird's eye view on the SlideWiki MVC architecture

canvas. Editing SVG images is supported by SVG-edit[3] with some predefined shapes which are commonly used in presentations. For logical structuring of presentations, SlideWiki utilizes a tree structures in which users can append new or existing slides/decks and drag & drop items for positioning. When creating presentation decks, users can assign appropriate tags as well as footer text, default theme/transition, abstract and additional meta-data to the deck.

SlideWiki follows the "anyone can edit" philosophy of Wiki for creating synergistic presentations. In order to manage the changes made by users, SlideWiki defines a revisioning system as described in the following.

Change Management. As shown in Figure 4, there are different circumstances in SlideWiki for which new slide or deck revisions have to be created. For decks, however, the situation is slightly more complicated, since we wanted to avoid an uncontrolled proliferation of deck revisions. This would, however, happen due to the fact, that every change of a slide would also trigger the creation of a new deck revision for all the decks the slide is a part of. Hence, we follow a more retentive strategy. We identified three situations which have to cause the creation of new revisions:

- The user specifically requests to create a new deck revision.
- The content of a deck is modified (e.g. slide order is changed, change in slides content, adding or deleting slides to/from the deck, replacing a deck content with new content, etc.) by a user which is neither the owner of a deck nor a member of the deck's editor group.

[3] http://code.google.com/p/svg-edit/

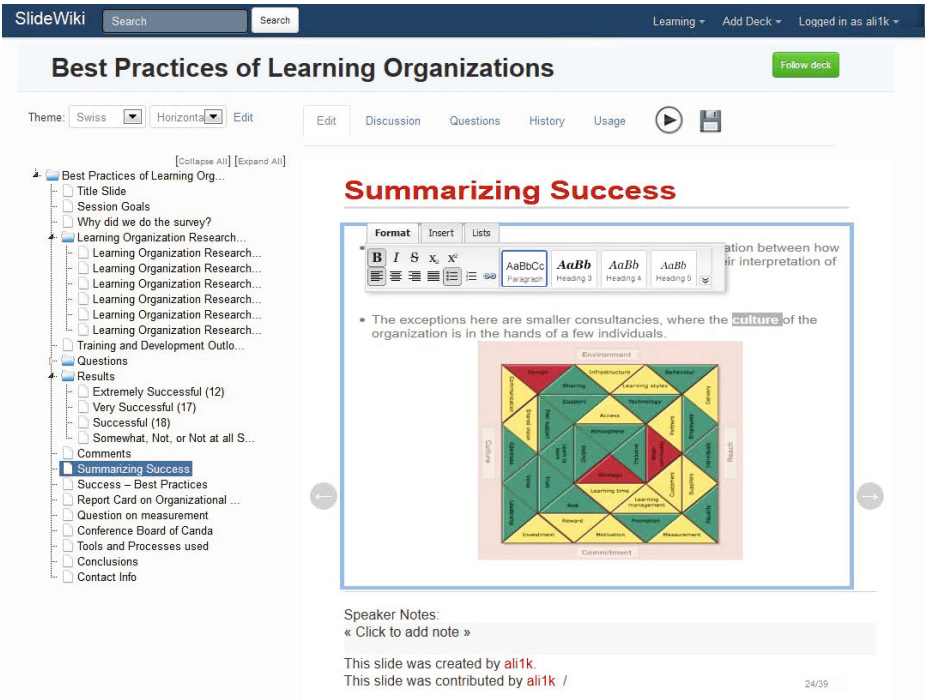


Fig. 3. The screenshot of the SlideWiki application: Inline authoring of presentations

– The content of a deck is modified by the owner of a deck but the deck is used somewhere else.

In addition, when creating a new deck revision, we always need to recursively spread the change into the parent decks and create new revisions for them if necessary.

Import/Export. A crucial feature of SlideWiki is an ability to import and export data into different formats. Without the possibility to import and export data (i.e. making backups and transferring data to other data mediums or applications) a user will be discouraged from contributing and maintaining data on the platform. The main data format used in SlideWiki is HTML. However, there are other popular presentation formats commonly used by desktop application users, such as Office Open XML (ECMA-376 or ISO/IEC 29500) and LaTeX. Thus we implemented importing and exporting functionality for above-mentioned formats. Currently, SlideWiki supports importing from Microsoft Office implementation of ECMA-376 format (i.e. files with .pptx extension) and exporting to the deck.js[4] HTML+JS format. LaTeX and OpenOffice ECMA-376 implementation support are prepared but not yet completed.

[4] http://imakewebthings.github.com/deck.js/

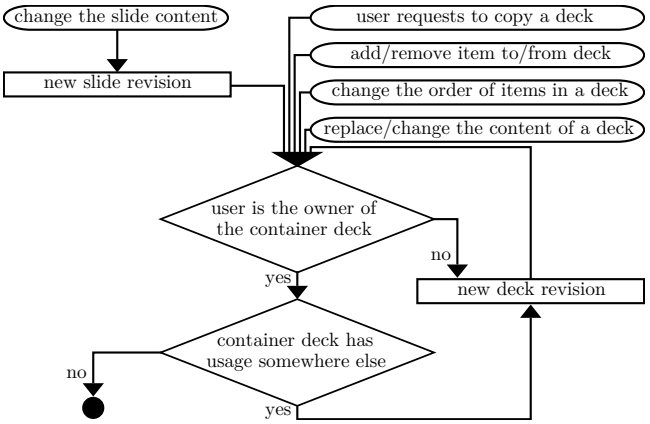


Fig. 4. Decision flow during the creation of new slide and deck revisions

Frontend. SlideWiki makes extensive use of HTML5 features to provide users with intuitive and responsive interfaces. In addition to the overall MVC architecture, SlideWiki utilizes a *client-side MVC* approach (implemented in JavaScript and running inside the users Web browser). The client-side MVC handler as (singleton) controller listens to the hash fragment of the requested URLs and once a change has occurred the handler triggers the corresponding actions. Each action has a JavaScript template (implemented using *jQuery templates*) with the corresponding variable place holders. For each action an Ajax call is made and the results are returned to the controller in JSON format. Subsequently, the controller fills the templates with the results and renders them in the browser.

Additionally, SlideWiki supports *progressive loading* of presentations to guarantee the scalability when a large presentation is loaded. Progressive loading is a design pattern for web applications which adds content to a web page incrementally. It results in gradually increasing the workload over time when loading a large presentation thereby improving the performance of the system.

Social Networking. As a social enterprise software, SlideWiki supports different types of social networking activities. Users can follow items such as decks, slides and other users. They can also rate, tag and discuss decks and slides. Content syndication in multiple formats such as RSS, ATOM, OPML and JSON is provided for created items so that users can subscribe to them. We are currently integrating SlideWiki with popular social networking sites like Twitter, Facebook, GooglePlus and LinkedIn.

E-Learning. SlideWiki supports the creation of questionnaires and self-assessment tests from presentation slides. Questionnaires (like decks and slides) are supported by a revisioning system so that users can create, edit and share questionnaires according to the Wiki collaboration style. The created questionnaires can be employed for training organization members or to interview prospective members. Educators can manually or automatically generate

evaluation tests from the questionnaires based on their preferences to assess the knowledge of employees to be trained.

Linked Data Interface. SlideWiki implementations can be easily equipped with a Linked Data interface. We employed the RDB2RDF mapping tool *Triplify* [1] to map SlideWiki content to RDF and publish the resulting data on the Data Web. Triplify is based on mapping HTTP-URI requests onto relational database queries. It transforms the resulting relations into RDF statements and publishes the data on the Web in various RDF serializations, in particular as Linked Data. Triplify neither defines nor requires to use a new mapping language, but exploits and extends certain SQL notions with suitable conventions for transforming database query results (or views) into RDF and Linked Data. The Triplify configuration for SlideWiki was created manually. The SlideWiki Triplify Linked Data interface is available via: `http://slidewiki.aksw.org/triplify`.

Search and Browsing. There are three ways of searching in SlideWiki: by *keywords*, by *metadata* and by *user* (who contributed or follows certain content). We combined keywords and tag search so that users can either 1. search by keywords and then add a tag filter, or 2. show all slides or decks having the tag and then running an additional keyword search on the results. In both cases an ordering a user might have applied is preserved for subsequent searches. In addition to the deck tree user interface for browsing the presentations, a breadcrumb navigation bar is implemented in SlideWiki. Breadcrumb improves the accessibility of system by increasing the user awareness when browsing nested presentations.

Styling. In order to create flexible and dynamic templates and styles for presentations, SlideWiki utilizes *Saas* (Syntactically Awesome Stylesheets) language[5]. Sass extends CSS by providing several mechanisms available in programming languages, particularly object-oriented languages, but not available in CSS3 itself. When Sass script is interpreted, it creates blocks of CSS rules for various selectors as defined by the Sass file. Using Saas, SlideWiki users can easily create and reuse presentation themes and transitions.

5 Evaluation

The SlideWiki concept was evaluated in several ways: Firstly, as a proof-of-concept we developed a comprehensive implementation, which is available at: `http://slidewiki.aksw.org`. The SlideWiki platform is currently used for accompanying an information systems lecture with more than 80 students. We performed a preliminary usability study which is described in the sequel.

Determine whether we succeeded to effectively hide SlideWiki's data model complexity, we performed a usability user study with 13 subjects. Subjects were drawn from the members of AKSW research group, the computer science department at the university of Leipzig. We first showed them a tutorial video

[5] `http://sass-lang.com/`

Table 1. SlideWiki feature usage per user

Feature	**Usage**
WYSIWYG slide authoring	76.92%
Importing pptx presentations	46.15%
Using LaTeX in slide content	58.85%
Using/Creating themes for decks	46.15%
Searching slides/decks	76.92%
Adding existing slides/decks to your deck	69.23%
Following slides/decks/users	53.85%
Contributing to other's slides/decks	53.85%
Using other revisions of slides/decks	76.92%
Playing slides/decks	**92.31**%
Downloading the decks for offline preview	**38.46**%
Adding comments about slides/decks	61.54%

of using different features of SlideWiki then asked each one to create a presentation with SlideWiki. After finishing the task, we asked the participants to fill out a questionnaire which consisted of three parts: demographic questions, feature usage questions and usability experience questions. Table 1 summarizes the different SlideWiki features as well as their usage during the evaluation. We used the *System Usability Scale* (SUS) [14] to grade the usability of SlideWiki. SUS is a standardized, simple, ten-item Likert scale-based questionnaire[6] giving a global view of subjective assessments of usability. It yields a single number in the range of 0 to 100 which represents a composite measure of the overall usability of the system. The results of our survey showed a mean usability score of 69.62 for SlideWiki which indicates a reasonable level of usability. Of course, this is a very simplified view on usability and we expect even better results could be achieved by putting more effort into the SlideWiki development (the development of SlideWiki only consumed 5 man months). However, our goal was to demonstrate that SlideWiki implementations with good usability characteristics can be created with relatively limited effort. In addition to quantitative results, we also collected a number of user suggestions. For instance some users suggested improving the WYSIWYG editor for adding predefined shapes, providing autosave feature, supporting more import/export formats, defining user groups etc.

6 Related Work

Related work can be roughly divided into the following three categories:

Wiki-based Collaborative Knowledge Engineering The importance of wikis for collaborative knowledge engineering is meanwhile widely acknowledged. In [16],

[6] http://www.usabilitynet.org/trump/documents/Suschapt.doc

for example, a knowledge engineering approach which offers wiki-style collaboration is introduced aiming to facilitate the capture of knowledge-in-action which spans both explicit and tacit knowledge types. The approach extends a combined rule and case-based knowledge acquisition technique known as Multiple Classification Ripple Down Rules to allow multiple users to collaboratively view, define and refine a knowledge base over time and space. In a more applied context, [11] introduces the concept of wiki templates that allow end-users to define the structure and appearance of a wiki page in order to facilitate the authoring of structured wiki pages. Similarly the Hybrid Wiki approach [15] aims to solve the problem of using (semi-)structured data in wikis by means of page attributes. SlideWiki differs from such general purpose wiki-based knowledge engineering methodologies due to its domain-specific functionalities which are provided for presentations. The wiki paradigm was meanwhile also applied to domain-specific applications, such as, for example, *Adhocracy*[7] – a policy drafting tool for distributed groups.

Semantic Wikis. Another approach to combine wiki technology with structured representation are semantic wikis [17]. There are two types of semantic wikis. Semantic text wikis, such as *Semantic MediaWiki* [13] or *KiWi* [18] are based on semantic annotations of the textual content. Semantic data wikis, such as *OntoWiki* [12], are based on the RDF data model in the first place. Both types of semantic wikis, however, suffer from two disadvantages. Firstly, their *performance and scalability* is restricted by current triple store technology, which is still an order of magnitude slower when compared with relational data management, which is regularly confirmed by SPARQL benchmarks such as *BSBM* [3]. Secondly, semantic wikis are generic tools, which are not particularly adapted for a certain domain thus substantially increase the usage complexity for users. The latter problem was partially addressed by OntoWiki components such as *Erfurt API*, *RDFauthor* and *Semantic Pingback*, which evolved OntoWiki into a framework for Web Application development [12].

Presentation Management Systems. There are already many Web-based platforms that provide services for online creation, editing and sharing of presentations. *SlideShare.net* is a popular Website for sharing presentations[8]. Comparing to SlideWiki, it does not provide any feature to create and reuse the content of presentations. *SlideRocket.com* and *Prezi.com* are other related works which help people to create fancy and zoomable presentations. In contrast to SlideWiki, they focus more on the visualization aspects rather than the content of the presentations. *Microsoft SharePoint Online* [9] and *SlideBank.com* are two commercial solutions which provide the feature of slide libraries to allow users to work with

[7] http://trac.adhocracy.cc/

[8] Other examples include: *authorSTREAM* (http://www.authorstream.com), *SlideServe* (http://www.slideserve.com), *Scribd* (http://www.scribd.com) and *slideboom* (http://www.slideboom.com)

[9] http://sharepoint.microsoft.com

PowerPoint slide decks stored in the cloud. Despite SlideWiki which is an online platform, these tools adopt the Software-As-A-Service approach to enable a synchronization between desktop applications and Web service providers.

7 Conclusion

In this paper we presented the SlideWiki platform for elicitation and sharing of corporate knowledge using presentations. SlideWiki addresses weaknesses of conventional presentation tools currently used by organizations. It provides a crowd-sourcing platform for the collaboratively authoring of presentations. The created presentations will help to effectively shape the organizational memory by utilizing crowd feedback. SlideWiki also addresses e-learning as well as social aspects of knowledge management by providing features such as creating questionnaires, following, tagging and discussing the presentation content.

We see this effort as the first step in a larger research and engineering agenda to employ presentations for knowledge management. As a future work, we envision to provide crowd-sourced translation of presentations. This will enable multi-national organizations to employ the full potential of their dispersed branches for knowledge representation and exchange. We also aim to improve the interactivity of the system by providing features like video chats or shared blackboards synchronized with the presentations.

Acknowledgments. We would like to thank our colleagues from AKSW research group for their helpful comments and inspiring discussions during the development of SlideWiki. This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumueller, D.: Triplify: Lightweight linked data publication from relational databases. In: WWW 2009, Spain. ACM (2009)
2. Auer, S., Khalili, A., Tarasova, D., Ermilov, I.: Wikiapp - engineering of domain-specific wiki applications (2012), http://svn.aksw.org/papers/2012/WISE2012_SlideWiki/public.pdf
3. Bizer, C., Schultz, A.: The berlin sparql benchmark. Int. J. Semantic Web Inf. Syst. 5 (2009)
4. Blankenship, S., Ruona, W.: Exploring knowledge sharing in social structures: Potential contributions to an overall knowledge management strategy. Advances in Developing Human Resources 11(3) (2009)
5. Bradley, A.J.: Ten primary design considerations for delivering social software solutions: The plant seeds framework. Technical report, Gartner (2009)
6. Casey, A.J., Olivera, F.: Reflections on organizational memory and forgetting. Journal of Management Inquiry 20(3), 305–310 (2011)
7. Cobb, J., Steele, C.: Association learning management systems (April 2011)

8. Doumont, J.-L.: The Cognitive Style of PowerPoint: Slides Are Not All Evil. Technical Communication 52(1), 64–70 (2005)
9. Fonstad, N.O.: Tangible purposes and common beacons: The interrelated roles of identity and technology in collaborative endeavors. In: OKLC (Organizational Learning, Knowledge and Capabilities) (2005)
10. Gongla, P., Rizzuto, C.R.: Evolving communities of practice: IBM Global Services experience. IBM Systems Journal 40(4), 842–862 (2001)
11. Haake, A., Lukosch, S., Schümmer, T.: Wiki-templates: adding structure support to wikis on demand. In: Int. Sym. Wikis, pp. 41–51. ACM (2005)
12. Heino, N., Dietzold, S., Martin, M., Auer, S.: Developing semantic web applications with the ontowiki framework. In: Networked Knowledge - Networked Media, SIC, pp. 61–77. Springer (2009)
13. Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., Studer, R.: Semantic Wikipedia. Journal of Web Semantics 5 (2007)
14. Lewis, J.R., Sauro, J.: The Factor Structure of the System Usability Scale. In: Kurosu, M. (ed.) HCD 2009. LNCS, vol. 5619, pp. 94–103. Springer, Heidelberg (2009)
15. Matthes, F., Neubert, C., Steinhoff, A.: Hybrid wikis: Empowering users to collaboratively structure information. In: ICSOFT (1), pp. 250–259. SciTePress (2011)
16. Richards, D.: A social software/web 2.0 approach to collaborative knowledge engineering. Inf. Sci. 179 (2009)
17. Schaffert, S., Bry, F., Baumeister, J., Kiesel, M.: Semantic wikis. IEEE Software 25 (2008)
18. Schaffert, S., Eder, J., Grünwald, S., Kurz, T., Radulescu, M., Sint, R., Stroka, S.: Kiwi - a platform for c wiki social software. In: SemWiki, vol. 464. CEUR-WS.org (2009)
19. Valaski, J., Malucelli, A., Reinehr, S.: Ontologies application in organizational learning: A literature review. Expert Systems with Applications 39(8), 7555–7561 (2012)

A Knowledge-Based Approach to Augment Applications with Interaction Traces

Olivier Curé[1], Yannick Prié[2], and Pierre-Antoine Champin[2]

[1] Université Paris-Est, LIGM, CNRS UMR 8049, France
ocure@univ-mlv.fr
[2] Université Lyon 1, LIRIS, CNRS UMR 5205, F-69622, France
{yannick.prie,pierre-antoine.champin}@liris.cnrs.fr

Abstract. This paper presents a trace-based framework for assisting personalization and enrichment of end-user experience in an application. We propose a modular ontology-based architecture, to provide semantics for interaction traces, observed elements and their associated objects, and we extend existing inference services, with a declarative and generic approach, in order to reason with those interaction traces. We present the architecture of our framework and its reasoning levels, provide a proof of concept on a medical Web application, and emphasize that different kinds of actors can benefit from the supported inferences.

1 Introduction

We present an assistance framework for personalizing and enriching end-user application interactions. These features rely on user profiles generated from interaction traces that provide information on how data is used and created. Our declarative approach is based on explicitly modeled interaction traces consisting of recordings of *observed elements* (henceforth *obsels*) that are collected during the use of the application.

A first contribution is to provide a Description Logics (DL) [4] based approach to modeling traces by constructing a global knowledge base of entities and actions that can be reasoned upon. The semantics of interaction traces, obsels and their associated objects are provided through mappings to DL concepts or roles. DL formalism has been selected because it enables to reason in a sound and complete manner, and enables interoperability by underlying Semantic Web technologies. Another contribution is to define means of reasoning over various levels of interaction knowledge. Standard (*i.e.* subsumption) and non-standard DL inferences support these services. In order to represent uncertainty, a probabilistic approach is used. Various levels of reasoning are presented that support various kinds of user assistance based on the manipulated objects, traces, and user profiles. Moreover, we have implemented this approach in an existing medical application, and have shown its interest by adding several functionalities that have proven valuable to various actors of this system, from admin to end-users.

This paper is organized as follows. Sec. 2 is dedicated to related works. Sec. 3 details how an application can be augmented with our framework for managing interaction traces, while Sec. 4 describes the different reasoning approaches. In Sec. 5, we provide an evaluation on our running example, emphasize on the system's inferences adequacy.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 317–326, 2012.

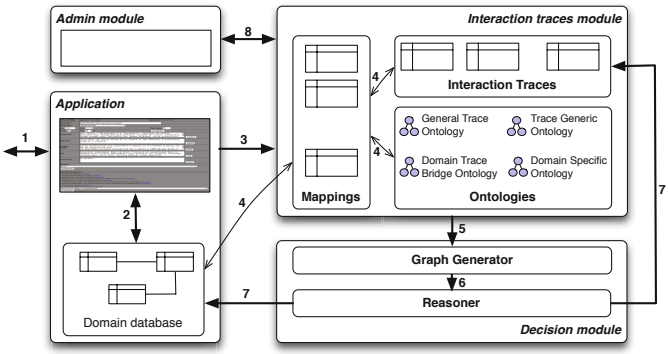


Fig. 1. Architecture overview of an application enriched with a trace-based system for assistance

2 Related Works

Interaction traces have long been considered a valuable source of information, either for an *a posteriori* analysis of the activity, or during the traced activity in order to provide some kind of user-assistance [13]. The latter is based on the assumption that the traces can help to automatically build a user's profile, instead of requiring the user's to manually provide their personal information and preferences [11]. Different paradigms have been explored to provide trace-based user-assistance, such as case-based reasoning (traces can be used to dynamically generate cases corresponding to a problem at hand [10]), or machine learning (traces are used as a training set for further recommendations [11]). Traces can also be modeled using ontologies, allowing complex inferences to be performed in order to assist the user [7]. Like our system, this work uses Semantic Web technologies to represent ontologies. There is indeed a growing interest in modeling temporal information related to users' activity on the Semantic Web [6]. [1] combine statistical analysis with a semantic model and rules in order to measure the health of on-line communities by analyzing the behavior of their users. There has been other work aiming at integrating time in DL [3] and Semantic Web technologies [2,12]; however those approaches consider time as a dimension orthogonal to the data. As such, they are not adapted to our approach where time is merely a property of every obsel.

3 A Framework for Integrating Interaction Traces in Applications

Fig. 1 describes the different information flows between the components of our architecture. The user interacts *(1)* with the application, causing updates to the domain database *(2)* and the interaction traces component *(3)*. In the latter, a subset of the end-user interactions is stored as temporally situated interaction traces, *e.g.* start of a session or description of an item. This subset of interactions is defined through a set of mapping assertions (*Mappings* component) between interaction traces and elements of the ontologies. Note that some mapping assertions also relate the database with the ontology *(4)*. A set of interaction traces, mapping assertions and ontologies is used by the *Graph generator* to define a global graph *(5)*, containing the overall information involved in the process of describing interactions. Moreover, it integrates metadata such

as object specific information coming from the ontologies and/or the domain database. The global graph is passed to the reasoner *(6)* to generalize on the represented information, computing subsumption relationships between ontology concepts. The generalizations are both stored as (transformed) interaction traces and in the domain database *(7)*, *e.g.* to store end-user models which will be used to personalize the application together with information directly coming from the traces. Finally, the *Admin* module enables to manage and query information stored as raw or transformed interaction traces *(8)* to detect assets and drawbacks of the application, e.g. objects that are rarely accessed, *etc*.

Representation of Interaction Traces. We use the meta-model proposed in [9] for managing interaction traces: a trace is defined as a list of timestamped *obsels*, holding a set of attributes and relations. Each trace is related to a *trace model*, defining the different types of obsel the trace can contain, and which attributes and relations obsels of each type can have. A trace-based management system (TBMS) collects obsels from the application in so-called *primary traces*. A TBMS also manages *transformed traces*, which are computed based on the content of other traces, providing higher levels of interpretation. Hence we have three knowledge containers for each trace: its metadata , its model, and the list of contained obsels. In our framework, the set of obsels recorded in an interaction trace is the representation of an end-user's interactions with a given application. They mainly correspond to CRUD (*Create*, *Read*, *Update*, *Delete*) operations, but can be extended to higher level operations (*e.g. Copy*, *Paste*).

Each end-user session is represented by a single trace, satisfying the schema (*idT, idU, ssDate, seDate*) where attributes respectively identify an interaction trace, the subject of the trace (*i.e.* the end-user whose activity is recorded), the session start and end dates. Traces then contain a number of obsels describing the data operations performed within the application during the session. Obsels satisfy the schema (*idT, obselDate, idObj, op, objField, oldValue, newValue*) where *idT* references the containing trace, *obselDate* is the timestamp of the obsel (and must be comprised between *ssDate* and *seDate* of the containing trace), *idObj* identifies the object impacted by the interaction, *op* identifies the data operation performed on that object, *objField* identifies the object's field impacted by the operation, if appropriate, and *oldValue* and *newValue* contain, if appropriate, the field value before and after the operation respectively. The last three attributes are optional, depending on the kind of operation. That is a *Retrieve* operation does not impact a specific field of an object and has none of those three attributes; a *Delete* operation only has an old value, an *Insert* operation only has a new value and an *Update* operation has the three attributes filled in.

Ontologies for traces. A particular attention has been given to the ease of plugging in and out ontologies and to provide a comprehensive decoupled organization. The default setting of our framework consists of the following 4 ontologies. The *general trace ontology* provides a model for a high-level view of interaction traces. It contains 3 concepts: *Trace*, *Obsel* and *Subject* (the user whose interaction was traced); 2 object properties: *describes* and *composedOf*, respectively relating an interaction trace to its subject and its obsels; 4 datatype properties, supporting subject identification and temporal information. The *trace generic ontology* supports the definition of a set of obsels and corresponds to a hierarchy of concepts some of which subsume concepts of the *general trace ontology*. It aims to specify data operations needed to reason with interaction traces.

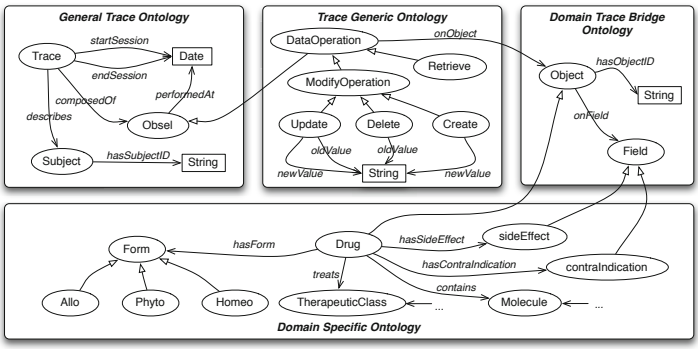


Fig. 2. Ontology-based framework for considering both traces and domain objets

The *domain specific ontology* describes the semantics of the objects observed in the interaction traces. In opposition to the other ontologies of the architecture, this component may integrate an ABox.Obviously, the more information is provided on observed objects (either in a TBox or an ABox), the more accurate the results of our inferences. The *domain trace bridge ontology* links the *trace generic ontology* to the *domain specific ontology*. It contains 2 concepts: *Object* and *Field*, and 2 properties to identify an object and relate an object to its fields. These 2 concepts subsume some of the concepts of the *domain specific ontology* and are related to some of the *trace generic ontology* concepts via object properties, *e.g. onObject* and *hasField*.

Example 1. Fig .2 provides a comprehensive view of a framework instance for our medical application. The *domain trace bridge ontology* provides information on drug objects which have a certain form (*i.e.* allopathy or homeopathy), treat some therapeutic classes and contain some molecules. Two other concepts of this ontology, namely *ContraIndication* and *SideEffect* are subsumed by the *Field* concept of the *domain trace bridge ontology*. Note that only *Drug* is subsumed by the *Object* concept from the bridge ontology, so only operations on drugs will be recorded in the trace. However, *Molecule* and *TherapeuticClass* can be used as external knowledge to support inferences on drugs and the related obsels.

If our framework was to be used in a completely different domain, the *domain specific ontology* would have to be replaced. The modular structure would however make it easy to reuse an existing Semantic Web ontology for the new application domain, and link it to the other ontologies through mappings to the *domain trace bridge ontology*.

4 Reasoning over Traces

We model interaction traces using a set of ontologies, using DL to provide them with model-theoretic semantics [4]. A DL knowledge base (KB) is composed of a TBox and an ABox which respectively correspond to a set of terminological axioms and concept/property assertions. A key feature of DLs is to integrate in the system the following set of standard inferences: concept satisfaction and subsumption, instance

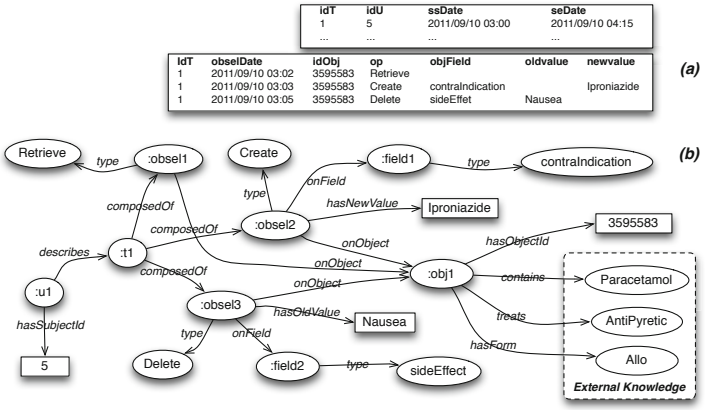


Fig. 3. A trace extract and its graph instance (with some data omitted for readability)

checking, realization and retrieval. Although we use some of these inferences, we also adapt two non standard DL inferences named Most Specific Concept (MSC) and Least Common Subsumer (LCS) [4]. The MSC of an individual α wrt a KB is the concept description C such that (i) α is an instance of C and (ii) if there is another concept C' of which α is an instance of, then C' is more general w.r.t. subsumption than C. The LCS of a given set S of concept descriptions is the most specific concept (wrt the subsumption relationship) subsuming all concepts descriptions in S. In general, they are computed with either a structural subsumption algorithm or via the specification of a normal form. Within our framework, MSC is used to generate DL concepts (hence models) from individuals corresponding to instances of interaction traces, obsels or their objects. Given these instances, the system generates concept description in the $\mathcal{EL}$ DL [5] which underpins the OWL2 EL fragment. This DL presents good properties in terms of reasoning: subsumption is polynomial even if one allows for cyclic terminologies and the MSC of an ABox individual always exists. Moreover, users' goals and intentions are too complex to be entirely accounted for by formal inferences. To handle this uncertainty, we use Probabilistic $\mathcal{EL}$[8], an extension of $\mathcal{EL}$ that associates probabilities to concept descriptions.

As seen in Fig. 1(5), the reasonner works on a global graph generated from different knowledge sources. Fig. 3(a) presents some interaction traces performed over our medical application while Fig. 3(b) displays its associated instance graph. Note that the *contains*, *treats* and *hasForm* properties associated to the *:obj1* node can not be generated from the interaction traces. In fact they correspond to external knowledge retrieved from the domain database. This is a difference with the original TBMS approach [9]. A major advantage of using ontologies is the ability to generalize patterns discovered with type inference, *i.e.* the most common concept that satisfies a given situation.

Inferencing over various levels of interaction knowledge. The analysis of the end-user's behavior can be performed at different levels. We propose 5 of them which cover a wide range of relevant features and still support a generic approach. They correspond to the main DL concept abstractions found in our ontologies: *Field*, *Object*, *Obsel*, *Trace* and

Subject. These levels require different forms of processing: a finite model checking approach handled with a simple query answering interface; or a more elaborate logical reasoning. Inference services provided by each level build on the previous ones.

The *Field level* is the most specialized level and little information can be retrieved from it. Intuitively, it enables to retrieve which kind of fields specified in the *Domain specific ontology* have been observed. Hence a simple query answering interface is sufficient. For instance, the question: "Which fields have been observed?" may be answered with an enumeration of subconcepts of the Field concept that have been effectively observed. This query may help to detect which fields are never operated upon.

The *Object level* is central as it is the first one to require logical reasoning and all remaining levels build on its inferences. Both query answering and logical reasoning are needed at this level. For instance, query answering can be used to identify objects operated upon. The main logical reasoning of this level enables to generalize on a given object description (detailed in Sec. 4).

The *Obsel level* enables to study the operation associated to an observed object. Typical queries are CRUD-like and reasoning amounts to generalize obsel description.

The *Trace level* is an interesting level due to the non-functionality of the *composedOf* property, *i.e.* a trace can be composed of several obsels. Moreover, an interaction trace is associated to some temporal values representing its begin and end session dates. Thus, a query answering approach can answer to questions such as "When was a trace recorded? How long did it last?". Logical reasoning is mainly concerned with generalizing a given interaction trace. An interesting aspect of reasoning at this level involves considering ordered sequences of generalized traces. By storing the result of that generalization as *transformed* traces, we can then query sequences of similar sessions.

Finally, the *Subject level* enables to infer over a set of interactions performed by an end-user. Query answering can be used to reply to the following questions for a given end-user: "When did this user use the application for the last time?", "How often does this user use the application?". We can also consider queries involving a group of end-users, *i.e.* queries aggregating over end-users. Logical reasoning amounts essentially in defining an end-user *profile* (i.e. concept descriptions) given her set of interaction traces. In our running example, this user profile aims to define her pharmacology expertise, *i.e.* in terms of molecules, therapeutic classes and drug forms she is an expert in.

Reasoning method. The reasoning methods needed by our levels of analysis are based on the following 3 algorithms. The first algorithm, *simpMSC*, corresponds to a simplified version of MSC and aims to generate a DL concept description for any individual present in the global graph. These DL concepts are specified by the following normal form: $\bigsqcap A_i \sqcap \bigsqcap \exists r_j A_k$ where A_i and A_k are atomic concepts and r_j is an object property. The algorithm takes as input an individual α and acts as follows. For all assertions of the form $\Gamma(\alpha)$ (*i.e.* defining the type of α), it creates a conjunction AC of Γs. Moreover, for all assertions of the form $r(\alpha, \beta)$ with r an object property assertion, it creates a conjunction EQ of $\exists r.\beta$. The algorithm returns as output $AC \sqcap EQ$. This algorithm is used at both the *Object* and *Obsel* levels respectively to produce a DL concept for an object in terms of elements of the *Domain specific ontology* and an obsel in terms of elements of the *Trace generic ontology* and *Domain specific ontology*.

Example 2. Consider the *:obj1* individual in Fig. 3. Its assertion set contains: a type definition *Drug*(:obj1), 3 object property assertions (*contains*(:obj1, Paracetamol), *hasForm*(:obj1, Allo) and *treats*(:obj1,AntiPyretic)) and a data type property assertion: (*hasObjectID*(:obj1,3595583)). The execution of *simpMSC* over this individual returns: $Drug \sqcap \exists contains.Paracetamol \sqcap \exists treats.AntiPyretic \sqcap \exists hasForm.Allo$.

The second algorithm, *probLCS*, corresponds to an approximation of LCS extended with probabilities on concepts. We use the notation of [8]: a concept $Trace \sqcap \exists composedOf(P_{=1/2}Retrieve \sqcap \exists onObject.O1)$ describes an interaction trace composed of an obsel on object O1 and a *Retrieve* data operation with probability 0.5. We restrict probabilities to occur in front of DL concepts and allow disjunction in the form $A \sqcup B \sqsubseteq C$ since it can be eliminated modulo the introduction of the new concept C.

The input of this algorithm is a set of concept descriptions produced by *simpMSC*. In a first step, a disjunction of atomic concepts is computed. That is, it counts the number of occurrences of each non quantified atomic concepts over the set of descriptions and produces a probabilistic formula for each of them where the probability is the number of occurrences of this concept divided by the total number of occurrences of the set. In the second step, for each distinct existentially quantified property, the first step is applied over its set of concepts. The *probLCS* algorithm serves at the *Trace* level to specify a probabilistic view of the obsels present in an interaction trace.

Example 3. Consider the obsels of Fig. 3 (*i.e.* :obsels$_i$ with i ∈ [1,3]). Applying *simpMSC* to these obsel individuals yields the respective concepts (where O1 is the concept description from Ex. 2): $Retrieve \sqcap \exists onObject.O1$, $Create \sqcap \exists onField.ContraIndication \sqcap \exists onObject.O1$, $Delete \sqcap \exists onField.SideEffect \sqcap \exists onObject.O1$. Computing *probLCS* over this set yields the following concept: $(P_{=1/3}Retrieve \sqcup P_{=1/3}Create \sqcup P_{=1/3}Delete) \sqcap \exists onField.(P_{=1/3}$ ContraIndication$\sqcup P_{=1/3}$ SideEffect$) \sqcap \exists onObject.P_{=3/3}O1$

The last algorithm, *setProb*, operates on a set of probabilistic concepts to create a generalized concept, using the notion of concept comparability.

Definition 1. The comparability property, defined over the normal form of a probabilistic concept, *i.e.* $\bigsqcap(SC) \sqcap \bigsqcap(\exists rSC)$ where SC, standing for Simple Concept, is a disjunction of probabilistic atomic concept, is stated as follows: non quantified simple concepts are comparable and simple concepts existentially quantified by the same property are also comparable. No other concepts are comparable.

Given this comparability property, the generalization of probabilistic concepts is defined as summing the probabilities of comparable simple concepts. To compute a relevant sum, all probabilities are expressed over the total number of occurrences found locally. This approach enables to sum properties and to have a coherent global view of the distribution of probabilities. The *setProb* algorithm is used at the *Subject* level when a generalization of interaction traces described by a given end-user is required.

Example 4. Consider the composition of the 2 following interaction traces:
- $(P_{=1/3}Retrieve \sqcup P_{=1/3}Create \sqcup P_{=1/3}Delete) \sqcap \exists onField.(P_{=1/3}$ ContraIndication $\sqcup P_{=1/3}$ SideEffect$) \sqcap \exists onObject.P_{=3/3}O1$
- $(P_{=2/4}Retrieve \sqcup P_{=2/4}Update) \sqcap \exists onField.(P_{=2/4}$ SideEffect$) \sqcap \exists onObject.P_{=4/4}O2$

The execution of *setProb* over these interaction traces yields: $(P_{=3/7}Retrieve \sqcup P_{=1/7}Create \sqcup P_{=1/7}Delete \sqcup P_{2/7}Update) \sqcap \exists onField.(P_{=1/7}$ ContraIndication $\sqcup P_{=3/7}$ SideEffect) $\sqcap \exists onObject.(P_{=3/7}$ O1 $\sqcup P_{4/7}$ O2)

The probabilities of the concept description computed by *setProb* represent the uncertainty of our conclusions about the user's activity at a certain level. In the context of the *Subject* level, this concept description serves to create an approximation of a user model. A threshold θ is used to cope with the uncertainty of our approximation.

Definition 2. Given a probabilistic DL concept, a certain rewriting of this concept wrt a threshold θ is computed by retaining only the inner concepts whose probability is superior or equal to θ.

The certain rewriting of a probabilistic concept wrt to a threshold θ serves to generate a model for a given subject. That is the remaining concepts of the description specify the expertise/point of interest of the end-user.

Example 5. Consider the probabilistic concept generated in Ex. 4 and a threshold θ=1/3. The certain version corresponds to: (*Retrieve*) $\sqcap \exists$ onField.(SideEffect) $\sqcap \exists$ onObject.(O1 $\sqcup$ O2) where the drug O1 (resp. O2) has form Allo, treats Cough and contains Dextromethorphan (resp. has form Phyto and treats the respiratory system). This enables to define a user model corresponding to a domain expert involved in *Retrieve* data operation and side effect field over allopathic and phytotherapeutic drugs of both the Cough and respiratory system, containing the Dextromethorphan molecule.

Finally, testing subsumption of concepts filling the *onObject* property is performed. Intuitively, using the Domain specific ontology, the system checks for concepts covering wrt. to subsumption. Thus a set of subsumed concepts are replaced with the super concept. This approach limits the size of generated user models and improves their relevance by relating them to named concepts rather than complex concepts expressions.

5 Trace-Based Assistance in a Self-prescription Application

Reasoning occurs at the different levels of interaction knowledge. Several kinds of actors (end-user, staff manager and developer) can benefit from various inference services.

At the *Object* level, the system can efficiently compute statistics on frequently modified objects and fields and hence provide up-to-date information on market evolution.

The *Obsel* reasoning level enables to detect repetitive operations performed on same category objects. The system then proposes the end-user to execute them automatically on the remaining objects in this category, as soon as it has been repeated more than a given threshold (defaulting to 3). This kind of information also supports the discovery of some integrity constraints, e.g. which fields or objects are most frequently updated for the drug category of homeopathy or respiratory system. For example, it is relatively frequent in the medical domain that some molecules are being attributed new properties. This may cause drugs containing that molecule to change from OTC to requiring a prescription. After the end-user has changed the state of 3 such drugs, the system will offer to make the change to all remaining drugs. Before the adoption of our trace reasoning level, all modifications had to performed manually.

In our medical application, all modifications are double-checked by an expert of this product's category, *i.e.* either form, molecule or therapeutic class. For instance, a homeopathy expert usually does not have the expertise to check information on the allopathy drug category. The *Subject* level automatically generates this expertise profile for each user based on their interaction traces. Based on these user models, we are able to personalize the home page of each end-user in the following way: (i) she is only asked to check drugs in her domain of expertise, (ii) she is provided with a list of contacts in her domain of expertise and (iii) health news displayed in the application are prioritized according to her domain of expertise. This approach has simplified the tasks of our team of health care professionals and has significantly lowered time delivery of data updates.

The administration tool is also very useful to improve our system. Due to different forms of reasoning, it enables to identify drug categories not frequently operated upon (obsel level), to control the cleaning/checking activity of a given domain expert (trace level), to detect domains of expertise missing in a team of end-users, to discover who checks and/or overrules the updates of a given expert (subject level), *etc*.

Evaluation. We have conducted an evaluation to highlight our framework's assets and weaknesses on certain criteria related to *Subject* level reasoning and user profiles: a) productivity gain of end-users, b) correctness of the models generated for each end-user. The experimentations have been conducted on our medical application over a real dataset involving 12 health care professionals over 3 months and resulting in the recording of 420 interaction traces and over 23000 obsels.

The first experiment tackles the gain of productivity of end-users. The experimentation took place after a period of 2 months of recording interaction traces.None of them were aware of the existence of the framework. Given the user models generated after the recording of 2 months of interaction traces, we divided the group of 12 end-users into 3 homogeneous (*i.e.* based on the precision of their models) groups of 4 persons. Then over a period of 3 weeks, we conducted a survey over the evolution of the precision of the *check box*. This box is displayed on the home page of each end-user and contains a list of drugs that have been recently modified and which need a checking by another health care professionals before being put into production. Hence, it is an invitation for end-users to control the information associated to a given drug. For group #1, the *check box* did not benefit from the inferences of the framework. The end-users hence had to browse the list of drugs to find by themselves the ones in which they have some expertise. For group #2, the *check box* was progressively taking benefit from the 2 months of analysis. That is in the first week, the box benefited from the user model deduced after 3 weeks of analysis, in the second week, it benefited from 6 weeks of analysis and in the last week it used the models generated over the 2 months of analysis. In the case of group #3, the *check box* benefited from the 2 months of analysis right from day one of the experimentation. Participants in each group rated the adequacy of the drugs presented in the *check box* according to their medical expertise. A Likert scale with 5 ordered responses was used, ranging from '5=strongly adapted' to '1= not adapted'. Fig. 4(a) emphasizes that group #1 does not see any improvement in adequacy of the box list. As expected, through the period of the evaluation, participants of group #2 sensed an improvement in the adaptability of the proposed list of drugs. Finally, the improvement was felt right away for members of group #3.

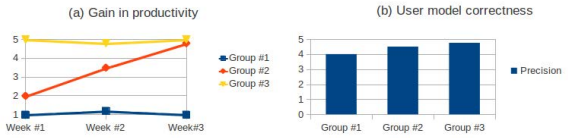


Fig. 4. Results of experimentations

The second experiment was conducted right after the first one and once all end-users were informed about the existence and aim of the framework. We presented a comprehensive view (*i.e.* presented in natural language) of the user model generated for each participants (out of the 2 months plus 3 weeks of storing interaction traces on experiment #1) and asked them to rate (using the same Likert scale as in (1)) the precision of their profile. Fig. 4(b) presents the average of individual results over members of the 3 groups. The averages range from 4.5 for group #1 to 4.75 for group #3. We consider these results to be satisfactory as it rewards the work invested on this framework. Anyhow, the quality of these results is related to the well-defined expertise of each health care professional participating to the project.

References

1. Angeletou, S., Rowe, M., Alani, H.: Modelling and Analysis of User Behaviour in Online Communities. 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. 35–50. Springer, Heidelberg (2011)
2. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW, New York, NY, USA, pp. 635–644 (2011)
3. Artale, A., Franconi, E.: Temporal description logics. In: Fisher, M., et al. (eds.) Handbook of Temporal Reasoning in Artificial Intelligence, vol. 1, pp. 375–388. Elsevier (2005)
4. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the el envelope. In: IJCAI, pp. 364–369 (2005)
6. Champin, P.-A., Passant, A.: SIOC in Action – Representing the Dynamics of Online Communities. In: 6th Int. Conf. on Semantic Systems, I-SEMANTICS 2010, Graz, Austria (2010)
7. Groza, T., Handschuh, S., Müller, K.: The NEPOMUK project - on the way to the social semantic desktop, Graz, Austria (2007) (peer-reviewed)
8. Gutiérrez-Basulto, V., Jung, J.C., Lutz, C., Schröder, L.: A closer look at the probabilistic description logic prob-el. In: AAAI (2011)
9. Laflaquière, J., Settouti, L.S., Prié, Y., Mille, A.: A trace-based System Framework for Experience Management and Engineering. In: EME 2006 in Conjunction with KES 2006 (2006)
10. Smyth, B., Briggs, P., Coyle, M., O'Mahony, M.P.: A Case-Based Perspective on Social Web Search. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 494–508. Springer, Heidelberg (2009)
11. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. In: WWW, New York, USA, pp. 675–684 (2004)
12. Tappolet, J., Bernstein, A.: Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. 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. 308–322. Springer, Heidelberg (2009)
13. Wexelblat, A.: History-rich tools for social navigation. In: CHI 1998 Conference Summary on Human Factors in Computing Systems, CHI 1998, New York, NY, USA, pp. 359–360 (1998)

Building a Library of Eligibility Criteria to Support Design of Clinical Trials

Krystyna Milian[1,2], Anca Bucur[2], and Frank van Harmelen[2]

[1] Vrije Universiteit, Amsterdam
[2] Philips Research, Eindhoven
k.milian@vu.nl, anca.bucur@philips.com, frank.van.harmelen@cs.vu.nl

Abstract. The completion of clinical trial depends on sufficient participant enrollment, which is often problematic due to the restrictiveness of eligibility criteria, and effort required to verify patient eligibility. The objective of this research is to support the design of eligibility criteria, enable the reuse of structured criteria and to provide meaningful suggestions of relaxing them based on previous trials. The paper presents the first steps, a method for automatic comparison of criteria content and the library of structured and ordered eligibility criteria that can be browsed with the fine-grained queries. The structured representation consists of the automatically identified contextual patterns and semantic entities. The comparison of criteria is based on predefined relations between the patterns, concept equivalences defined in medical ontologies, and finally on threshold values. The results are discussed from the perspective of the scope of the eligibility criteria covered by our library.

Keywords: Modeling clinical trial data, Semantic annotation, Medical ontologies, Eligibility criteria, Formalization of eligibility criteria, Supporting design of eligibility criteria.

1 Introduction

Insufficient recruitment is often a barrier that prevents the finalization of a clinical trial and obtaining evidence about new prevention, diagnostic or treatment methods. The recruitment process is time and effort consuming, as for each patient that is considered for enrollment it requires verification of whether the patient satisfies all eligibility criteria of the trial. Additionally, the completion of the trial depends on enrolling a sufficient number of participants. Applications assisting the investigators while designing a trial and further when recruiting patients could help to automate the process.

A few studies have addressed the task of supporting the verification of patient eligibility [1]. However, little attention has been devoted to the support of the design of eligibility criteria. The main purpose of the study reported here was to address this issue. Our objective is to enable the reuse of structured eligibility criteria of existing trials, which can be used to provide meaningful suggestions to trial designers during the definition of new sets of criteria, for example concerning the revision of unnecessarily restrictive conditions.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 327–336, 2012.

We approached the problem by analyzing eligibility criteria of studies published at ClinicalTrials.gov. We extended our previous work on formalization of criteria using contextual patterns [2] by enlarging the set of patterns and identifying in criteria semantic entities i.e. ontology concepts, measurements and numbers. Next, we applied it to automatically structure and classify eligibility criteria of breast cancer trials. Further, we designed a method for comparing the criteria content and their restrictiveness. Using obtained results we created a library of structured eligibility criteria. The paper contains examples describing its content and possible usage. In our future work we will connect the formalized eligibility criteria with queries, which will be used to assess whether a given patient satisfies the entry conditions of a trial. Since many eligibility criteria are very similar or even identical across the trials, reusing computable criteria could significantly enhance the recruitment process.

The paper is organized as follows. The next section introduces a method we developed to interpret eligibility criteria by first formalizing the meaning of the criteria and then comparing their restrictiveness. Section 3 describes the model of the library and the way it was populated with data: formalized criteria of existing trials. Further, section 4 gives quantified results of the library content. Related work is described in section 5, the last chapter contains conclusions.

2 Interpreting Eligibility Criteria

This section describes our method designed to build a library of structured eligibility criteria. Our aim was to enable the reuse of structured representation, and provide meaningful suggestions of relaxing criteria to enable enrolling a larger number of participants. Because of similarities and repetitions of criteria across the trials, our claim is that by formalizing eligibility criteria of a large corpus of clinical trials we can create a sufficiently rich library to serve the task. Our method relies on:

1. Extracting eligibility criteria from a corpus of publicly available clinical trials
2. Formalizing their content
3. Identifying similarities between the criteria and determining relations, i.e. which one is more restrictive

2.1 The Method for Formalizing Eligibility Criteria

The method of formalization of eligibility criteria consists of several steps, depicted in Figure 1.

We start with the pre-processing of criteria, delimiting the sentences using GATE [3], the open source framework for text processing. Next, whenever possible, we recognize the domain of the criteria, e.g. Age, Cardiovascular, Chemotherapy etc. Further follow the two main steps of criteria formalization.

First, we recognize the general meaning of a criterion, by detecting the patterns providing the contextual information about the semantic entities mentioned in the criterion. The set of patterns was initially described in our previous

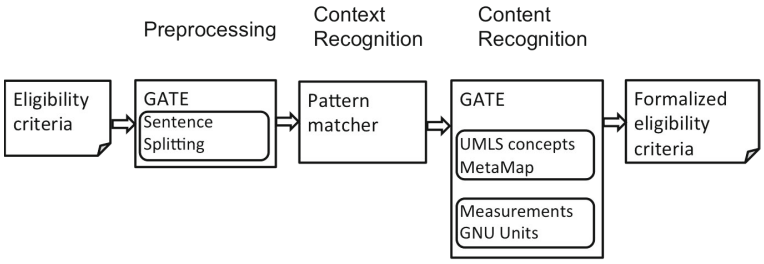


Fig. 1. The pipeline of processing steps of eligibility criteria

work [2] and further extended. It was manually defined by analyzing eligibility criteria published at ClinicalTrials.gov and contains 165 items that reflect the typically occurring constraints. The patterns cover criteria related to patient characteristics (e.g. Age over ()), disease characteristics (e.g. T () stage) and prior and concurrent therapies (e.g. No concurrent () except for ()). They are classified according to several dimensions that characterize the content of corresponding eligibility criteria from various perspectives:

- Temporal status (TS): prior, current, planned event
- Time independent status (TIS): present, absent, conditional
- Constraint types (CT): temporal (start, end, duration), confirmation, co-occurrence, exception, inclusion
- Subject: patient, family of a patient

The algorithm of pattern identification is based on regular expressions, it finds the longest patterns together with the nested ones. In total we defined 468 regular expressions corresponding to the 165 patterns.

Detection of patterns enables recognizing the context in which semantic entities occur. Next, we identify these semantic entities, which can be instantiated by diseases, treatments, lab measurement, value or temporal constraints. We approached the task by incorporating state of the art tools i.e. GATE the NLP framework, providing a library of semantic taggers, and MetaMap, an UMLS [4] ontology annotator. In the workflow of text processing steps we used a tokenizer, sentence splitter, Number, Measurement and MetaMap taggers, wrapped in our application using the GATE API. A result of MetaMap annotation is metadata about identified mapping (or a list of candidates), the UMLS concept id, its preferred name, semantic type (ST), score of mapping, and list ontologies covered by UMLS that specify the concept. The measurement plugin, based on GNU Units [5], recognizes the measurements, including value, unit and dimension, and additionally normalizes the values according to the standard units. Recognition of mentioned entities enables the interpretation of criteria meaning and processing of normalized representation (terms identified in text can be replaced by unique UMLS identifiers, measurements by normalized values and units).

Following example illustrates the approach. In criterion: 'No prior malignancy except for nonmelanoma skin cancer', first, we detect the pattern 'No prior ()

for ()', and second, the concepts 'malignancy' and 'nonmelanoma skin cancer'. To evaluate criteria, the patterns can be linked to predefined templates, the incomplete queries, which after filling with the semantic entities identified and mapped to corresponding items in patient record, can be executed to verify patient eligibility.

2.2 The Method for Comparing Eligibility Criteria

By formalizing eligibility criteria we have created the basis for automated mining of the criteria content. We used obtained results to determine relations between concrete eligibility criteria, in order to assist the designer with proposing alternative, less restrictive, but still meaningful suggestions. This section describes our approach to the criteria comparison based on identified context patterns, ontology concepts and value constraints.

Comparison of Criteria That Match the Same Context Pattern. Recognizing syntactic patterns enables capturing the general meaning of criteria. Information that two criteria match the same pattern provides valuable information about their similarity. Further, depending on their instantiation, they can be classified as comparable. For instance, although the two following criteria: No chemotherapy within last month and No prior lung cancer with last year match the same pattern: No prior () within (), comparing them for our purpose is irrelevant. Criteria can be compared when have the same main argument and different value constraints, i.e.:

- Lower or upper thresholds for lab values, e.g. Bilirubin less than 2.0 mg/dL can be compared with: Bilirubin less than 1.5 mg/dL.
- Temporal constraints, which restrict: start, end or duration of some medical event, for example: At least 1 week since prior hormonal therapy can be compared with At least 4 weeks since prior hormonal therapy.

In both cases the comparison is possible when the values have the same normalized unit identified by MetaMap.

Comparison Based on the Relations between the Context Patterns. To compare criteria with different syntax we designed another strategy. We predefined relations between some patterns (canRelax, canBeRelaxedBy), indicating which pattern can be relaxed by which. These relations express the possibility that corresponding criteria can be in the relation isMoreRelaxed/ isMoreStrict, when they are instantiated with the same argument. The relations between the patterns are based on:

- Explicitly stated exceptions e.g.: No prior () can be relaxed by: No prior () unless (), No prior () except for ()
- Specified value constraints: temporal, confirmation, number of occurrences. The constraints, depending on the context (Time independent status), relax or restrict the primary pattern, for example:

- No prior () can be relaxed by: No () within (), At least () since ()
- History of () within () or History of () confirmed by () can be relaxed by: History of (), because the latter requires the presence of the event at any point in time, and does not restrict the evidence type.

In total we have defined 36 relaxing relations between the patterns.

Using the described methods for the formalization and comparison of eligibility criteria, we processed inclusion and exclusion criteria from the corpus of clinical trials and populated the library.

3 Library of Eligibility Criteria

3.1 The Model of the Library

This section describes the model of the library of eligibility criteria, designed to reflect the most relevant information about their content.

The library was modeled as ontology to enhance semantic reasoning. The lists of classes, and properties are displayed in Figure 2. The model captures data related to Trial (hasID, hasCriterion), Criterion (hasContent, hasDomain), its Dimensions of classification (hasTemporalStatus, hasTimeIndependentStatus, hasSubject, etc) the formalization of its Content - one from a set of Pattern Instance or Concept. Pattern Instances have modeled corresponding value constraints (hasContent, hasValue, see the list of object and data (sub)properties). Concept has specified its metadata (hasConceptId - UMLS id, hasSemanticType, hasSource - definining ontology and occursIn - links to the criteria where it occurs). Additionally, the model explicitly defines transitive relations between the patterns (canRelax/canBeRelaxedBy), and concrete criteria (isMoreRelaxed/isMoreStrict). The criteria and extracted data are represented as individuals. Modeling the library as an ontology enables sharing it, extending or linking to other sources.

3.2 Populating the Model

Clinical trials that were used to build the library of criteria come from the ClinicalTrials.gov repository, a service of the U. S. National Institute of Health, containing data about clinical trials conducted worldwide. We focused on clinical trials related to breast cancer and processed eligibility criteria from 300 studies. The model was populated using the results of the processing steps described in the previous section. Firstly we split the sentences, next we recognized corresponding patterns and the semantic entities mentioned. For the simplification purpose we took into account only the criteria that match a single pattern.

Each pattern has labelled arguments in order to facilitate the task and correctly associate recognized items. For example a pattern 'No prior () within () except for ()' has labelled its 3 arguments as: main argument, end time constraint and exception, which after detection were saved as values of corresponding object or data properties. Finally, we compared corresponding criteria. The results were saved as triples using the OWL API.

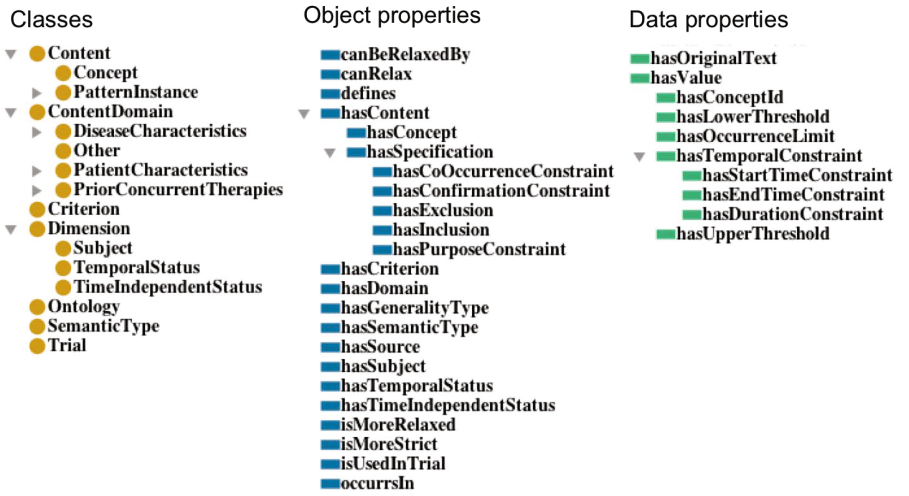


Fig. 2. Library model

4 Results

4.1 Characteristics of the Library

This section describes the final result of populating the library. Its content is quantitatively characterized in Table 4.1. The library contains 1799 structured eligibility criteria out of 10067 (17.9%) used in the experiment, which come from 268 different clinical trials out of all 300 processed. This result indicates the need of improving the recall of the method. One limitation is caused by the fact that our method takes into account only criteria that match one pattern, while many of them are more complex. The interpretation of such criteria would require correct identification of relations of recognized patterns i.e. conjunction, disjunction, nesting. Filtering out criteria, which were matched to some pattern, but the annotation of their arguments with ontology concept by MetaMap did not return any results, caused another reason of low recall.

The table contains also information about the ontology concepts identified, i.e. 1241 UMLS concepts were recognized that belong to 91 various semantic types, and are defined in 46 ontologies covered by UMLS. With respect to the result

Table 1. The library characteristics

Eligibility criteria	1799/10067 (17.9 %)
Trials	268/300 (89.3 %)
Concepts	1241
Semantic Types	91
Ontologies from UMLS	46
Relaxations based on value threshold	202
Relaxations based on semantic modifiers	87

of criteria comparison, in total the algorithm identified 289 cases of eligibility criteria that could be potentially relaxed by one of the other conditions included in the library. This accounts for 16% of the entire number of formalized criteria.

Table 2 characterizes the type of formalized criteria, by giving number of criteria belonging to a few major classes (not mutually exclusive).

Table 2. The characteristics of formalized eligibility criteria

Type of criteria	Count
TS = prior, TIS = absent	13.6%
Criteria requiring confirmation by particular test	3.8%
Criteria with temporal constraint	9.2%
Criteria with value constraints	24.5%
Criteria containing some exception	9.6%

The detailed evaluation of precision of obtained results should be addressed in future work. It depends on precision of pattern detection algorithm, MetaMap, GATE semantic taggers and comparison algorithm.

4.2 Scenarios of Usage

The following scenarios show how the library of criteria could enhance the reuse of formalized criteria by trials designers. Modeling the content of eligibility criteria enabled browsing the library with the fine-grained queries, which correspond to the properties of patterns and instantiating concepts, and find criteria that:

1. Mention a specific concept e.g. 'Tamoxifen'
2. Mention a specific concept in a particular context. Following examples present criteria that mention Tamoxifen in various semantic contexts:

Context	Example of criteria related to Tamoxifen
TS= Planned event	Must be scheduled to receive adjuvant chemo-therapy with or without tamoxifen
TIS= Absence	No concurrent tamoxifen
ST= Mental or Behavioral Dysfunction	No serious toxicity (e.g. depression) thought to be due to tamoxifen
CT= Temporal constraint	At least 12 months since prior tamoxifen, raloxifene, or other antihormonal therapy

3. Mention some concept with a specific semantic type e.g.:

Semantic type	Example of criteria
Enzyme	Transaminases less than 3 times normal
Hormone	No adrenal corticosteroids
Laboratory procedure	Fasting blood glucose normal

4. Have specific domain e.g.:

Content domain	Example of criteria
Biologic therapy	No prior bone marrow transplantation
Cardiovascular	No history of deep vein thrombosis
Neurologic	No dementia or altered mental status

5. Are less restrictive than provided criterion e.g.:

Criterion	Possible relaxation
1. Creatinine < 1.2 mg/dL	Thresholds: 1.3, 1.8, 2.2, 2.5
2. At least 3 months since prior hormonal therapy	Thresholds: 1, 2, 3, 4 weeks
3. No prior endocrine therapy	No prior hormonal therapy for breast cancer
4. No prior malignancy	No other malignancy within the past 5 years except nonmelanomatous skin cancer or excised carcinoma in situ of the cervix

The first and second case represent examples of relaxations based on identifying less restrictive value thresholds. It is worth noting that because of using normalized representations of measurements, it was possible to compare number of months and weeks, as both thresholds were represented in seconds. Suggesting a threshold that was used by another medical expert should be more relevant than suggesting any arbitrary lower value. In the third case (No prior endocrine therapy), the potential relaxation was identified because of detecting ontology concepts, endocrine and hormonal therapy are synonyms, have the same UMLS identifier. The consequence of using this relaxation would be inclusion of patients that obtained such treatment for another purpose than breast cancer. The last example (No prior malignancy) represents a case of finding a relaxation based on both temporal constraint and stated exception. This alternative criterion considers eligible patients who had malignancy more than 5 years ago, or patients with such specific type of disease e.g. nonmelanomatous skin cancer. There is a significant need for providing meaningful suggestions, which can be illustrated by the fact, that searching for the subtypes of malignant disorder only in SNOMED CT, which is one of many ontologies covered by UMLS, returns 48 hits. Proposing those that were used in other eligibility criteria is a way of implicit incorporation of domain knowledge. However, the medical relevance of such suggestions will need to be verified by medical experts.

Apart from finding relevant criteria, the model enables to track their source, the trials where they are mentioned, and browse other criteria that they specify.

Presented methods for knowledge extraction from natural text could be possibly applied for other types of specialized language.

5 Related Work

There are several repositories that contain clinical trial data. The major one is ClinialTrials.gov, at the date of access contained 125301 trials. Its search engine

allows browsing the content by specifying trial metadata such as phase of a trial, interventions used etc. However, besides age and gender other eligibility criteria are not structured. Another source of clinical trial data is provided by the LinkedCT project [6], published according to the principles of Linked Data, enriched with the links to other sources. This repository has the same limitation; namely eligibility criteria are represented as free text.

Many studies have focused on the problem of formalization of eligibility criteria and clinical trial matching. There are several languages, which could be applied for expressing eligibility criteria e.g. Arden syntax [7], Gello [8], ERGO [9] and others. Weng et al [10] present the rich overview of existing options. However, no complete solution to the problem of automatic formalization of free text of criteria has been published. A considerable amount of work in that area is described in [11], where the authors describe their approach to semi-automatic transformation of free text of criteria into queries. It is based on manual preprocessing steps and further, automatic annotation of text with the elements of ERGO, which is a frame-based language. The authors describe how the results can be used to create the library of conditions, organized as a hierarchy of DL expressions, generated from ERGO annotations. They also note that creating such library could help creating criteria more clearly and uniformly. Because of the required manual steps the method cannot be directly reused.

Understanding free text of eligibility criteria could benefit from the information retrieval field, the overview of machine learning and knowledge based methods for relation extraction can be found in [12].

The general task of supporting design of clinical trials has not been broadly addressed in the literature. The system Design-a-trial [13] provides support for design of statistical measurements, i.e. suggesting minimal number of participants and kind of statistical test, ethical issues (e.g. choosing a drug with the least side effects) and preparing required documentation. It does not provide the support for designing eligibility criteria.

6 Conclusions and Future Work

This paper presents the study we conducted with the aim to enhance the reuse of structured eligibility criteria in order to support trial design.

We described our method for automatic formalization of eligibility criteria and the comparison of their restrictiveness. It is based on our pattern detection algorithm, and applies the semantic taggers from GATE, and MetaMap ontology annotator. Using our method we processed eligibility criteria from 300 clinical trials, and created a library of structured conditions. The library covers 18 % of encountered inclusion and exclusion criteria. It can be browsed with fine-grained queries thanks to the detailed modeling of criteria content. The supported scenarios of usage allow searching for eligibility criteria that mention specific data items in particular context, defined by various dimensions (temporal status, time independent status, specification type) and that are less restrictive than a given criterion. The method can be directly used for another trial set.

Possibly, similar strategy for knowledge extraction from natural text could be applied also for documents from other domains defined with specialized language.

The presented study needs additional research. The precision of the methods should be verified. The next challenge is to improve the scope of the library. The first obvious way is to increase the number of clinical trials used for populating. Another more interesting line is to improve the recall of the method for criteria formalization, to increase the variety of criteria that are covered. Additionally, we plan to assess the applicability of presented methods using patient data and feedback of clinical investigators about reusability of structured criteria, and medical relevance and ranking of provided suggestions. Although we created the bases, the empirical study should verify whether it can lead to increased enrollment of patient into clinical trials.

If the library of criteria is linked to a hospital database (EHR), another interesting issue 'trial feasibility' could be addressed using historical patient data. Namely, we could provide information about the consequence of modifying a given criterion in a certain way on the number of potentially eligible patients.

References

1. Cuggia, M., Besana, P., Glasspool, D.: Comparing semi-automatic systems for recruitment of patients to clinical trials. International Journal of Medical Informatics 80(6), 371–388 (2011)
2. Milian, K., ten Teije, A., Bucur, A., van Harmelen, F.: Patterns of Clinical Trial Eligibility Criteria. In: Riaño, D., ten Teije, A., Miksch, S. (eds.) KR4HC 2011. LNCS, vol. 6924, pp. 145–157. Springer, Heidelberg (2012)
3. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th Anniversary Meeting of the ACL (2002)
4. Aronson, A.R.: Metamap: Mapping text to the umls metathesaurus. In: Proceedings AMIA Symposium (2001)
5. G. Units, Units (November 2006), http://www.gnu.org/software/units/
6. Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R.J., Wang, M.: Linkedct: A linked data space for clinical trials, CoRR abs/0908.0567
7. Wang, S., Ohno-Machado, L., Mar, P., Boxwala, A., Greenes, R.: Enhancing arden syntax for clinical trial eligibility criteria. In: Proceedings AMIA Symposium
8. Sordo, M., Ogunyemi, O., Boxwala, A.A., Greenes, R.A.: Gello: An object-oriented query and expression language for clinical decision support. In: Proceedings AMIA Symposium, vol. (5), p. 1012 (2003)
9. Tu, S., Peleg, M., Carini, S., Rubin, D., Sim, I.: Ergo: A templatebased expression language for encoding eligibility criteria. Tech. rep. (2009)
10. Weng, C., Tu, S.W., Sim, I., Richesson, R.: Formal representations of eligibility criteria: A literature review. Journal of Biomedical Informatics (2009)
11. Tu, S., Peleg, M., Carini, S., Bobak, M., Rubin, D., Sim, I.: A practical method for transforming free-text eligibility criteria into computable criteria
12. Cheng, X.-y., Chen, X.-h., Hua, J.: The overview of entity relation extraction methods. Intelligent Computing and Information Science 134, 749–754 (2011)
13. Nammuni, K., Pickering, C., Modgil, S., Montgomery, A., Hammond, P., Wyatt, J.C., Altman, D.G., Dunlop, R., Potts, H.W.W.: Design-a-trial: a rule-based decision support system for clinical trial design. Knowledge-Based Systems

Realizing Networks of Proactive Smart Products

Mathieu d'Aquin, Enrico Motta, Andriy Nikolov, and Keerthi Thomas

Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
{m.daquin,e.motta,a.nikolov,k.thomas}@open.ac.uk

Abstract. The sheer complexity and number of functionalities embedded in many everyday devices already exceed the ability of most users to learn how to use them effectively. An approach to tackle this problem is to introduce 'smart' capabilities in technical products, to enable them to *proactively* assist and co-operate with humans and other products. In this paper we provide an overview of our approach to realizing networks of proactive and co-operating smart products, starting from the requirements imposed by real-world scenarios. In particular, we present an ontology-based approach to modeling proactive problem solving, which builds on and extends earlier work in the knowledge acquisition community on problem solving methods. We then move on to the technical design aspects of our work and illustrate the solutions, to do with semantic data management and co-operative problem solving, which are needed to realize our functional architecture for proactive problem solving in concrete networks of physical and resource-constrained devices. Finally, we evaluate our solution by showing that it satisfies the quality attributes and architectural design patterns, which are desirable in collaborative multi-agents systems.

Keywords: Proactive Problem Solving, Smart Products, Knowledge Systems, Ontology Engineering, Distributed Problem Solving.

1 Introduction

The sheer complexity and number of functionalities embedded in many everyday devices already exceed the ability of most users to learn how to use them effectively. This is not just the case for mass consumer devices, such as mobile phones, but it also applies to work settings, where the increased diversity within product lines introduces new complexities in both product assembly and maintenance processes [1]. An approach to tackle this problem is to introduce 'smart' capabilities in technical products, to enable them to better assist and co-operate with humans and other products [1]. In the context of the *SmartProducts* project[1] we have investigated these issues in a number of scenarios, drawn from the aerospace, car and home appliances industries. In particular, a key requirement for *smart products* imposed by our scenarios is that they need to be able to exhibit *proactivity*, both to improve the level of assistance to users and also to be able to co-operate effectively with other smart products in shared task scenarios. Proactivity can be informally characterized as the ability of a smart product to take initiative and perform some action, without having been specifically instructed to do so

[1] http://www.smartproducts-project.eu/

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 337–352, 2012.

by a user or another smart product [2]. In addition, in order to be able to form and join networks with other smart products and engage in co-operative problem solving, smart products must also be capable of *self-organisation.* This second requirement is a cornerstone of our approach, to ensure that our solutions can be applied in open, networked scenarios, where a closed, top-down design of co-operative problem solving solutions would be too restrictive, if not unfeasible [1].

While the SmartProducts project has looked at a whole range of issues, which need to be tackled to support effective networks of context-aware and proactive smart products, including *user interaction* [3], *access control models* [4], and *distributed storage* [5], in this paper we focus on the core challenge posed by the project scenarios: *the design and implementation of a computational infrastructure to realize networks of proactive smart products.* Hence we will provide a complete overview of our approach, covering both the *knowledge level* and the *symbol level* [6] elements of our solution and in particular showing how we have integrated semantic technologies with ubiquitous computing solutions, to equip resource-constrained physical devices with the capability of representing and reasoning with knowledge structures and engaging in co-operative problem solving.

Specifically, the discussion in the paper will cover: i) our characterization of the notion of 'proactive behaviour', ii) the ontological basis of our framework, iii) a knowledge-level, task-centric problem solving architecture for characterizing co-operative and proactive problem solving in networks of smart products, and iv) a concrete realization of this functional architecture on networks of Android devices. In addition, we will also show that v) our approach to proactive, distributed problem solving satisfies the quality attributes and architectural design patterns, which are desirable in collaborative multi-agents systems [7].

We will start the discussion in the next section, by introducing one of three real-world scenarios which have motivated the work in the SmartProducts project, which will then be used in the rest of the paper to illustrate our solutions.

2 A Scenario

The scenario we consider in this paper is one in which a user, whom we refer to as Ashley, is organizing a dinner for her friends. To this purpose she uses a *Cooking Assistant* application, available on a tablet device, which allows her to specify the parameters for the event, including dietary requirements, food preferences, the number of people attending, the date of the event, known people attending, etc. Ashley's *Smart Kitchen* also includes a *Smart Fridge* and *Smart Cupboards*, which are aware of their contents and can contribute to the shared task scenario –e.g., by providing information about items which are due to expire soon, thus minimizing food wastage. Another smart product available to Ashley is the *Shopping Assistant*, an application[2] which maintains a history of food purchases and is also able to provide

[2] In this paper we use the term 'smart product' to refer both to physical products, such as a smart fridge, and also to software products, such as a meal planner, which is an application running on a particular device. Here, it is important to emphasize that we do not use the term 'smart product' to refer to generic computing platforms, such as a tablet device. These are characterized instead as 'containers', in which numerous smart (software) products may be installed.

information about currently discounted items at a supermarket of choice, which may be relevant to the current meal plan.

Once Ashley has selected a *meal plan* from the various suggestions provided by the *Cooking Assistant* with the co-operation of other smart products, a *shopping list* is produced by the *Shopping Assistant*, on the basis of what is required by the meal plan and what is already available in the house.

Once at the supermarket, the *Shopping Assistant* interacts with *Supermarket Agents* to identify the best deals for the items in the shopping list. However, while Ashley is shopping at the supermarket, a member of the family removes the bowl of strawberries (an ingredient needed for one of the selected recipes) from the fridge. At this point Ashley receives a notification from the *Smart Fridge* that the strawberries are no longer available, and therefore she can choose to add strawberries to the shopping list.

Hence, this scenario requires cooperation between a number of smart products, specifically: *Cooking Assistant, Smart Fridge, Smart Cupboards, Shopping Assistant*, and *Supermarket Agents*, with the interactions between these smart products occurring around shared tasks in two different *ambiances*[3], a *Smart Kitchen* and a *Supermarket*. The requirement for proactivity here means that the smart products must be able to proactively contribute to shared tasks, such as meal planning and shopping at the supermarket, both at the time when requests for information are broadcast but also, as in the example of the strawberries being removed from the fridge, when some event occurs which has implications for the tasks currently being executed.

3 Proactivity in Networks of Smart Products

For an agent to be *proactive*, it needs to be able to take action in scenarios in which such action has not been explicitly programmed in the agent, or explicitly delegated to the agent by a user or another agent. An approach to realizing proactive behaviour entails the development of *personal assistants* [8], which are able to monitor and learn from user behaviour, to anticipate their needs and exhibit proactive assistance. However, in the context of our project, we are less interested in these *application-specific* [2] solutions, than in enabling proactivity in open scenarios, where proactive problem solving takes place in the context of networks of smart products.

So, what are the requirements for proactivity in such scenarios? As discussed in [9], in co-operative problem solving scenarios *"the ability to anticipate information needs of teammates and assist them proactively is highly desirable…"*. In particular, the ability to proactively provide task-related information to a task-performing smart product is especially useful in an open network of smart products, because it avoids the need to know a priori who can provide certain information or carry out a particular task, thus allowing the dynamic formation of networks of co-operating smart products. Hence, a key type of proactive behaviour we want to support concerns precisely this *ability of a smart product to proactively contribute information to another smart product in a task-centric context.*

In addition, achieving proactive, co-operative problem solving and self-organization requires flexible mechanisms for representing and manipulating problem

[3] An ambiance denotes an environment comprising specific smart products, in which collaborative problem solving can take place – see Section 3 for more details.

solving knowledge. Since networks of smart products do not have a predefined organizational structure with prescribed roles, and the capabilities of involved products may vary, smart products can contribute to shared tasks in various ways. For example, the meal planning task presented in Section 2 will be executed differently depending on which smart products are part of the *Smart Kitchen ambiance*, and their capabilities. Hence it is undesirable to design a complete process decomposition in advance and we require instead mechanisms for dynamic process composition, task assignment and proactive problem solving.

An *ambiance* denotes an environment comprising specific smart products, in which collaborative problem solving can take place. While in many cases ambiances reflect physical spaces, such as the ambiance comprising all smart products in a kitchen, the notion itself is flexible and simply denotes any collection of smart products which come together to engage in co-operative problem solving. Both the inclusion of products into an ambiance and the set of permissible behaviours are regulated by *joining policies*. For instance, within a supermarket ambiance, it may be desirable to allow smart products belonging to customers (i.e., on mobile devices) to join the ambiance and receive information, adverts and recommendations from supermarket agents (or even directly from smart food items), but these devices may not be allowed to expose arbitrary tasks to other customers' devices.

In sum, in order to comply with the requirements from our scenarios, the key aspect of proactive behaviour we wish to support concerns the ability of smart products to engage in distributed, collaborative, proactive problem solving, including the ability i) to expose and be aware of shared tasks, ii) to proactively provide task-specific information to other smart products in a particular problem solving context, and iii) to provide actual solutions through problem solving for those tasks which fall within a smart product's set of capabilities.

4 Semantic Technologies for Smart Products

4.1 The SmartProducts Network of Ontologies (SPO)

The *SmartProducts Network of Ontologies* (*SPO*)[4] has been developed both to provide a clear specification of the conceptual model underlying the work on the SmartProducts project, and also to maximise interoperability not just among all SmartProducts applications, but also between these and other applications in related domains. SPO comprises three different sets of modules, which reflect different levels of abstraction and reuse, from the most generic to the application-specific ones. Because each layer is itself divided into several sub-modules, we obtain a highly modular design, which makes it possible to reduce the parts to be used by a device, depending on its functionalities –e.g., no need to use the process model on devices that only serve as data providers.

Figure 1 shows the configuration of SPO, which was used for the smart kitchen scenario. The other project scenarios were modelled in a similar way, reusing the external and generic modules and plugging-in alternative application-specific ones.

[4] `http://projects.kmi.open.ac.uk/smartproducts/ontologies/`

SPO builds on established external ontologies, including the DOLCE Ultra Lite Ontology (DUL)[5], which provides the foundational layer to interpret SPO entities. For example, SPO reuses the *Quality-Region-Parameter* design pattern from DUL to model physical qualities of objects, and the *Plan-PlanExecution* pattern to distinguish between the definition of problem-solving behaviour and its actual execution. Concepts and properties dealing with representation of time are imported from the W3C time ontology[6].

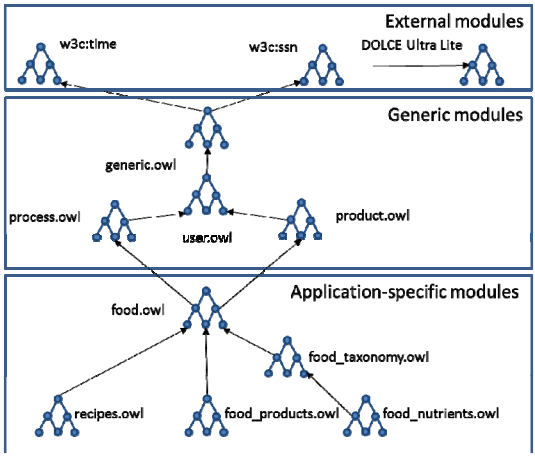


Fig. 1. The SmartProducts Network of Ontologies configured for the smart kitchen scenario

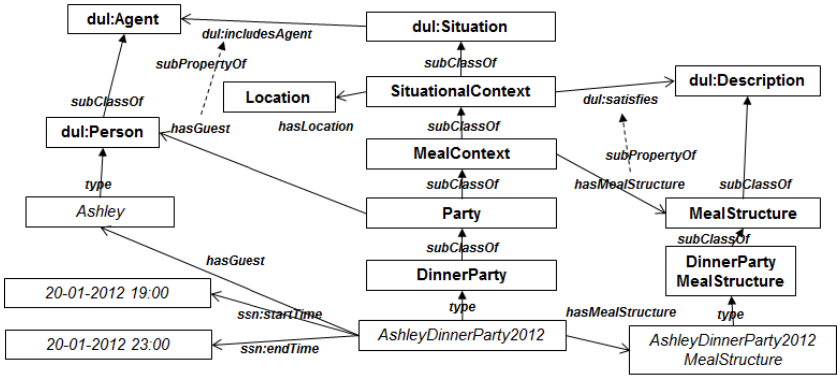


Fig. 2. Modelling situational context

An essential part of SPO covers the *context model*, which is needed to ensure that smart products are able to assess the available information about the state of the world. In particular, the context model includes both 'low-level' contexts, i.e., atomic facts about the state of the environment and its changes –e.g., smoke being detected, as well as 'high-level' ones, i.e., abstracted information about a situation as a whole, which can

[5] http://www.loa-cnr.it/ontologies/DUL.owl
[6] http://www.w3.org/2006/time

be inferred from aggregated low-level context facts –e.g., an emergency situation being recognised on the basis of smoke detection. Sensing devices, which can be either embedded –e.g., a thermometer in a fridge, or external –e.g., a weather service, serve as sources of low-level context information. To model their output, SPO utilizes the recently proposed *Semantic Sensor Network (SSN)*[7] ontology, in particular by reusing its *Observation* design pattern. This allows external sources of sensor data relevant to the products within an ambiance –e.g., readings provided by a weather service, to be smoothly integrated and used alongside data produced by embedded sensors.

To model a high-level situational context, which is usually abstracted from low-level context data, we reuse the design patterns developed in situational awareness research [10] and we characterize a situation as an abstract interpretation of a specific state of the environment. Examples of situations include a dinner party, a snowstorm on the driving route, or a visit to a car workshop. Situations are characterized by their space and time constraints, participating objects, specific relations between these objects, and the situation type [10]. Depending on the situation type, specific types of roles for objects and relations can be defined. For example, a meal can have one or more guests, while a car service operation may involve the car owner, one or more technicians, and a manager. In SPO, the class *SituationalContext* (see Figure 2) defines generic descriptions for high-level contexts. It extends the class *dul:Situation*, by localizing its instances in space and time. Specific properties determining different types of relations between entities and situations are defined by reusing relevant subproperties of the generic *dul:isSettingFor* relation –e.g., *dul:includesAgent*, *dul:includesEvent*, etc.

4.2 Task-Based Problem Solving Architectures

Research on generic models of problem solving in knowledge engineering has developed a number of libraries, architectures, languages and tools to support the specification of generic and reusable problem-solving components [6 11 12 13 14]. A key feature of these architectures is that they support a strong separation of the different building blocks for intelligent systems, for example, distinguishing between *task*, *method*, *domain* and *application knowledge* [11]. Thus, they provide a sound epistemological and architectural basis for developing robust knowledge systems by reuse. In particular, here we build on the *problem solving architecture*[8] defined by the *TMDA framework* [11], which provides a rich modelling framework, based on *task* and *method ontologies* [11], which supports the specification of problem solving components in detail.

Although the TMDA architecture was originally conceived for 'closed' application scenarios in knowledge-based systems, it has been modified in recent years to provide the basis for an open distributed semantic web service architecture [15]. However, our scenarios impose new requirements on the TMDA architecture, as methods and tasks are executed and exposed by specific smart products, and problem solving takes place in specific ambiances, where specific policies are enforced. Hence, as a first step we adapted the TMDA framework to the SmartProducts scenarios, as discussed in the next section.

---

[7] `http://purl.oclc.org/NET/ssnx/ssn`

[8] A problem solving architecture focuses on knowledge-level components for problem solving, in contrast with a system architecture, which concerns technical design issues – see Section 5.

4.3 Ontological Modelling of Problem Solving Knowledge

A key element of SPO is the ontological support for modelling proactive and co-operative problem solving in networks of smart products and to this purpose SPO extends the *task ontology* provided by the TMDA library [11], by introducing the concepts needed to characterize smart products and ambiances and integrating these notions with the modelling of tasks and problem solving methods. At the core of the SPO ontology is the concept of *TaskInAmbiance* (see Figure 3), which is defined in terms of *input* and *output roles*, a *goal specification*, the *ambiances* in which it is exposed, and an optional *closing time* and *closing condition*. Input and output roles define the information types specifying the input and output of a task. A goal defines a condition that the output needs to fulfil, in order for the task to be achieved, and it is represented in the ontology as a subclass of *owl:ObjectProperty*, i.e., as a meta-property. This representation allows us to use specific instances of class *GoalProperty*, i.e., specific goal-defining object properties, to represent the goal of a particular task. The optional closing time and closing condition are used to specify precise end points (as a time point or as a logical condition) for other smart products to consider, if they are willing to provide information or tackle the task. In particular closing conditions are also modelled as meta-properties, using the same mechanism used for representing goals. If no closing time or condition are given, the goal specification provides a default closing condition for a task.

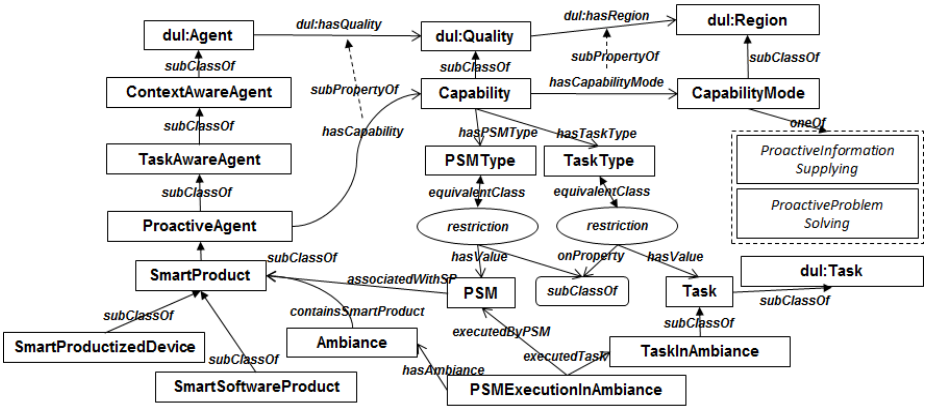


Fig. 3. Main classes and relations for modelling problem solving knowledge

A task is solved by a *Problem Solving Method (PSMInAmbiance)*, which defines a procedure to solve a class of tasks and is defined as a subclass of *dul:Plan*. An *ApplicabilityCondition* can be specified for a PSM[9] to determine whether it can be applied to a task in a specific problem solving context. An *ApplicabilityCondition* for a PSM is defined as a relation object with two arguments: *TaskType*, which defines the class to which the method can be applied (either the actual class or a set of restrictions describing a class definition) and *AmbianceType*, which defines a class of

[9] For the sake of brevity, in the rest of this paper we will use the terms *Task* and *PSM* as synonyms for *TaskInAmbiance* and *PSMInAmbiance*.

ambiances in which the method can be executed. *TaskType* and *AmbianceType* are defined as meta-classes: i.e., their instances are themselves classes, which are subclasses of *Task* and *Ambiance* respectively. For instance, an applicability condition may refer to the class of tasks *MealPlanningTask* and restrict the class of ambiances to those which belong to the specific user (i.e., have the value of *hasOwner* property set to a specific user ID).

A *SmartProduct* is defined as both a *Product* and a *ProactiveAgent*. Two subclasses of *SmartProduct* are considered in the ontology, *SmartProductizedDevice* and *SmartSoftwareProduct*, to cater for both physical and software products, as pointed out in Footnote 2. A relation, *hasCapability*, is used to define a *Capability* for a *ProactiveAgent* (and therefore for a *SmartProduct*). A *Capability* is defined as a quality of an agent and represented as a tripartite relation object *<PSMType, CapabilityMode, TaskType>*, where *TaskType* describes the class of tasks the agent can contribute to solve, *PSMType* describes the method the agent in question will apply, to tackle the instance of *TaskType,* and *CapabilityMode* specifies the modality by which the agent can contribute to solving the task. Currently, we consider three types of *CapabilityMode:*

- *ProactiveProblemSolving*. This value specifies that the method can be applied directly to solve the task.
- *ProactiveInformationSupplying*. This value specifies that the method provides information relevant to the task execution, by modifying some aspect of the task –typically the input roles.
- *TaskExposing*. This value refers to the generic capability of an agent to expose a task to an ambiance and is associated with a default method for exposing tasks.

The class *Ambiance* is used to specify networks of smart products, given that collaboration between smart products is only allowed within a particular ambiance. The following properties are defined for class *Ambiance*:

- *containsSmartProduct:* links the ambiance to a smart product, which is currently in the ambiance in question.
- *hasOwner:* links the ambiance to its (human) administrator.
- *hasJoiningPolicy:* links the ambiance to a joining policy descriptor. There can be several descriptors defined for the same ambiance.

Joining policies are necessary in order to regulate the inclusion of products into the ambiances. For example, the owner might not want products belonging to non-trusted users to join her home ambiance. Moreover, she may want to restrict certain capabilities of products within it. For instance, within a supermarket ambiance, it may not be desirable to allow smart products belonging to customers (mobile devices) to advertise arbitrary tasks to other customers' smart products.

5 Realizing Networks of Proactive Smart Products

Here we describe how the conceptual framework presented in Section 4 (the *knowledge level*) has been realized at *symbol level*. In particular, we focus on two

critical technical design issues: i) the realization of a protocol implementing proactive, distributed problem-solving over networks of peer-to-peer devices, and ii) the realization of the semantic data management components needed to store and manipulate knowledge in smart products.

5.1 The SmartProducts Task Messaging Protocol

To achieve a peer-to-peer model where all smart products are connected to each other in a network, we implemented a system architecture where smart products are instantiated as applications deployed on mobile computing devices, specifically smartphones and tablet devices running Android OS. For the actual communication between smart products, we chose the *MundoCore* [16] communication middleware, which provides facilities, such as creation of network *zones,* which can be readily mapped to the concept of ambiance in our framework. MundoCore also supports the notion of *channels* and implements a *publish/subscribe mechanism* to enable agents to interact and exchange messages within a zone. These features (zones and channels) were used to realize ambiances as networks within which tasks, their outputs and related information can be exchanged between smart products. Moreover, MundoCore also allows devices to subscribe to multiple channels in multiple zones, therefore allowing smart products to participate in more than one ambiance at a time –e.g., the *Shopping Assistant* in our scenario can be at the same time in the kitchen and in the supermarket ambiance. Finally, MundoCore supports 'device discovery' in such a way that it allows devices to join and leave the network at any time without having to reconfigure the network or the devices.

Besides the straightforward reuse of the zone and channel mechanisms, the realization of the distributed, proactive problem solving approach described in the previous sections was achieved by implementing a dedicated protocol on top of MundoCore, which we refer to as the *SmartProducts Task Messaging Protocol* (*SPTM*). SPTM implements the proactive task-based problem solving approach described earlier by relying on a coordination mechanism similar to the one used in *contract-nets* [17].

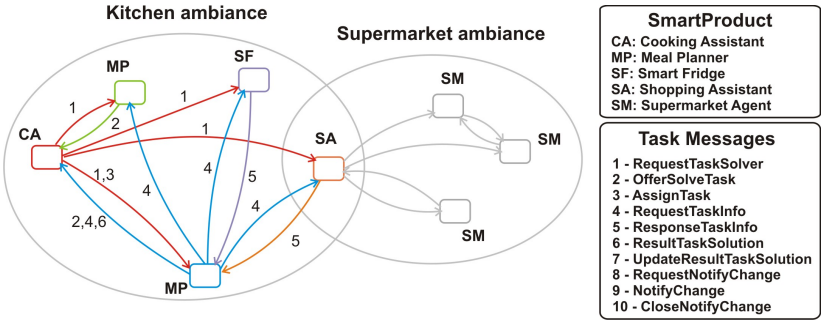


Fig. 4. SmartProducts Task Messaging protocol

As shown in Figure 4, SPTM provides mechanisms i) to broadcast and assign tasks; ii) to request and gather contributions to tasks; iii) to handle the management of tasks within multiple ambiances –e.g., the *Shopping Assistant* is able to receive

messages in both the kitchen and supermarket ambiances; and iv) to react to events affecting ongoing tasks in an ambiance –e.g., a change in the content of the fridge triggering an alert. In particular, Figure 4 shows a subset of the flow of messages in our scenario, where the *Cooking Assistant* first broadcasts a meal planning task (type *RequestTaskSolver* in the figure), to which the *Meal Planner* responds by sending a message of type *OfferSolveTask*. The *Meal Planner* is then delegated to tackle the meal planning task (type *AssignTask* in the figure) and to this purpose it broadcasts a message of type *RequestTaskInfo*, to which the *Smart Fridge* responds by providing information about its contents (type *ResponseTaskInfo*).

Since tasks are sent to the environment without pre-compiled knowledge or assumptions about the capabilities of other smart products, they carry associated closure conditions –in the simplest form, a time delay. The protocol is relatively lightweight, as it relies on only 10 types of messages (6 of which are shown in the diagram on the left-hand-side of Figure 4), and we have not observed any significant messaging overheads compared to similar designs based on the contract-net protocol.

The implementation of the conceptual framework for distributed proactive problem solving into the concrete SPTM protocol allows us to achieve a number of useful properties, as will be discussed in detail in Section 6. In a nutshell, this implementation reflects the flexibility of the framework, making it possible to add smart products to an ambiance without any need for reconfiguration of the network. The implementation also optimizes the distribution of knowledge amongst smart products, as information is stored locally in each node of the network and exchanged only when needed for a specific task.

5.2 Semantic Data Management Infrastructure

A key issue related to the realization of our approach in concrete networks of smart products concerns the need to equip resource-limited devices, such as mobile phones, with the ability to store and reason with semantic data. In [18] we compared different frameworks for semantic data management on resource-limited devices, and showed that small to medium scale data could be handled adequately on modern smartphones. In particular, we chose the *Sesame*[10] triple-store for its low requirements in terms of memory usage. Unsurprisingly, this study also showed that computational demands increased linearly with the amount of data stored, and with the use of advanced features, such as embedded inference engines. Hence, this study provided us with a basis to assess the amount of resources needed for a particular smart product depending on its required local knowledge and capabilities. For example the *Smart Fridge* only requires the limited resources that can be found on a smartphone, while the *Cooking Assistant*, which applies potentially complex inferences on thousands of cooking recipes, requires the extra memory and CPU capacity typically available on a tablet device.

Consistently with these findings, we developed a modular, Sesame-based architecture which can easily be scaled-up or down depending on the specific needs of a smart product and the resources available on a device, while ensuring a homogeneous architecture across heterogeneous devices. In particular we adapted the Sesame triple store for the Android system, and successfully applied it, through

[10] `http://www.openrdf.org/`

dedicated wrappers, on Android smartphones and tablet devices [19]. If required, this infrastructure component can be extended by enabling basic ontology reasoning or, when needed, a dedicated forward-chaining reasoner. In particular, following the results of the evaluation presented in [20], we have successfully used BaseVISor[11] to provide such inferencing support.

6 Evaluation

An evaluation of the SPO network of ontology can be found in [21]. Here, we focus instead on the evaluation of our technical architecture for realizing networks of smart products and we employ an approach based on the *ATAM methodology* [7], which is designed for evaluating architectures of collaborating agents. In particular, we consider the three following aspects[12], which are commonly used to evaluate distributed architectures –see also [22]:

(i) *Extensibility*: this aspect refers to the ability to add agents to the network which may implement additional capabilities.
(ii) *Reliability:* this aspect refers to the extent to which the failure of one or more agents in the network might affect the network as a whole.
(iii) *Security*: this aspect refers to the extent to which attacks on one or more elements of the network might expose the collective knowledge of the whole network.

In order to evaluate how our design choices impact on the properties considered above, we compare our architecture with the most obvious alternative solution, where ambiances are managed by a central 'facilitator' agent, which has the role of gathering the collective knowledge of the ambiance and providing matchmaking mechanisms to associate agents' needs to the capabilities of other agents. This solution differs from our approach because:

- It is *less distributed* –i.e., not *fully connected* [22]. In particular, the knowledge of each agent is not stored locally but at a central server.
- It is *reactive*, rather than proactive, as agents request features from other agents, rather than having smart products proactively contributing to shared tasks.

To evaluate the two alternative solutions, we consider four scenarios, which jointly cover the three properties mentioned earlier, and for each of them we assess qualitatively to what extent the two alternative architectures satisfy the relevant requirements. The scenarios have been chosen to illustrate generic situations that can be assessed independently from specific implementation details, and where the characteristics of the alternative architectures have a significant impact, as for example, when considering the security and privacy issues created by the introduction of a malevolent smart product.

[11] `http://vistology.com/basevisor/basevisor.html`

[12] Here we do not evaluate the *performance* of the architecture, as this is highly dependent on the implementation of each agent, as well as on the properties of the physical network used for communication between agents.

Scenario 1 (Extensibility): *Adding a smart product with capabilities and knowledge unknown to the network.*

In our framework, adding a smart product to an ambiance simply requires the corresponding device to join the peer-to-peer network. As knowledge is being held locally on the device and its capabilities used proactively to contribute to advertised tasks, there is no need to exchange any additional information. On the contrary, in a network where a facilitator maintains a directory of the available services/capabilities and aggregates the collective knowledge of the network, adding a smart product to an ambiance requires that the smart product registers its capabilities with the facilitator and constantly sends updates about newly generated knowledge. Besides the added complexity and communication overhead, this solution also requires that the facilitator either possesses a representation of any type of capability and knowledge that might be useful in a network, or is able to update its representation regularly, to comprise new capabilities and knowledge as they are introduced by smart products.

Extensibility in this sense is therefore one of the strong points of our proactive problem solving architecture: new smart products can contribute problem solving methods and knowledge directly as they join an ambiance, without the need for other agents in the network to have been explicitly programmed to manage such knowledge and capabilities.

Scenario 2 (Extensibility): *Generating an ambiance by aggregating smart products.*

Because it is based on peer-to-peer communication, our framework supports the ability to create new ambiances by simply aggregating smart products in a network. In particular, this means that *ad-hoc* ambiances can be created 'on the fly', for example to connect the *Shopping Assistants* of a group of shoppers to allow them to share dynamically and proactively information in a supermarket –e.g., about what they are buying, the location of products in the shop, etc. Again, it is obvious that realizing this scenario would be far more complex if we were relying on a centralized, global knowledge architecture, where a selected smart product plays a controlling or facilitating role. The network would disappear as soon as this particular device becomes unavailable.

Another important aspect is that, because a smart product may inhabit multiple ambiances at the same time, it can transfer knowledge generated in one ambiance to another one –e.g., the *Shopping Assistant* inhabits both the supermarket and the kitchen ambiance, thus being able to share knowledge about special offers with other smart products in the kitchen ambiance. If a centralized approach were used, communication between 'facilitators' would be needed to achieve this result, creating an overhead in the best case and being simply unfeasible in most realistic scenarios.

Scenario 3 (Reliability): *One of the smart products in the network suddenly stops being operational.*

In our framework, as in any other, the impact of a particular agent's failure depends on the type of the agent. If this were a smart product with only an information-providing capability, such as the *Smart Fridge*, the impact would only be that the knowledge it contains would stop being available to the rest of the network, resulting in sub-optimal decision making. However, if the failing agent has more sophisticated capabilities (such as providing methods for meal planning), such problem solving

capability would stop being available to the network, unless another smart product realizes a similar capability.

The situation is similar in networks relying on a facilitator agent, with some added complexity for the facilitator to handle situations in which registered features are requested, but are not available because of the corresponding device not being operational. A worst-case scenario in this type of network however is when the facilitator itself stops being operational, therefore invalidating all the capabilities and knowledge of all the smart products in the network.

Scenario 4 (Security): *A malevolent smart product is included in an ambiance.*

Here, we assume that a smart product has been created to join an ambiance in order to 'attack' it, meaning that its goal is to extract as much information as possible from the other agents in the network, or to disrupt collaborative problem solving. In the case of our framework, as all knowledge is localized in individual smart products and only shared when necessary for a particular task, the malevolent agent would need to be able to broadcast the right task and interpret properly the contributions from other agents to try and reconstruct the knowledge of other agents in the network. In addition, policies can be put in place on each device regarding the tasks they might contribute to, depending on the ambiance and the smart product that originated the task. For example, the *Shopping Assistant* might contribute to any task for which its capabilities are relevant in the kitchen ambiance, as it constitutes a trusted network, but may prefer to ignore tasks broadcast in the supermarket ambiance, as it has no reason to contribute there, and cannot verify whether these tasks are legitimate.

Of course, similar mechanisms can be put in place in the case of a facilitator-based network. Once again however, added complexity would be generated, as the facilitator would be required to implement mechanisms to consistently handle and manage policies for all smart products in the network. Obviously, the worst-case scenario here is when the problematic agent is the facilitator itself, since, as long as other agents can be tricked into joining its network, it would naturally collect all the knowledge of the other smart products in the ambiance. This is especially problematic in those cases where ad-hoc networks are formed (as discussed in Scenario 2), as one of the devices that might not be trustable will have to take the role of the facilitator, and could also potentially obtain information from the facilitators of other ambiances which have some devices in common with the one managed by the 'rogue' smart product.

7 Related Work

Proactive behaviour in artificial agents has been studied in the distributed AI community since the 70s and implementations of agents, which are able to exhibit proactivity, are often based on different variations of the *belief-desire-intention* framework (*BDI*) [23]. Problem-solving knowledge is usually decoupled into *goals* (what should be achieved) and *plans* (how to achieve it), in a similar way to the task-method decoupling in PSM research [11]. However, the notion of goal, while representing a necessary condition for achieving proactivity, does not *per se* reflect the behavioural patterns commonly associated with proactive behaviour –an agent can pursue a goal simply because it is asked to do so, as opposed to exhibiting proactivity, which requires that the agent actually takes the initiative in problem solving.

As already mentioned, an area of agent research, which specifically focuses on these issues, deals with the development of *user assistant agents* –e.g., see [2 8]. These studies consider proactivity as the capability of an agent "to anticipate needs, opportunities, and problems, and then act on its own initiative to address them" [2].

Our approach differs from the aforementioned ones in several respects. First, the use of standard Semantic Web representation makes it easier to integrate additional domain-specific information in our applications and take it into account during reasoning –e.g., as we do with information about food and recipes in our smart kitchen application [24]. Second, our approach involves exposing tasks as part of the shared context information, rather than by direct pairwise communication between agents. The reason for this is the need to deal with open environments, in which agents do not have prescribed roles. Thus, additional reasoning about whom to tell certain information is avoided and, while broadcasting may be considered in principle less efficient than direct agent to agent communication, in practice we have not found this to be an issue and we expect that even with reasonably large networks, our solution is unlikely to cause performance issues. In addition, we would also claim that our approach is more suitable for resource-constrained environments, as it uses more lightweight decision models than those used in most theories based on the BDI framework. In particular, we believe that unless we consider application-specific proactivity [2], where strong task models and learning mechanisms can be realized, our approach, which only requires task-based collaboration and does away with reasoning about other agents' beliefs and desires, provides a more 'agile' architecture to realise collaborative and proactive problem solving in networks of smart products.

In the ubiquitous computing area several approaches involving the use of ontologies and Semantic Web technologies have emerged, and some of them model the agent's activities [25 26]. However, these approaches pay less attention to the capabilities aspect, and the corresponding context broker implementations choose actions to perform using condition-action rules. The ontology developed in the CoDAMoS project [27] models user tasks and activities, as well as services provided by devices. Similarly, in the AMIGO project[12] process modelling is based on the notion of services, and a standard process representation ontology (OWL-S) is used to represent processes. These models allow matching tasks with device functionalities/services and representing process decomposition structures. Thus, decisions about when to contribute to a task can be made. However, they do not consider different capability modes, nor the participation of agents to multiple ambiances.

Several works have also targeted the integration of semantic data in small and mobile devices. In [28] an ad-hoc mechanism for storing and querying semantic web data on a mobile phone running iOS is presented, while [29] and [30] also describe mechanisms to integrate the use of semantic data within Android mobile phones. However, these solutions are restricted to the storage and manipulation of semantic data within 'closed' applications, while our approach provides the mechanisms needed to allow devices to exchange semantic data. Moreover, in contrast with our solution, which integrates Sesame with BaseVisor, none of these works consider the integration of inference engines. Tools such as μOR [31] exist for lightweight ontological reasoning on small devices, but do not integrate with common semantic data management infrastructures or with other types of reasoners.

[12] http://www.hitech-projects.com/euprojects/amigo/

8 Conclusions

In this paper we have provided an extensive overview of our work on smart products, in particular presenting a computational framework for realizing networks of smart products. The architecture is fully implemented and a demo of the smart kitchen application can be found at http://projects.kmi.open.ac.uk/smartproducts/demos/. For the future we plan to extend this work by investigating the augmentation of smart products with 'social intelligence', e.g., to enable them to act as 'social mediators' between users in open ambiances, such as a supermarket. In parallel we are also discussing with commercial partners the deployment of our architecture in large retail settings, where the ability for smart products to engage in proactive problem solving promises to open up new opportunities for customer-centric services.

References

1. Mühlhäuser, M.: Smart Products: An Introduction. In: Mühlhäuser, M., et al. (eds.) AmI 2007 Workshops. CCIS, vol. 32, pp. 158–164. Springer, Heidelberg (2008)
2. Yorke-Smith, N., Saadati, S., Myers, K., Morley, D.: Like an Intuitive and Courteous Butler: A Proactive Personal Agent for Task Management. In: Eighth Int. Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2009), Budapest, Hungary (2009)
3. Vildjiounaite, E., Kantorovitch, J., Kyllönen, V., Niskanen, I., et al.: Designing Socially Acceptable Multimodal Interaction in Cooking Assistants. In: International Conference on Intelligent User Interfaces (IUI 2011), Palo Alto (2011)
4. Beckerle, M., Martucci, L.A., Ries, S.: Interactive Access Rule Learning: Generating Adapted Access Rule Sets. In: Second International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE 2010), Lisbon, Portugal (2010)
5. Miche, M., Baumann, K., Golenzer, J., Brogle, M.: A Simulation Model for Evaluating Distributed Storage Services for Smart Product Systems. In: Puiatti, A., Gu, T. (eds.) MobiQuitous 2011. LNICST, vol. 104, pp. 162–173. Springer, Heidelberg (2012)
6. Schreiber, A.T.: Pragmatics of the Knowledge Level. Ph.D. Thesis, University of Amsterdam, `http://www.few.vu.nl/~guus/papers/Schreiber92c.pdf`
7. Woods, S., Barbacci, M.: Architectural evaluation of collaborative agent-based systems. Technical Report CMU/SEI-99-TR-025, SEI, Carnegie Mellon University, Pittsburgh, USA (1999), `http://www.sei.cmu.edu/reports/99tr025.pdf`
8. Maes, P.: Agents that reduce work and information overload. CACM 37(7), 30–40 (1994)
9. Zhang, Y., Volz, R. A., Loerger, T. R., Yen, J.: A decision-theoretic approach for designing proactive communication in multi-agent teamwork. In: SAC 2004, pp. 64–71 (2004)
10. Baumgartner, N., Gottesheim, W., Mitsch, S., Retschitzegger, W., Schwinger, W.: BeAware! - Situation awareness, the ontology-driven way. Data & Knowledge Engineering 69 (2010)
11. Motta, E.: Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving. IOS Press, Amsterdam (1999)
12. Chandrasekaran, B.: Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design. IEEE Expert 1(3), 23–30 (1986)
13. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., et al.: Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press (2000)

14. Murdock, J.W., Goel, A.K.: Meta-case-based reasoning: self-improvement through self-understanding. J. Exp. Theor. Artif. Intell. 20(1), 1–36 (2008)
15. Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., et al.: IRS-III: A Broker-based Approach to Semantic Web Services. Journal of Web Semantics 6(2), 109–132 (2008)
16. Aitenbichler, E., Kangasharju, J., Mühlhäuser, M.: MundoCore: A light-weight infrastructure for pervasive computing. Pervasive Mobile Computing 3(4), 332–361 (2007)
17. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers 29(12) (December 1980)
18. d'Aquin, M., Nikolov, A., Motta, E.: How Much Semantic Data on Small Devices? In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 565–575. Springer, Heidelberg (2010)
19. d'Aquin, M., Nikolov, A., Motta, E.: Building SPARQL-Enabled Applications with Android Devices. Demo at 10th International Semantic Web Conference (ISWC 2011) (2011)
20. Nikolov, A., Li, N., d'Aquin, M., Motta, E.: Evaluating semantic data infrastructure components for small devices. In: Int. Workshop on Evaluation of Semantic Technologies (IWEST 2010) at 9th International Semantic Web Conference (ISWC 2010) (2010)
21. Nikolov, A., d'Aquin, M., Li, N., Lopez, V., et al.: Evaluation of active components. SmartProducts Project Deliverable, D.2.5.1, `http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D2.5.1_Final.pdf`
22. Lee, S.K., Hwang, C.S.: Architecture modeling and evaluation for design of agent-based system. Journal of Systems and Software 72(2), 195–208 (2004)
23. Rao, A.S., George, M.P.: Modeling rational agents within a BDI-architecture. In: 2nd Int. Conference on Principles of Knowledge Representation and Reasoning (KR 1991) (1991)
24. Fernandez, M., Zang, Z., Lopez, V., Uren, V., Motta, E.: Ontology Augmentation: Towards Healthy Meal Planning. In: 6th Int. Conf. on Knowledge Capture (K-CAP 2011), Banff, Canada (2011)
25. Chen, H., Finin, T., Joshi, A.: The SOUPA Ontology for Pervasive Computing. In: Ontologies for Agents: Theory and Experiences, pp. 233–258. Birkhüuser (2005)
26. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology Based Context Modeling and Reasoning using OWL. In: 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18–22 (2004)
27. Preuveneers, D., et al.: Towards an extensible context ontology for ambient intelligence. In: 2nd European Symposium on Ambient Intelligence, pp. 148–159 (2004)
28. Weiss, C., Bernstein, A., Boccuzzo, S.: i-MoCo: Mobile conference guide – storing and querying huge amounts of Semantic Web data on the iPhone/iPod Touch. Billion Triple Challenge ISWC 2008, Karlsruhe, Germany (2008)
29. David, J., Euzenat, J.: Linked data from your pocket: The Android RDFContent-Provider. Demo at 9th International Semantic Web Conference (ISWC 2010) (2010)
30. Le-Phuoc, D., Parreira, J.X., Reynolds, V., Hauswirth, M.: Rdf on the go: A rdf storage and query processor for mobile devices. Demo at 9th International Semantic Web Conference (ISWC 2010) (2010)
31. Ali, S., Kiefer, S.: μOR – A Micro OWL DL Reasoner for Ambient Intelligent Devices. In: Abdennadher, N., Petcu, D. (eds.) GPC 2009. LNCS, vol. 5529, pp. 305–316. Springer, Heidelberg (2009)

LODStats – An Extensible Framework for High-Performance Dataset Analytics

Sören Auer, Jan Demter, Michael Martin, and Jens Lehmann

AKSW/BIS, Universität Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org

Abstract. One of the major obstacles for a wider usage of web data is the difficulty to obtain a clear picture of the available datasets. In order to reuse, link, revise or query a dataset published on the Web it is important to know the structure, coverage and coherence of the data. In order to obtain such information we developed LODStats – a statement-stream-based approach for gathering comprehensive statistics about datasets adhering to the Resource Description Framework (RDF). LODStats is based on the declarative description of statistical dataset characteristics. Its main advantages over other approaches are a smaller memory footprint and significantly better performance and scalability. We integrated LODStats with the CKAN dataset metadata registry and obtained a comprehensive picture of the current state of a significant part of the Data Web.

1 Introduction

For assessing the state of the Web of Data in general, for evaluating the quality of individual datasets as well as for tracking the progress of Web data publishing and integration it is of paramount importance to gather comprehensive statistics on datasets describing their internal structure and external cohesion. We even deem the difficulty to obtain a clear picture of the available datasets to be a major obstacle for a wider usage of the Web of Data. In order to reuse, link, revise or query a dataset published on the Web it is important to know the structure, coverage and coherence of the data.

In this article we present LODStats – a statement-stream-based approach for gathering comprehensive statistics from resources adhering to the Resource Description Framework (RDF). One rationale for the development of LODStats is the computation of statistics for resources from the Comprehensive Knowledge Archive (CKAN, "The Data Hub"[1]) on a regular basis. Datasets from CKAN are available either serialised as a file (in RDF/XML, N-Triples and other formats) or via SPARQL endpoints. Serialised datasets containing more than a few million triples tend to be too large for most existing analysis approaches as the size of the dataset or its representation as a graph exceeds the available main memory, where the complete dataset is commonly stored for statistical processing. LODStats' main advantage when compared to existing approaches is its

[1] http://thedatahub.org

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 353–362, 2012.

superior performance, especially for large datasets with many millions of triples, while keeping its extensibility with novel analytical criteria straightforward. It comes with a set of 32 different statistics, amongst others are those covering the statistical criteria defined by the Vocabulary of Interlinked Datasets [1] (VoID). Examples of available statistics are property usage, vocabulary usage, datatypes used and average length of string literals. Our implementation is written in *Python* and available as a module for integration with other projects.

Obtaining *comprehensive* statistical analyses about datasets facilitates a number of important use cases and provides crucial benefits. These include:

Quality analysis. A major problem when using Linked Data is quality. However, the quality of the datasets itself is not so much a problem as assessing and evaluating the expected quality and deciding whether it is sufficient for a certain application. Also, on the traditional Web we have very varying quality, but means were established (e.g. page rank) to assess the quality of information on the document web. In order to establish similar measures on the Web of Data it is crucial to assess datasets with regard to incoming and outgoing links, but also regarding the used vocabularies, properties, adherence to property range restrictions, their values etc. Hence, a statistical analysis of datasets can provide important insights with regard to the expectable quality.

Coverage analysis. Similarly important as quality is the coverage a certain dataset provides. We can distinguish *vertical* and *horizontal* coverage. The former providing information about the properties we can expect with the data instances, while the later determines the range (e.g. spatial, temporal) of identifying properties. In the case of spatial data, for example, we would like to know the region the dataset covers, which can be easily derived from minimum, maximum and average of longitude and latitude properties (horizontal coverage). In the case of organizational data we would like to determine whether a dataset contains detailed address information (vertical coverage).

Privacy analysis. For quickly deciding whether a dataset potentially containing personal information can be published on the Data Web, we need to get a quick overview on the information contained in the dataset without looking at every individual data record. An analysis and summary of all the properties and classes used in a dataset can quickly reveal the type of information and thus prevent the violation of privacy rules.

Link target identification. Establishing links between datasets is a fundamental requirement for many Linked Data applications (e.g. data integration and fusion). Meanwhile, there are a number of tools available which support the automatic generation of links (e.g. [11,10]). An obstacle for the broad use of these tools is, however, the difficulty to identify suitable link targets on the Data Web. By attaching proper statistics about the internal structure of a dataset (in particular about the used vocabularies, properties etc.) it will be dramatically simplified to quickly identify suitable target datasets for linking. For example,

the usage of longitude and latitude properties in a dataset indicates that this dataset might be a good candidate for linking spatial objects. If we additionally know the minimum, maximum and average values for these properties, we can even identify datasets which are suitable link targets for a certain region.

The contributions of our work are in particular: **(1)** A *declarative representation of statistical dataset criteria*, which allows a straightforward extension and implementation of analytical dataset processors and additional criteria (Section 2.1). **(2)** A comprehensive *survey of statistical dataset criteria* derived from RDF data model elements, survey and combination of criteria from related work and expert interviews (Section 2.2). **(3)** A LODStats *reference implementation* (Section 3), which outperforms the state-of-the-art on average by 30-300% and which allows to generate a statistic view on the complete Data Web in just a few hours of processing time. We provide an overview on related work in the areas of RDF statistics, stream processing and Data Web analytics in Section 4 and conclude with an outlook on future work in Section 5.

2 Statistical Criteria

In this section we devise a definition for statistical dataset criteria, survey analytical dataset statistics and explain how statistics can be represented in RDF.

2.1 Definition

The rationale for devising a declarative definition of statistical criteria is that it facilitates *understandability and semantic clarity* (since criteria are well defined without requiring code to be analyzed to understand what is actually computed), *extensibility* (as a statistical dataset processor can generate statistics which are defined after design and compile time) and to some extent *scalability* (because the definition can be designed such that statistical analyses can be performed efficiently). This definition formalizes our concept of a statistical criteria:

Definition 1 (Statistical criteria). *A statistical criterion is a triple* (F, D, P)*, where:*

- F *is a SPARQL filter condition.*
- D *is a data structure for storing intermediate results and a description how to fill this data structure with values from the triple stream after applying* F*.*
- P *is a post-processing filter operating on the data structure* D*.*

Explanations: F serves as selector to determine whether a certain triple triggers the alteration of a criteria. We use single triple patterns (instead of graph patterns) and additional filter conditions to allow an efficient stream processing of the datasets. The dataset is processed triple by triple and each triple read is matched against each triple pattern of each criterion.[2] With the filter condition

[2] In fact many criteria are triggered by the same triple patterns and have thus to be tested only once, which is used as an optimization in our implementation.

we can further constrain the application of a criteria, e.g. only to triples having a literal as object (using `isLiteral(?o)`).

For the data structure D, we usually make use of some counter code referring to variables of the triple pattern, for example, `H[ns(?subject)]++`, where H is a hash map and the function `ns` returns the namespace of the IRI supplied as a parameter. The counter code is an assertion of values to certain elements of the hash table D. Our survey of statistical criteria revealed that simple arithmetics, string concatenation (and in few cases the ternary operator) are sufficient to cover many purposes. Of course there are tasks for which LODStats is not suitable, e.g. measuring data duplication via string similarities.

In the simplest case P just returns exactly the values from the data structure D, but it can also retrieve the top-k elements of D or perform certain additional computations. In most cases, however, post-processing is not required.

Example 1 (Subject vocabularies criterion). This example illustrates the statistical criterion *subject vocabularies*, which collects a list of the 100 vocabularies mostly used in the subjects of the triples of an RDF dataset.

- *Criterion name:* Subject vocabularies
- *Description:* Lists all vocabularies used in subjects together with their occurrence
- *Filter clause:* - (empty)
- *Counter data structure:* hash map (initially empty)
- *Counter code:* `H[ns(?subject)]++` (`ns` is a function returning the namespace of the IRI given as parameter, i.e. the part of the IRI after the last occurrence of '/' or '#')
- *Post-processing filter:* `top(H,100)`

Statistical criteria can also be understood as a rule-based formalism. In this case, F is the condition (body of the rule) and D an action (head of the rule). P is only applied after executing such a rule on all triples. In Table 1, we present the more compact rule syntax of our statistical criteria to save space. Statistical criteria are evaluated in our framework according to Algorithm 1. Note that the triple input stream can be derived from various sources, specifically RDF files in different syntactic formats and SPARQL endpoints.

2.2 Statistical Criteria Survey

In order to obtain a set of statistical criteria which is as complete as possible, we derived criteria from:

(1) *analysing RDF data model elements*, i.e. possible elements as subjects, predicates and objects in an RDF statement, composition of IRIs (comprising namespaces) and literals (comprising datatypes and language tags, **(2)** *surveying and combining statistical criteria from related work* particularly from VoID and RDFStats [9], **(3)** *expert interviews*, which we performed with representatives from the AKSW research group and the LOD2 project.

Algorithm 1. Evaluation of a set of statistical criteria on a triple stream.

```
Data: CriteriaSet CS; TripleInputStream S
forall the Criteria C ∈ CS do
    initialise datastructures
while S.hasNext() do
    Triple T = S.next();
    forall the Criteria C ∈ CS do
        if T satisfies C.T and C.F then
            execute C.D;
            if datastructures exceed threshold then
                purge datastructures;
forall the Criteria C ∈ CS do
    execute C.P and return results
```

While collecting and describing criteria, we put an emphasis on *generality* and *minimality*. Hence, we tried to identify criteria which are as general as possible, so that more specialized criteria can be automatically derived. For example, collecting minimum and maximum values for all numeric and date time property values also allows us to determine the spatial or temporal coverage of the dataset, by just extracting values from the result list for spatial and temporal properties. The 32 statistical criteria that we obtained can be roughly divided in schema and data level ones. A formal representation using the definition above is given in Table 1. A complete list of all criteria and detailed textual explanations is available from the LODStats project page[3].

Schema Level. LODStats can collect comprehensive statistics on the schema elements (i.e. classes, properties) defined and used in a dataset. LODStats is also able to analyse more complex schema level characteristics such as the class hierarchy depth. In this case we store the depth position of each encountered class, for example, in a hash table data structure. In fact, since the position can not be determined when reading a particular `rdfs:subClassOf` triple, we store that the hierarchy depth of the subject is one level more than the one of the object. The exact values then have to be computed in the post-processing step. This example already illustrates that despite its focus on efficiency in certain cases the intermediate data structure or the time required for its post-processing might get very large. For such cases (when certain pre-configured memory thresholds are exceeded), we implemented an approximate statistic where the anticipated least popular information stored in our intermediate data structure is purged. Such a proceeding guarantees that statistics can be computed efficiently even if datasets are adversely structured (i.e. a very large dataset containing deep class hierarchies such as the the NCBI Cancer ontology). Results are in such cases appropriately marked to be approximate.

[3] `http://aksw.org/Projects/LODStats`

Table 1. Definition of schema level statistical criteria. Notation conventions: G = directed graph; M = map; S = set; i, len = integer. `+=` and `++` denote standard additions on those structures, i.e. adding edges to a graph, increasing the value of the key of a map, adding elements to a set and incrementing an integer value. `iris` takes a set as input and all elements of it, which are IRIs. `ns` returns the namespace of a resource . `len` returns the length of a string. `language` and `dtype` return datatype resp. language tag of a literal. `top(M,n)` return first `n` elements of the map `M`.

	Criterion	Rules (Filter → Action)		Postproc.
1	used classes	`?p=rdf:type && isIRI(?o)`	→ `S += ?o`	–
2	class usage count	`?p=rdf:type && isIRI(?o)`	→ `M[?o]++`	`top(M,100)`
3	classes defined	`?p=rdf:type && isIRI(?s) && (?o=rdfs:Class||?o=owl:Class)`	→ `S += ?s`	–
4	class hierarchy depth	`?p = rdfs:subClassOf && isIRI(?s) && isIRI(?o)`	→ `G += (?s,?o)`	`hasCycles(G) ? ∞ : depth(G)`
5	property usage		→ `M[?p]++`	`top(M,100)`
6	property usage distinct per subj.		→ `M[?s] += ?p`	`sum(M)`
7	property usage distinct per obj.		→ `M[?o] += ?p`	`sum(M)`
8	properties distinct per subj.		→ `M[?s] += ?p`	`sum(M)/size(M)`
9	properties distinct per obj.		→ `M[?o] += ?p`	`sum(M)/size(M)`
10	outdegree		→ `M[?s]++`	`sum(M)/size(M)`
11	indegree		→ `M[?o]++`	`sum(M)/size(M)`
12	property hierarchy depth	`?p=rdfs:subPropertyOf && isIRI(?s) && isIRI(?o)`	→ `G += (?s,?o)`	`hasCycles(G) ? ∞ : depth(G)`
13	subclass usage	`?p = rdfs:subClassOf`	→ `i++`	–
14	triples		→ `i++`	–
15	entities mentioned		→ `i+=size(iris( {?s,?p,?o}))`	–
16	distinct entities		→ `S+=iris({?s,?p,?o})`	–
17	literals	`isLiteral(?o)`	→ `i++`	–
18	blanks as subj.	`isBlank(?s)`	→ `i++`	–
19	blanks as obj.	`isBlank(?o)`	→ `i++`	–
20	datatypes	`isLiteral(?o)`	→ `M[dtype(?o)]++`	–
21	languages	`isLiteral(?o)`	→ `H[language(?o)]++`	–
22	∅ typed string length	`isLiteral(?o) && datatype(?o)=xsd:string`	→ `i++; len+=len(?o)`	`len/i`
23	∅ untyped string length	`isLiteral(?o) && datatype(?o) = NULL`	→ `i++; len+=len(?o)`	`len/i`
24	typed subj.	`?p = rdf:type`	→ `i++`	–
25	labeled subj.	`?p = rdfs:label`	→ `i++`	–
26	sameAs	`?p = owl:sameAs`	→ `i++`	–
27	links	`ns(?s) != ns(?o)`	→ `M[ns(?s)+ns(?o)]++`	–
28	max per property {int,float,time}	`datatype(?o)={xsd:int| xsd:float|xsd:datetime}`	→ `M[?p]=max( M[?p],?o)`	–
29	∅ per property {int,float,time}	`datatype(?o)={xsd:int| xsd:float|xsd:datetime}`	→ `M[?p]+=?o; M2(?p)++`	`M[?p]/M2[?p]`
30	subj. vocabularies		→ `M[ns(?s)]++`	–
31	pred. vocabularies		→ `M[ns(?p)]++`	–
32	obj. vocabularies		→ `M[ns(?o)]++`	–

Data Level. In addition to schema level statistics we collect all kinds of data level ones. As the simplest statistical criterion, the number of all triples seen is counted. Furthermore, entities (triples with a resource as subject), triples with blanks as subject or object, triples with literals, typed subjects, labeled subjects and triples defining an `owl:sameAs` link are counted. A list of the different datatypes used for literals, i.e. string, integer etc., can be compiled by LODStats. Data about which languages and how often they are used with literals by the dataset is made available to the user. If string or untyped literals are used by the dataset, their overall average string length can be computed. Statistics about internal and external links (i.e. triples where the namespace of the object differs from the subject namespace) are also collected. Spatial and temporal statistics can be easily computed using the min/max/avg. per integer/float/time property.

2.3 Representing Dataset Statistics with VoiD and Data Cube

Currently, 32 different statistical criteria can be gathered with LODStats. To represent these statistics we used the *Vocabulary of Interlinked Datasets* (VoID, [1]) and the *Data Cube Vocabulary* [8]. VoID is a vocabulary for expressing metadata about RDF datasets comprising properties to represent a set of relatively simple statistics. DataCube is a vocabulary based on the SDMX standard and especially designed for representing complex statistics about observations in a multidimensional attribute space. Due to the fact that many of the listed 32 criteria are not easily representable with VoID we encode them using the Data Cube Vocabulary, which allows to encode statistical criteria using arbitrary attribute dimensions. To support linking between a respective `void:Dataset` and a `qb:Observation`, we simply extended VoID with the object property `void-ext:observation`.

Using the default configuration of LODStats the following criteria are gathered and represented using the listed VoID properties: *triples* (`void:triples`), *entities* (`void:entities`), *distinct resources* (`void:distinctSubjects`), *distinct objects* (`void:distinctObjects`), *classes defined* (`void:classes`), *properties defined* (`void:properties`), *vocabulary usage* (`void:vocabulary`). Additional criteria such as *class/property usage*, *class/property usage distinct per subject*, *property usage distinct per object* are represented using `void:classPartition` and `void:propertyPartition` as subsets of `void:document`.

3 LODStats Architecture and Implementation

LODStats is written in Python and uses the Redland library [4] and its bindings to Python to parse files and process them statement by statement. Resources reachable via HTTP, contained in archives (e.g. zip, tar) or compressed with gzip or bzip2 are transparently handled by LODStats. *SPARQLWrapper*[4] is used for augmenting LODStats with support for SPARQL endpoints. Statistical criteria may also have an associated equivalent SPARQL query that will be used in

[4] `http://sparql-wrapper.sourceforge.net/`

case a certain dataset is not available in serialised form. Classes for gathering statistics support this by implementing a method with one or more SPARQL queries that produce results equivalent to those gathered using the statement-based approach. However, during our experiments it turned out that this variant is relatively error prone due to timeouts and resource limits. A second option we experimentally evaluated is to emulate the statement-based approach by passing through all triples stored in the SPARQL endpoint using queries retrieving a certain dataset window with `LIMIT` and `OFFSET`. Note that this option seems to work generally in practice, but requires the use of an (arbitrary) `ORDER BY` clause to return deterministic results. Consequently, this option also did not always render satisfactory results, especially for larger datasets due to latency and resource restrictions. However, our declarative criteria definition allows SPARQL endpoint operators to easily generate statistics right on their infrastructure. We plan to discuss support for the generation, publishing, advertising and discovery of statistical metadata by directly integrating respective modules into triple store systems.

LODStats has been implemented as a Python module with simple calling conventions, aiming at general reusability. It is available for integration with the Comprehensive Knowledge Archiving Network (CKAN), a widely used dataset metadata repository, either as a patch or as an external web application using CKAN's API. Integration with other (non-Python) projects is also possible via a command line interface and a RESTful web service.

4 Related Work

In this section we provide an overview on related work in the areas of RDF statistics, stream processing and Data Web analytics.

RDF Statistics. Little work has been done in the area of RDF statistics, mainly including *make-void*[5] and *rdfstats* [9]. *Make-void* is written in Java and utilizes the Jena toolkit to import RDF data and generate statistics conforming to VoID using SPARQL queries via Jena's ARQ SPARQL processor. *RDFStats* uses the same programming environment and library for processing RDF as make-void. It does not aim at generating VoID, but rather uses the collected statistics for optimising query execution.

RDF Stream Processing and SPARQL Stream Querying. Processing data datum by datum is commonplace in general (e.g. SAX[6]) and especially was so before the advent of computing machinery with larger main memory and software facilitating the retrieval of specific data. Processing RDF resources statement by statement is not a new concept as well; most RDF serialisation parsers internally work this way. Redland [4] additionally offers a public interface (`Parser.parse_as_stream`) for parsing and working with file streams in this manner. Furthermore, there are approaches for querying RDF streams with SPARQL

[5] https://github.com/cygri/make-void
[6] http://www.saxproject.org

such as *Continous SPARQL* (C-SPARQL) [3] and *Streaming SPARQL* [6]. Both represent adaptions of SPARQL which enable the observation of RDF stream windows (most recent triples of RDF streams). A further advancement, *EP-SPARQL* [2], is a unified language for event processing and stream reasoning. The main difference between these approaches and LODStats is that LODStats works on single triple patterns, thus does not require processing windows and offers even higher performance while limiting the processing capabilities. Such a processing limitation is sensible for the generation of dataset statistics, but not acceptable for general purpose stream processing.

Data Web analytics. Since 2007 the growth of the Linked Open Data Cloud is being examined. As a result, a visualization of the Linked Data space, its distribution and coherence are periodically published[7]. The main difference to `stats.lod2.eu` is that this information is partially entered manually in *The Data Hub* and updated infrequently, whereas using LODStats we can perform those calculations automatically. Both approaches lead to interesting information on the structure of the LOD cloud. Statistics about the Linked Data Cloud are summarized in [5], containing multiple aspects such as the usage of vocabularies as well as provenance and licensing information in published datasets. Furthermore, a comprehensive analysis of datasets indexed by the semantic web search engine Sindice[8] was published in [7]. It also contains a set of low-level statistical information such as the amount of entities, statements, literals and blank nodes. Unlike Sindice, which also indexes individual Web pages and small RDF files, LODStats focuses on larger datasets.

Another related area is data quality. One of the purposes of collecting statistics about data is to improve their quality. For instance, vocabulary reuse is improved, since `stats.lod2.eu` allows to easily check which existing classes and properties are widely used already. Since we do not directly pursue an improvement of data quality in this article, but focus on dataset analytics, we refrain from referring to the large body of literature in the data quality area.

5 Conclusions

With LODStats we developed an extensible and scalable approach for large-scale dataset analytics. By integrating LODStats with CKAN (the metadata home of the LOD Cloud) we aim to provide a timely and comprehensive picture of the current state of the Data Web. It turns out that many other statistics are actually overly optimistic – the amount of really usable RDF data on the Web might be an order of magnitude lower than what other statistics suggest. Frequent problems that we encountered are in particular outages and restricted access to or non-standard behavior of SPARQL endpoints, serialization and packaging problems, syntax errors and outdated download links. Some of these problems

[7] `http://lod-cloud.net/state/`
[8] `http://sindice.com/`

can also be attributed to CKAN entries not being sufficiently up-to-date. However, even though a dataset might be available at a different URL this poor user experience might be one of the reasons why Linked Data technology is still often perceived being too immature for industrial-strength applications. We hope that LODStats can contribute to overcoming this obstacle by giving dataset providers and stakeholders a more qualitative, quantitative and timely picture of the state of the Data Web as well as its evolution.

Future Work. While LODStats already delivers decent performance, a direct implementation of the approach in C/C++ and parallelization efforts might render additional performance boosts. Preliminary experimentation shows that such an approach would result in an additional performance increase by a factor of 2-3. Our declarative criteria definition allows SPARQL endpoint operators to easily generate statistics right on their infrastructure. We plan to integrate support for the generation, publishing, advertising and discovery of statistical metadata by directly integrating respective modules into the triple store systems (our Python implementation can serve as reference implementation). Last but not least we plan to increase the support for domain specific criteria, which can be defined, published and discovered using the Linked Data approach itself. For example, for geo-spatial datasets criteria could be defined, which determine the distribution or density of objects of a certain type in certain regions.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. In: 2nd WS on Linked Data on the Web, Madrid, Spain (April 2009)
2. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: WWW. ACM (2011)
3. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying rdf streams with C-SPARQL. SIGMOD Record 39(1), 20–26 (2010)
4. Beckett, D.: The design and implementation of the redland rdf application framework. In: Proc. of 10th Int. World Wide Web Conf., pp. 449–456. ACM (2001)
5. Bizer, C., Jentzsch, A., Cyganiak, R.: State of the LOD Cloud, Version 0.3 (September 2011)
6. 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)
7. Campinas, S., Ceccarelli, D., Perry, T.E., Delbru, R., Balog, K., Tummarello, G.: The Sindice-2011 dataset for entity-oriented search in the web of data. In: 1st Int. Workshop on Entity-Oriented Search (EOS), pp. 26–32 (2011)
8. Cyganiak, R., Reynolds, D., Tennison, J.: The rdf data cube vocabulary (2012), http://www.w3.org/TR/vocab-data-cube/
9. Langegger, A., Wöß, W.: Rdfstats - an extensible rdf statistics generator and library. In: DEXA Workshops, pp. 79–83. IEEE Computer Society (2009)
10. Ngonga Ngomo, A.-C., Auer, S.: Limes - a time-efficient approach for large-scale link discovery on the web of data. In: Proc. of IJCAI (2011)
11. 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)

Implementing an Automated Ventilation Guideline Using the Semantic Wiki KnowWE

Reinhard Hatko[1], Dirk Schädler[2], Stefan Mersmann[3], Joachim Baumeister[4], Norbert Weiler[2], and Frank Puppe[1]

[1] Institute of Computer Science, University of Würzburg, Germany
{hatko, puppe}@informatik.uni-wuerzburg.de
[2] Department of Anesthesiology and Intensive Care Medicine, University Medical Center Schleswig-Holstein, Campus Kiel, Kiel, Germany
{dirk.schaedler, norbert.weiler}@uk-sh.de
[3] Dräger Medical GmbH, Lübeck, Germany
stefan.mersmann@draeger.com
[4] denkbares GmbH, Würzburg, Germany
joachim.baumeister@denkbares.com

Abstract. In this paper, we report on the experiences made during the implementation of a therapeutic process, i.e. a guideline, for automated mechanical ventilation of patients in intensive care units. The semantic wiki KnowWE was used as a collaborative development platform for domain specialists, knowledge and software engineers, and reviewers. We applied the graphical guideline language DiaFlux to represent medical expertise about mechanical ventilation in a flowchart-oriented manner. Finally, the computerized guideline was embedded seamlessly into a mechanical ventilator for autonomous execution.

1 Introduction

Using expert system technologies to build knowledge-based applications for the medical domain is beneficial, as these systems can improve therapy quality and decrease costs at the same time [3]. SmartCare® by Dräger Medical, Germany, is a methodology and technology to efficiently develop such knowledge-based applications with a high degree of versatility [9]. Similar to SmartCare, most of these applications use a rule-based approach, which appeared to be difficult to maintain and to extend during their entire life cycles. Guideline-based approaches were proposed to represent health care processes, so called computer-interpretable guidelines [4]. Such graphical representations seem to be more suitable in terms of knowledge acquisition and maintenance, since domain specialists are able to understand and to modify existing knowledge bases more easily. With the DiaFlux language, we introduced a graphical knowledge representation that is based on the process metaphor, but is still capable to model detailed decision knowledge [5]. Embedded into a semantic wiki system, a community-based collaboration is feasible and enables distributed knowledge engineering. The available tool support for DiaFlux takes further practical requirements into account: Knowledge engineering tools need to be accessible, understandable, and flexible. The term

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 363–372, 2012.

accessible means, that knowledge engineers as well as domain specialists need to be able to access the most current state of the knowledge base at any time. Also, the tool should present the knowledge base in an *understandable* manner, which significantly influences the efficiency during maintenance and evolution of domain knowledge. Last but not least, the tool should be *flexibly* adaptable to the user's established process of knowledge acquisition and maintenance. This includes a flexible organisation, representation, and modification of the knowledge base. Semantic wikis are an appropriate tool for fulfilling these requirements, as it was argued and demonstrated [1]. The semantic wiki KnowWE [2] offers an authoring system for the DiaFlux language, providing a suitable environment for the development of medical knowledge bases.

In Section 2, we introduce the medical application of automated mechanical ventilation. The development environment is described in Section 3. We report on practical experiences with the implementation of a clinical guideline in Section 4. The paper concludes with a summary in Section 5.

2 A Clinical Guideline for Mechanical Ventilation

Mechanical ventilation is a life-saving therapy for patients who are temporarily not or not fully able to breathe on their own. It is frequently applied in the field of critical care in so called intensive care units. It has been recognized that mechanical ventilation can lead to lung injury. Therefore, the major therapeutic goals are i) to reduce total ventilation time and ii) to find optimal ventilator settings resulting in minimal lung injury.

Clinical Guidelines (CGs) can be viewed as best practices in health care for a certain medical realm, following an underlying therapeutic strategy [3]. Typically, a CG depicts a goal-oriented, detailed therapeutical process for a specific intended use, e.g. mechanical ventilation. Non-computerized use of CGs decreases ventilation time as well as human errors [8]. However, a transfer of CGs into daily clinical practice is difficult [11]. Besides high expenses for a thorough inauguration and training, representation format and usability aspects are focal points to cope with. Apparently, computer applications, especially knowledge-based systems for the execution of CGs, can help to overcome these difficulties.

SmartCare/PS is a computerized CG for weaning patients from mechanical ventilation. This system was incorporated into a commercial ventilator [9] and was successfully evaluated in several clinical studies, e.g., [10]. But the intended use of this system is rather limited since it adjusts only one of at least four ventilator settings automatically. Hence, we developed a CG for mechanical ventilation that provides decision support (open-loop) or automated control (closed-loop) of an intensive care ventilator (Evita XL, Dräger Medical GmbH, Germany). This CG is able to automatically adjust the whole range of ventilator settings, i.e., inspired fraction of oxygen, breathing frequency, inspiration time, positive end-expiratory pressure, inspiratory pressure, and pressure support.

3 Computerizing Clinical Guidelines with KnowWE

In this section, we give a short introduction into the semantic wiki KnowWE and the computer-interpretable guideline language DiaFlux.

3.1 The Semantic Wiki KnowWE

Wikis became famous as a successful means for knowledge aggregation in an evolutionary 'self-organizing' way by possibly large and open communities. Semantic wikis extend the wiki approach by adding formal representations of the wiki content. While many semantic wikis provide means to create and populate light-weight ontologies, the approach can be generalized to create any kind of formal knowledge base. The knowledge engineering approach described in this paper focuses on the use of a semantic wiki to develop medical decision-support systems. The wiki is used as a tool for creating and maintaining formal concepts and relations, and informal knowledge containing their documentation. Each time the content is edited, the formal knowledge sections are processed by the wiki engine, updating the executable knowledge base accordingly. Different knowledge formalization patterns (e.g., DiaFlux models, rules, ...) are supported by the system, and can be captured by the use of various markups, cf. Figure 1. Following the principle of *freedom of structuring* proposed by wikis, the system does not constrain where formal knowledge should be defined. Formal concepts and relations can be used in any wiki article at any location, using the markup defined by the current system settings. KnowWE also provides components to test the current version of a knowledge base. The major strength of wikis is the generally low barrier for contribution due to its simple editing mechanism. However, the definition of a formal knowledge base using textual markups is a complicated task. Autonomous contribution by domain specialists still can be achieved using a step-wise formalization process, employing the metaphor of the knowledge formalization continuum [1]. The web-based collaborative access provided by wikis supports this evolutionary process.

3.2 The DiaFlux Guideline Language

Clinical guidelines are a means to improve patient outcome by offering a standardized treatment, based on evidence-based medicine. The increasing computerization and data availability, also in domains with high-frequency data as, e.g., intensive care units, allow for an automation of guideline application by medical devices [9]. Several formalisms for computer-interpretable guidelines have been developed, each one with its own focus [4]. Most of them are graphical approaches, that employ a kind of task network model to express the guideline steps. However, in the area of closed-loop devices, rule-based approaches are predominant, e.g., [7,9]. A downside of rule-based representations is their lower comprehensibility compared to graphical ones. This especially holds true, as medical experts are typically involved in the creation of guidelines. Therefore, we developed a graphical guideline formalism called DiaFlux [5]. Its focus lies

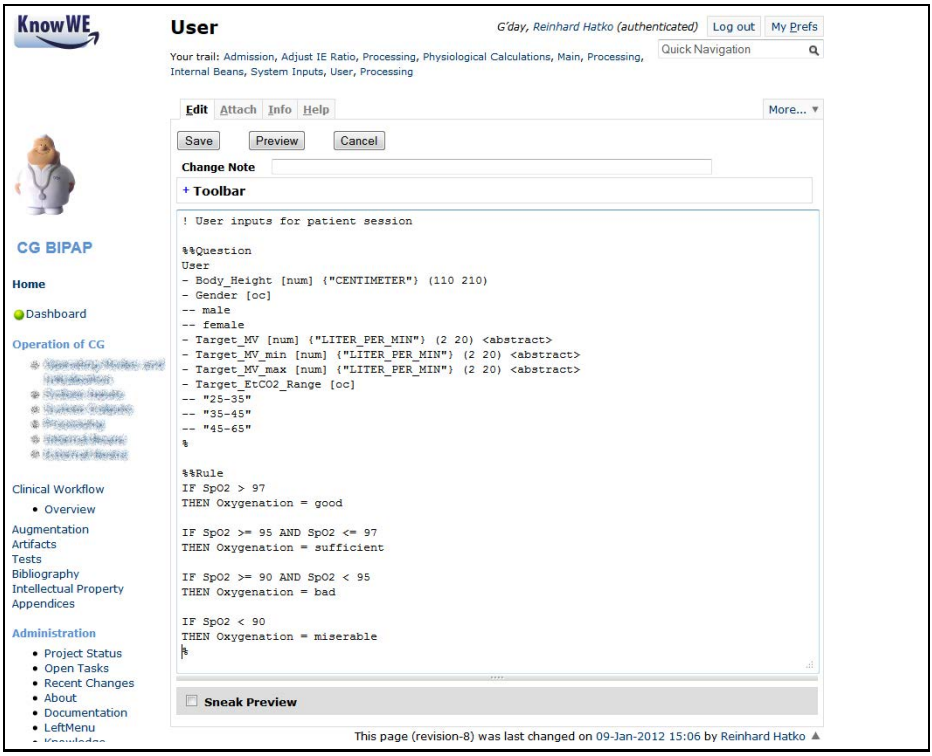


Fig. 1. The special markup for creating (parts of) the terminology and one set of abstraction rules for the derivation of a higher-level abstraction (in edit mode)

on the instant applicability and understandability by domain specialists. The main application area of DiaFlux guidelines are mixed-initiative devices that continuously monitor, diagnose, and treat a patient in the setting of an intensive care unit. Such closed-loop systems interact with the clinical user during the process of care. The clinician as well as the device are able to initiate actions on the patient. Data is continuously available as a result of the monitoring task. It is then used for repeated reasoning about the patient. Therapeutic actions are carried out to improve her or his state, if applicable.

Language Overview. To specify a clinical guideline, two different types of knowledge have to be effectively combined, namely declarative and procedural knowledge [4]. The declarative part contains the facts of a given domain, i.e., findings, diagnoses, treatments and their interrelations. The knowledge of how to perform a task, i.e., an appropriate sequence of actions, is expressed within the procedural knowledge. In DiaFlux models, the declarative knowledge is represented by a domain-specific ontology, which contains the definitions of findings and diagnoses. This application ontology is an extension of the task ontology of diagnostic problem-solving [2]. The ontology is strongly formalized and

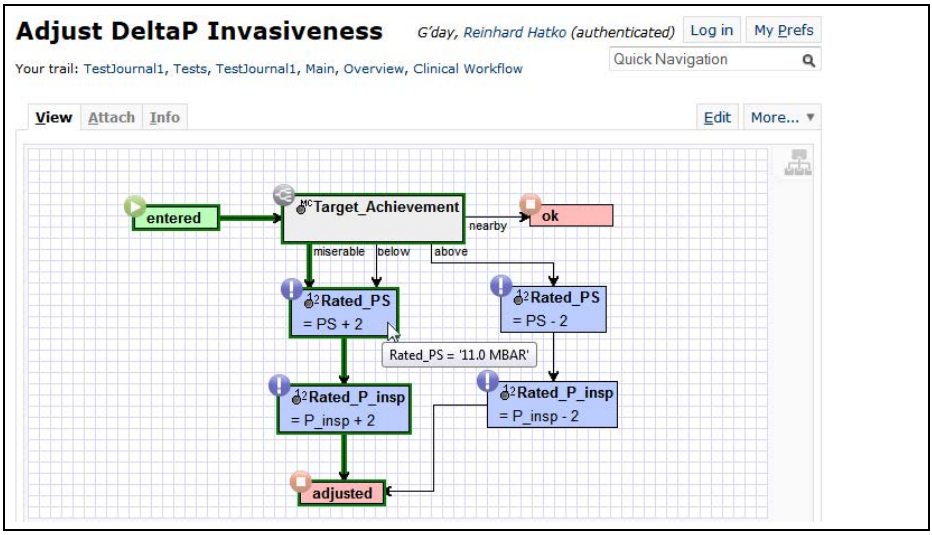


Fig. 2. A DiaFlux model for adjusting two ventilatory settings. The current execution status is highlighted (the leftmost, bold path). A tooltip shows the newly derived value.

provides the necessary semantics for executing the guideline. Like most graphical guideline languages, DiaFlux employs flowcharts as its task network model. They describe decisions, actions, and constraints about their ordering in a guideline plan. These flowcharts consist of nodes and connecting edges. Nodes represent different kinds of actions. Edges connect nodes to create paths of possible actions. Edges can be guarded by conditions, that evaluate the state of the current patient session, and thus guide the course of the care process.

Guideline Execution. The architecture of the DiaFlux guideline execution engine consists of three components. First, a knowledge base, that contains the application ontology and the DiaFlux models. Second, a blackboard, that stores all findings about the current patient session. Third, a reasoner, that executes the guideline and carries out its steps, depending on the current state as given by the contents of the blackboard. Therefore, the reasoner is notified about all findings, that enter the blackboard. A more thorough introduction to the DiaFlux language and its execution engine is given in [5].

The execution of the guideline has to be initiated by a clinical user. During the startup procedure, she or he provides certain information about the patient, e.g., gender and body height. Subsequently, the execution is time-driven. Each reasoning cycle starts by acquiring and interpreting data. Then, the guideline is executed. This may generate hints to the clinical user and may initiate therapeutic actions by the device. Finally, the time of the next execution is scheduled. Until then, the effects of the actions are monitored.

4 Experiences with Implementing the CG in KnowWE

We created an implementation of the automated ventilation CG described in Section 2. The semantic wiki KnowWE was used as the knowledge engineering environment and DiaFlux as the modeling language for the clinical process. The basis for our efforts were a complete description of the guideline by textual documents and a prototypal implementation using a rule-based system. The textual documents contained: First, a specification of the environment the guideline is intended to run on (e.g., preconditions on ventilatory settings and patient state), and second, the guideline algorithm, depicted as flowcharts using standard office visualization software, mainly created by the domain specialists. The rule-based implementation was an earlier proof-of-concept prototype and was used as a reference during the first phase of the re-implementation.

4.1 Structure of the Knowledge Base

The structure of the knowledge base can be categorized as follows:

- *Inputs:* The inputs for the knowledge base consist of the current settings of the ventilator (e.g., frequency, pressure), the measured patient responses (e.g., oxygen saturation) and certain patient characteristics (e.g., gender).
- *Abstractions:* Abstractions are used to store aggregated values of input data. These are simple calculations, like ratios of two inputs, as well as higher-level assessments of the patient state, e.g. the current lung mechanic.
- *Outputs:* During each iteration of the CG execution, new values for each of the ventilator settings may be calculated, in order to improve the mechanical ventilation. Operating in closed-loop mode, they are applied automatically.
- *Abstraction rules:* For each of the defined abstractions, a small set of rules exists. Those rules derive the abstraction value based on the current and older values of the inputs, e.g., to detect trends.
- *DiaFlux models:* The main part of the knowledge base consists of several hierarchically nested DiaFlux models. These flowcharts are directly executable and implement the ventilation strategy of the system.

The first three categories form the declarative part of the knowledge base. They are created using a special markup (cf. Figure 1) and represent the application-specific ontology [2]. The procedural knowledge consists of several DiaFlux models, that rely on the defined ontology. Overall, the knowledge base consists of 17 DiaFlux models, which contain 295 nodes and 345 edges in total.

4.2 Knowledge Engineering Process

Development Approach. In a prior project [6], we used a top-down development approach for creating a guideline from scratch, in accordance to the ideas of the knowledge formalization continuum [1]. Due to the mature stage of this CG, we employed a bottom-up approach instead. First, the terminology (inputs, abstractions and outputs) was prepared on various wiki articles. Then,

we implemented the abstraction rules (cf. Figure 1). Finally, the DiaFlux models were created, based on the readily available terminology. Furthermore, the complete documentation, contained in the textual documents, was transferred to KnowWE. The DiaFlux models completely replaced the graphical specification of the guideline. Therefore, the redundancy between specification and separate implementation was removed. All changes made to the flowcharts were directly executable without an additional implementation step by a knowledge engineer.

Unit Tests for Abstraction Rules. For validating the abstraction rules, we created unit-test-like test cases, to ensure their correctness. These test cases provided the inputs necessary to derive exactly one type of abstraction, decoupled from the rest of the knowledge base, especially from the clinical process, that uses them. This step checked, that the abstractions were correctly derived, before they were used to model the clinical process.

Systematic Creation of Test Cases. As soon as a first stable version was available, we systematically created test cases for the guideline. At first, we created cases for individual paths through the guideline, e.g. early aborts of the execution, in case the patient is not admittable. These small ones were created by manually entering data into the system. For each test case an individual wiki article was created, containing the test case itself, informal information about the tested aspect of the guideline, and a customized workbench for tracking down errors or unexpected behavior (cf. Figure 3).

Due to numerous dependencies between inputs and abstractions, the manual creation of test cases is tedious. This approach was not feasible for scenarios, like the reduction of one ventilator setting for certain patient states. Due to their complexity, these tests were created by running the guideline against an external software, that simulates a mechanically ventilated patient. It employs a physiological lung model to determine the effects of the current ventilation settings on the patient. It is able to deliver the necessary data (ventilation settings and measured patient state) to the guideline execution engine. Such a generated patient session is saved to a file, which can be attached to a wiki article and replayed for introspecting the guideline execution. The current state is visualized by highlighting the active path of the guideline (cf. Figure 2) and the abstraction rules, that fired. This explanation was easy to comprehend for the domain specialists, following the requirement for an accessible knowledge representation.

Clinical Evaluation Using a Patient Simulator. Clinically evaluating safety and performance of new medical systems in general and of computerized CGs in particular is essential and demanded by several laws and regulating authorities. Yet, clinical studies are extremely expensive and time consuming. We solve this issue by using a so called Human Patient Simulator (METI, Sarasota, USA) to conduct initial validation activities in an almost-real-life environment, cf. Figure 4. Such a simulator can provide all physiological systems of interest, e.g., respiratory and cardio-vascular system, fluid management, medication, and many

Time	Gender	Body_Height	Target_MV	Target_MV_min	Target_MV_max	Target_EtCO2_Range	T_insp	F	PEEP	P_in
0s	male	186.0	11.0	3.0	20.0	35-45	2.0	12.0	7.0	18.0
30s			11.0	3.0	20.0	35-45	2.0	12.0	7.0	18.0
2min 30s			11.0	3.0	20.0	35-45	2.1	11.0	7.0	21.0
7min 30s			9.9	3.0	20.0	35-45	2.1	11.0	7.0	21.0
12min			9.9	3.0	20.0	35-45	2.1	11.0	7.0	18.0

Fig. 3. A customized debugging workbench for one particular test case, showing the course of a value, a selection of current values, and the player for executing the test

others. During the clinical evaluation, the system was tested successfully in four programmed patient types and 11 incidents (data not shown).

4.3 Experiences

In our scenario of geographically distributed domain specialists and knowledge engineers, the use of KnowWE and DiaFlux showed the following advantages:

Advantages of the Wiki-Based Environment. The wiki-based approach offered several advantages over the document-centric one together with its separate implementation. The first one is of organizational nature. The wiki system used was running on a secured server and was accessible to all participants, and thus allowed for a collaborative development. The most recent version of the guideline was available for every participant at any time, without distributing new versions of the documents after each change, e.g., via e-mail. This also avoided "update anomalies", that can occur, if participants modify the documents concurrently. Second, the wiki-based approach allowed for the creation of individual workbenches by using separate wiki articles, e.g., for each test case. Therefore, they were not only persistent over time, but also accessible to each participant. For example, a knowledge engineer could define a debugging workbench for a test

Fig. 4. The current testing environment of the developed CG, showing the Human Patient Simulator and the mechanical ventilator

case that showed unexpected behavior. Instantly, it was accessible to a domain specialist for further consultation. Third, the integrated versioning mechanism of the wiki allows to track all changes made to the knowledge base during the development.

Advantages of DiaFlux. One of the motivations for the development of the DiaFlux language was to create a model consisting of a limited number of intuitive language elements, that would improve the accessibility for domain specialists. The developed guidelines should be clearly understandable during design time as well as during their execution. For the modeling of a guideline, it is sufficient to be familiar with a few different types of nodes. As we offer a graphical editor for DiaFlux models, large parts of the guideline can be created using a well-known drag-and-drop metaphor, once the terminology has been created. For the introspection of the execution, the decisions and actions taken can intuitively be tracked by a highlighting of active paths. With this functionality, DiaFlux provides a more comprehensible explanation than, e.g., rule traces. Another advantage comes from the use of a single formalism for specifying and executing the guideline, in contrast to the previous graphical specification with its separate rule-based implementation. Separate models for these two aspects bear the risk of creating deviations from one another over time, especially if a knowledge engineer by himself transfers the changes made by domain specialists.

5 Conclusion

In this paper, we reported on the experiences made during the implementation of an automated clinical guideline for mechanical ventilation. The semantic wiki KnowWE was used for the development among a geographically distributed community of domain specialists and knowledge engineers. It successfully supported the collaborative implementation and testing process. The clinical guideline is

represented in the graphical guideline language DiaFlux, which proved to be more understandable to the domain specialists than previous approaches used. For clinicians, even graphical-oriented knowledge representations might appear rather technical. Hence, further activities should investigate acceptance, practicability, and usability of collaborative development environments.

Acknowledgements. University of Würzburg, University Medical Center Schleswig-Holstein, Campus Kiel, and Dräger Medical GmbH are funded by the German Federal Ministry for Education and Research under the project "WiMVent" (Knowledge- and Model-based Ventilation), grant number 01IB10002.

References

1. Baumeister, J., Reutelshoefer, J., Puppe, F.: Engineering intelligent systems on the knowledge formalization continuum. International Journal of Applied Mathematics and Computer Science (AMCS) 21(1) (2011)
2. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE: A semantic wiki for knowledge engineering. Applied Intelligence 35(3), 323–344 (2011)
3. Chatburn, R.L., Deem, S.: Should weaning protocols be used with all patients who receive mechanical ventilation? Respir. Care 52(5), 609–621 (2007)
4. de Clercq, P., Kaiser, K., Hasman, A.: Computer-interpretable guideline formalisms. In: ten Teije, A., Miksch, S., Lucas, P. (eds.) Computer-based Medical Guidelines and Protocols: A Primer and Current Trends, pp. 22–43. IOS Press, Amsterdam (2008)
5. Hatko, R., Baumeister, J., Belli, V., Puppe, F.: Diaflux: A Graphical Language for Computer-Interpretable Guidelines. In: Riaño, D., ten Teije, A., Miksch, S. (eds.) KR4HC 2011. LNCS, vol. 6924, pp. 94–107. Springer, Heidelberg (2012)
6. Hatko, R., Reutelshoefer, J., Baumeister, J., Puppe, F.: Modelling of diagnostic guideline knowledge in semantic wikis. In: Proceedings of the Workshop on Open Knowledge Models (OKM 2010) at the 17th International Conference on Knowledge Engineering and Knowledge Management, EKAW (2010)
7. Kwok, H.F., Linkens, D.A., Mahfouf, M., Mills, G.H.: Rule-base derivation for intensive care ventilator control using ANFIS. Artificial Intelligence in Medicine 29(3), 185–201 (2003)
8. MacIntyre, N.R.: Evidence-based guidelines for weaning and discontinuing ventilatory support. Chest 120(suppl. 6), 375S–396S (2001)
9. Mersmann, S., Dojat, M.: SmartCare$^{\text{tm}}$ - automated clinical guidelines in critical care. In: ECAI 2004/PAIS 2004: Proceedings of the 16th European Conference on Artificial Intelligence, pp. 745–749. IOS Press, Valencia (2004)
10. Schädler, D., Engel, C., Elke, G., Pulletz, S., Haake, N., Frerichs, I., Zick, G., Scholz, J., Weiler, N.: Automatic control of pressure support for ventilator weaning in surgical intensive care patients. American Journal of Respiratory and Critical Care Medicine 185(6), 637–644 (2012)
11. Weinert, C.R., Gross, C.R., Marinelli, W.A.: Impact of randomized trial results on acute lung injury ventilator therapy in teaching hospitals. American Journal of Respiratory and Critical Care Medicine 167(10), 1304–1309 (2003)

Knowledge Management on the Desktop

Laura Drăgan and Stefan Decker

Digital Enterprise Research Institute, National University of Ireland, Galway
firstname.lastname@deri.org

Abstract. Knowledge management on the desktop is not a recent challenge. It has been around one way or another ever since the desktop emerged as a life (and work)-changing device. It has been around even before that, foreseen by visionaries like Bush, Engelbart and Nelson. Their ideas for solutions have been taken up by many projects in the field of PIM and KM. Semantic Web technologies have been regarded as a game-changer, and applying them to PIM has resulted in the Semantic Desktop. Many Semantic Desktops have been created over time, each focusing on problems specific or generic, on restricted areas like email or task management, or on providing a general solution. Mostly they have not received the uptake they envisioned. This paper describes the representative Semantic Desktop systems. We explore their similarities, and what they do differently; the features they provide, as well as some common shortcomings and sensitive areas.

1 Challenges, Solutions and a Definition

The fast growth of the digital world, while solving some of the problems of access to information and communication, has created new problems, foreseen by Nelson in the 70s: the amount of information available is so big that it becomes unusable and unmanageable. In this avalanche of information, protecting the privacy and ensuring trust became much harder.

We start by listing some of the challenges users faces when working on their daily tasks with the current tools. We show how the tools that help, also get in the way, and while having lots of data to work with is nice, it is also an inconvenience to sift through it to extract the useful information.

Information overload caused by the ease with which information can now be generated and shared. The amount of information that we see and create every day is enormous. It is usually stored for later reference, never to be found again.

Data silos are created by desktop applications which store their data in closed formats, in repositories inaccessible to other applications which could possibly reuse that information, thus having to recreate it in their own closed formats and store it in their own walled repositories. It is a vicious circle, which started back when the things a computer could do were much more limited. It was perpetuated by the desire of ensuring the security of the data stored in applications. But the negative effects are *data duplication*, and *disconnected information*.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 373–382, 2012.

Trustworthy information is key. Information changes over time, so information which was correct a week ago might not still be correct, and using deprecated information can have unwanted effects. The data silos problem above adds complexity to this – duplication of data in various applications makes it more challenging to decide which version is correct; disconnected information means the same changes have to be made in several places.

Associative trails as those described by Bush, following the way we think about things, by connections and associations, unconstrained by the way the information is stored or presented in various applications. They are tightly related to the data silos issue above. As long as data is locked away in application data silos, the associative trails can only be fragmented and thus not very useful.

The Semantic Web has gained considerable momentum, and developments in semantic technologies flourished. It brought the promise of better interconnected information, which in turn should enable better organization, better search, easier management and increased security. Applying these technologies to the domain of Personal Information Management (PIM), and the desktop, resulted into the Semantic Desktop.

The term *Semantic Desktop* was coined in 2003 by Decker and Frank [1] and taken up by Sauermann. We present below some of the systems created to take advantage of semantics on the desktop. The same goals and expectations were embodied by these new systems and tools, as were by the Memex before them: supporting and improving knowledge work by mimicking the way the brain works — Bush's associative trails reworked with the help of the [newly emerged] semantic technologies.

Since then, other systems have emerged, targeting the issues described above, on the desktop or other devices, with the help of semantic technologies. However, the focus has changed from exploring possible architectures and creating vocabularies, to a more application centric approach. We can claim that the Semantic desktop has matured, along with the semantic technologies it employs, and now new and more exciting, as well as harder, problems arise. The infrastructure has been put in place for most parts, what the Semantic Desktop awaits now is the killer app which would bring it, and the possibilities it opens, into the public eye. Siri[1] and Evi[2], as well as IBM's Watson have opened the door.

2 Old Solutions to Old Problems

The ideas behind the Semantic Desktop have been around for much longer than the actual devices (personal computers, netbooks, tablets or smart phones) used to run them. The common starting point is finding better ways for knowledge workers to manage the ever growing amount of information they need to work with and process. The most relevant historical precursors of the Semantic Desktop are, in chronological order:

[1] http://www.apple.com/iphone/features/siri.html
[2] http://www.evi.com

Vannevar Bush's *memex*. Bush was deeply concerned with the continuation of the collaboration between scientists after the war was over. Himself a scientist, he was also painfully aware of how the progress in science meant that a lot more research results are published than can be followed, and that because of old and inadequate publishing systems, as well as the sheer amount of information, much of the new research is unreachable, lost or overlooked. In a lengthy essay titled As We May Think, published in 1945 in The Atlantic Monthly[2] and in Life, Bush describes several devices - possible inventions of the future, as solutions to the problems of research publishing.

The most famous of the devices described by the article is the *memex* — a vision for a truly personal system for interlinking information. The *memex* was "... a device in which an individual stores all his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility". Leaving aside the physical description of the device, which was envisioned in the form of a desk with screens, drawers for storage, a keyboard, buttons, and levers, Bush has more or less described the functions of a present day personal computer — a device where we store and consult personal documents and communications. Furthermore, the memex, attempts to mimic the way associations in the brain works, by using associative trails of connected things. The trails can be created, modified, browsed, shared and collaborated on. Associative indexing was the essential feature of the *memex*.

The elegant ideas behind the *memex* influenced the field of information science, information management, the development of personal computing, hypertext and semantic technologies. The *memex* is the first semantic desktop described in the literature, although it was never realized.

Douglas Engelbart's Augment and NLS, and Ted Nelson's Xanadu. The influence of Bush's memex on the works of Engelbart and Nelson is undisputed, and stated by both in their writings. The directions were different though: Engelbart NLS was designed by engineers, focused on their needs, and encouraging collaboration. Xanadu on the other hand was intended for personal use, for capturing the train of thought of the user and serve as an extended memory. Some of the features envisioned by Nelson for Xanadu were incorporated in Engelbart's NLS.

From a desire to "improve the lot of the human race", Doug Engelbart has dedicated his career to "augmenting the human intellect". His 1962 report[3] describes a conceptual framework for his research, introducing the hypothesis that "better concept structures can be developed – structures that when mapped into a human's mental structure will significantly improve his capability to comprehend and to find solutions within his complex problem situations." The report describes and motivates the need for a well designed semantic model for such a system. He envisions that the extra effort required "to form tags and links [...] consciously specifying and indicating categories" would be rewarded by the capacity gained by the computer to understand and perform more sophisticated tasks. Allowing the creation of arbitrary links between elements of the hierarchy, and providing unique names for entities for better and faster access, represents

the realization of Bush's associative trails and makes Engelbart's On-Line System (NLS) the first functional predecessor of the Semantic Desktop.

In parallel with Engelbart, Nelson has foreseen the information overload crisis we are facing, and many of the developments that created it: the personal computer, enhanced communications, digital publishing, virtually infinite storage. He has imagined a solution to the problem, embodied in a system called Xanadu, the specification and requirements of which are detailed as early as 1965 in [4].

3 Modern Semantic Desktops

Inspired by the vision of the *memex* and by the possibilities open by the advancing semantic technologies, many Semantic Desktops were created. The have the same common goals, and tackle the challenges we described above. Some cover a broad spectrum of functionalities, aiming for a general solution, while some are focused on precise PIM tasks like email or task management. We describe here the systems which aim to provide a general framework, not just specific activities.

The **Haystack** system started as a personal system for Information Retrieval, where users manage their own private information. From the beginning it was a semantic system, as it created and saved associations between things in the user's corpus, and allowed these links to be followed from one document to another. Storage for the user's content was initially a database, and became later an RDF store. Haystack provided a basic data model for the objects, metadata and links; later it moved to RDF for describing the data. The RDF model is general, but it allows customization by users[5].

MyLifeBits uses the *memex* as a blueprint for a digital personal store[6]. The system doesn't impose a strict single hierarchy over the user's data, but uses links and annotations to organize the objects, and collections to group them. The links are bidirectional, and serve as well to establish authorship/ownership of connected pieces of data. Annotations play an important role, and collections are a special type of annotation. MyLifeBits uses a flexible predefined schema to describe the data collected and provides multiple visualizations.

Gnowsis *Gnowsis* is one of the first implementations of a Semantic Desktop which from the very beginning advocates the use of Semantic Web technologies on the desktop, and creating a "personal Semantic Web for PIM"[7]. It proposes that existing applications are modified to work with the semantic infrastructure, rather than being replaced completely. The Semantic Desktop would play the role of integration middleware by lifting semantic data from desktop formats and storing it in a central repository accessible back to the applications. The semantic data is described with ontologies, and the system provides a generic personal information model (PIMO), which is rich enough to cover most use cases, but also flexible and customisable by the users.

IRIS ("Integrate, Relate, Infer, Search")[8] is a Semantic Desktop system for integration. IRIS was part of CALO ("Cognitive Assistant that Learns and

Organizes"), as a personal information knowledge source. IRIS uses modular ontologies to describe user's data. One of the key features of IRIS and CALO as a whole, is the focus on machine learning.

SEMEX (Semantic Explorer) provides on-the-fly integration of personal and public data[9]. It aligns the information integration task with the user's environment, making it happen as a side effect of the normal daily tasks. The system provides a basic domain ontology, which can be personalized by the users either by importing new models, or by manually changing it. Reference reconciliation plays an important role in the system, particularly for the on-the-fly integration.

MOSE (Multiple Ontology based Semantic DEsktop) [10] is a framework for PIM supporting several *personal information applications* (PIA) which are specialized on certain tasks, like trip planning or bibliography management. Each PIA has its own ontology to describe the domain knowledge, a user interface and a specific workflow. The PIAs can communicate and share data through mappings of their ontologies. MOSE stores its data in several repositories which are populated by the services of the framework, and by the PIAs. The data can be browsed by association, modified and queried through the resource explorer, a browser-like interface. Other user interfaces are provided by the PIAs, which themselves can be customised or created from scratch by the users.

X-COSIM (Cross-COntext Semantic Information Management) [11] framework supports seamless PIM and information linkage across different contexts that the users might find themselves in. The system provides a reference ontology called X-COSIMO which describes the various possible contexts and relations between the concepts and contexts. Like other systems above, X-COSIM provides a browser for the semantic data it handles. The semantic functionalities are integrated into existing applications through plugins.

NEPOMUK. Started as a big research project, involving partners from academia and industry, the Nepomuk project[12] set out to define the blueprint for a generic Semantic Desktop, based on previous research as well as new studies. Many of the pioneers of the Semantic Desktop research were involved, and many of the systems presented above were surveyed.

4 Discussion — Differences and Similarities

4.1 Architecture

Most of the systems above agree on the need for a layered, modular architecture for semantic desktops. A general architecture can be divided in three major layers which build on top of each other, and there are dependencies and even overlaps between them.

Data Layer. The Semantic Desktop revolves around the [semantic] data that it contains. As such, the architecture is also data centric. This layer contains the data models, and the data itself. We discuss in more detail about data representation in the next section.

Service Layer. Based on the data, the Semantic Desktop provides an enabling framework of basic services, which can be either visible or transparent to end users. This enabling framework enables the creation and functionalities of the end-user applications. The set of basic services vary among the systems described, but are indispensable to them. These services are central to the desktop, or generally accessible from all applications. Some of the basic services include storage, extraction, integration, query, inference, annotation.

Storage service can vary from a database system like MS SQL server used by MyLifeBits, to a fully fledged RDF store like Sesame or Virtuoso in Gnowsis and Nepomuk. Many systems use Jena (Semex, IRIS) and some use files. Some use a combination of storage from the options above (MOSE has three – database, file and Jena RDF store). Depending on the type of storage, semantic resources are identified either by URIs or by unique database IDs.

Extraction service can come under several names, crawlers, wrappers, gatherers — however, they provide the same function, that of extracting semantic information from non-semantic sources, structured or unstructured. It can vary from simple extractors of metadata already available for files, to parsing and analysing unstructured text to extract information from multiple file formats (SEMEX, Gnowsis, NEPOMUK). CALO features natural language analysis to extract entities and relations among them. MOSE also extracts resources and relations from files, and stores two types of information – R-F links, which specify which resource was extracted from which file, and R-R links, which specify connections between resources. Semantic information extraction plays a significant role in providing the basic functions of a Semantic Desktop, and all the systems described here implement it. The extraction service generally comes bundled together with an instance matching service.

Integration service (instance matching, entity matching) has the role of checking if two instances are the same. The definition of *the same* varies from system to system. One of the main uses of this service is complementing the functionality of the extraction service, by checking if a newly extracted entity already exists, in which case, depending on the policies the existing entity is reused instead of creating a copy, or a new entity is created and linked to the existing one.

Query service All the systems allow querying their semantic data. Keyword search is supported by IRIS, Semex, Haystack, MOSE. For the keyword based search, an *indexing service* is used, which can be an independent service of the frameworks, or part of the storage or the extraction service. Structured query languages like RDQL and SPARQL are also supported in MOSE, IRIS, Gnowsis, X-COSIM, NEPOMUK. The ways in which the results are displayed are discussed in the presentation layer.

Inference service Inference on the semantic data is supported by IRIS, Gnowsis aor nd NEPOMUK. This service might not be standalone, but included in the extraction or the integration service. The power of the inference results depend on the engine used and the ontologies used to describe the data.

Annotation service All systems allow some type of annotation of resources, however, not always in the form of a standalone service. Annotation refer to

creation of metadata for resources, or of new connections between resources. Manual creation of new resources can also considered annotation. Some automatic annotation is performed by the extraction and the integration services. In Haystack, where there is one access point to the data, the annotation service is in fact a user application, as it happens directly. MyLifeBits allows sharing, annotation of annotations. The most basic types of annotations are tagging and grouping in collections.

Presentation / Application Layer. The user-facing interface of the Semantic Desktop makes up the presentation layer, built on top of the supporting framework. The systems have a large variety of user interfaces, providing functionality that varies from simple resource browser, to complex PIM tools like email clients and task managers.

Regarding the applications they provide, the systems are divided in two distinct categories, depending on whether they choose to enhance existing applications with semantic capabilities, or propose new semantically enabled applications. X-COSIM, Gnowsis and NEPOMUK belong to the first category, while IRIS and MOSE belong to the second.

The flexible and customisable visualization of information is one of the distinguishing features of Haystack. The system is a proper Semantic Web browser, providing a unified interface for the data it contains, with the added functionality of allowing edits and customisations. The feature that sets Haystack apart from other semantic browsers is the dynamic creation of user interfaces[5]. This is realized by recursively rendering the semantic resources. The way an object should be rendered is described with RDF. General visualizations are provided out of the box, but users can customise them according to their needs.

Most systems provide a resource browser and search interface in the style of Haystack, although usually not as flexible. Some browser include the underlying ontology in the view. They allow changes to be made to the data directly through this interface. Faceted browsing and browsing by association are available in all resource browsers.

MyLifeBits and SEMEX propose that multiple visualizations be supported for resources, depending on the context of the user.

The applications are the actual relevant end-result of the Semantic Desktops. They are realizing / solving the initial challenges.

Blackboard and Fuzzy Layers. The storage service is at the fuzzy border between the data layer and the service layer. It is responsible with the storage of the data and providing low level access to the data. Since it is deeply connected with the data, it can be seen as part of the data layer, but at the same time it is a foundational service, and as such it is part of the service layer.

Services providing any type of user interfaces are at the border between the service layer and the presentation layer. Inside the service layer itself we can determine layers based on the level at which the services operate. Some foundation services are more oriented towards the data - like an inference service or the analyser / extractor services. Other services provide more user-oriented functionality, like an annotation service, or a query service.

Services can use functionality provided by other services, communicating and building on top of each other. The communication between the services, as well as the communication between the services and applications can take several forms: through programming interfaces, Web standards (Gnowsis promotes a Web server as a desktop service), or P2P communication (MOSE, NEPOMUK).

The architecture of many of the systems uses the Blackboard pattern, where the blackboard is the data storage, or even the entire data layer, which is accessible to all services and applications. They have the role of the specialists in the pattern. The specialists populate the blackboard with data which can then be accessed and refined by other specialists. For example, a PDF extractor service can parse documents and extract titles, authors and affiliations. The data must then be processed by the integration services, so that duplicate authors and affiliations can be merged into a single unique representation. Further more, the inference service can then extract co-working relations between the people, based on common affiliations.

4.2 Data Representation

The semantic data is the most important part of the frameworks, and the way it is described influences the quality and amount of things that can be done. All the systems define a data model for the data they extract. The data models vary from very small and generic, like the one in SEMEX, with a restricted number of classes and associations, to the comprehensive one provided by X-COSIM.

SEMEX's ontology is small so it is not split in modules, but the bigger ones usually are modular. MOSE's ontology is divided in small application ontologies belonging to the PIAs and domain ontologies used by the services. CALO also has different ontologies for specific domains and used by specialized agents. X-COSIMO is divided into modules representing different aspects of the data. More than being modular, the set of ontologies used by NEPOMUK is also layered. The representational level defines the vocabulary used to define the other ontologies. The upper-level ontologies contain domain-independent abstract knowledge. And the lower level ontologies contain specialized concrete terms from the user's mental model. The low level PIMO plays an important role in both Gnowsis and NEPOMUK, as it is used to describe the data most often handled by the user. Most of the frameworks use RDF or OWL ontologies to describe the data. MyLifeBits and SIS do not mention the use of ontologies,the former using a flexible database schema to define the data model.

Another differentiating characteristic is whether or not the data model can be personalised by the users by creating new types and new relations. X-COSIM does not allow the modification of the domain ontology, but it does allow the creation of custom mappings from and to other external ontologies. Haystack and SEMEX on the other hand argue for the need for personalization of the ontologies by the user, as only in this way, a truly personal mental can be created. Most systems do support the customisation of the underlying model, as well as the import of new vocabularies in the system, by linking to them or through mappings. This fact raises the challenge of reconciling data described with customised models.

4.3 Findings, Shortcomings, Developments

Following again the layered structure used before, we continue with a discussion of some of the shortcomings and possible developments which appear to affect all or most of the Semantic Desktops.

At the data level, as the systems have evolved the ontologies they employ became more complex. While providing good coverage of the PIM domain, in comprehensive vocabularies with detailed relationships among types showed that the simpler, more general relations are used much more often, regardless of the possible loss of meaning[13]. Hence, rich ontologies are not necessarily better, since most users prefer simple relations between resources, which are enough to remind them of the connection, without specifying it fully. Detailed vocabularies might prove more useful in the case of automatic processing of data, but using them to manually annotate is costly. The same long-term study of using Gnowsis has shown that although customisation of the underlying ontologies is supported, the users only make minimal and rare changes. This result confirms the MyLifeBits hypothesis that although their schema is flexible and can be modified, users will probably not make any changes to it.

Moving into the service layer, the storage and the indexing services provided by the systems use semantic technologies which have evolved at a rapid pace. A slow repository, and thus slow response times for queries and browsing have at least partially been at fault for the poor uptake of the Semantic Desktops. Fast and memory efficient triple stores are now available.

Regarding the applications provided by the systems, we observed the distinction between integrating semantics into existing applications versus creating new semantic applications. Forcing the users to switch from the applications they know to avail of the power of the Semantic Desktop in applications they would need to learn how to use, has not proved to be a successful strategy. However, for systems like Gnowsis, X-COSIM and NEPOMUK, which use plugins to add semantic functionality to existing popular tools, there seems to be other reasons for the slow growth in popularity.

One of the reasons could be the cold start problem, which is observable despite multiple automatic ways of extracting, linking and importing resources, and kick-starting the system. This could prove that the user's manual annotations are more important than the automatically extracted data, which has its own important role though. However, since the automated data comes from sources which are accessible to the user anyway through conventional tools, there is no immediate incentive to use the semantic applications, which translates in little manual information being added into the system, and the benefits delayed further, despite several evaluations[11,13] proving that "Semantic Desktops are better" for PIM tasks.

It could also be just that the visualizations used for the automatically extracted data are not suitable for the purpose or not attractive enough. Generic graph or table visualizations are not appealing, and treating every resource the same is not an effective way of conveying information.

In recent years two developments occurred which influence the direction in which the Semantic Desktops evolve: (i) the exponential growth of semantic data available online, and (ii) the growing concern about privacy and security of personal data.Some of the information available as Linked Data might be relevant to the users of Semantic Desktops, so using it in applications and services to add value to the users is a low hanging fruit, awaiting to be picked. However, the open aspect of most of the available data causes concern especially when it becomes mixed with valuable private personal information. Privacy is not the only concern, albeit a very important one. Establishing if the Web information is trustworthy is another concern, and possibly a harder one to tackle.

5 Conclusion

Since the Semantic Desktop has been invented, by applying Semantic Web technologies to PIM, many systems have been created, some of them presented here. From their experience evolved a framework which, in the spirit of the memex, supports the creation of a interconnected network of knowledge.

References

1. Decker, S., Frank, M.: The social semantic desktop. Technical Report (May 2004)
2. Bush, V.: As we think. The Atlantic Monthly 3(176), 101–108 (1945)
3. Engelbart, D.C.: Augmenting Human Intellect: A Conceptual Framework. Stanford Research Institute, Menlo Park (1962)
4. Nelson, T.H.: Complex information processing: a file structure for the complex, the changing and the indeterminate. In: Proc. of the 20th Nat. Conf., pp. 84–100 (1965)
5. Karger, D.R., Bakshi, K., Huynh, D., Quan, D., Sinha, V.: Haystack: A customizable general-purpose information management tool for end users of semistructured data. In: Proc. of the CIDR Conf. (2005)
6. Gemmell, J., Bell, G., Lueder, R., Francisco, S.: MyLifeBits: A Personal Database for Everything. Technical report (2006)
7. Sauermann, L.: The Gnowsis Semantic Desktop approach to Personal Information Management. PhD thesis (2009)
8. Cheyer, A., Park, J., Giuli, R.: IRIS: Integrate. Relate. Infer. Share. In: Proceedings of the 1st Workshop on the Semantic Desktop (2005)
9. Dong, X.L., Halevy, A., Nemes, E., Sigurdsson, S.B., Domingos, P.: SEMEX: Toward On-the-fly Personal Information Integration. In: IIWEB 2004 (2004)
10. Xiao, H., Cruz, I.F.: A multi-ontology approach for personal information management. In: Proc. of Semantic Desktop Workshop at the ISWC (2005)
11. Franz, T., Staab, S., Arndt, R.: The X-COSIM integration framework for a seamless semantic desktop. In: Proc. of the 4th K-CAP, p. 143 (2007)
12. Bernardi, A., Decker, S., Van Elst, L., Grimnes, G.A., Groza, T., Handschuh, S., Jazayeri, M., Mesnage, C., Moeller, K., Reif, G., Sintek, M.: The Social Semantic Desktop A New Paradigm Towards Deploying the Semantic Web on the Desktop. In: Semantic Web Engineering in the Knowledge Society. IGI Global (2008)
13. Sauermann, L., Heim, D.: Evaluating Long-Term Use of the Gnowsis Semantic Desktop for PIM. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 467–482. Springer, Heidelberg (2008)

Improving the Quality of SKOS Vocabularies with Skosify

Osma Suominen and Eero Hyvönen

Semantic Computing Research Group (SeCo)
Aalto University, Department of Media Technology
University of Helsinki, Department of Computer Science
firstname.lastname@aalto.fi
http://www.seco.tkk.fi/

Abstract. Simple Knowledge Organization System (SKOS) vocabularies are commonly used to represent lightweight conceptual vocabularies such as taxonomies, classifications and thesauri on the Web of Data. We identified 11 criteria for evaluating the validity and quality of SKOS vocabularies. We then analyzed 14 such vocabularies against the identified criteria and found most of them to contain structural errors. Our tool, **Skosify**, can be used to automatically validate SKOS vocabularies and correct many problems, helping to improve their quality and validity.

1 Introduction

Controlled vocabularies such as taxonomies, classifications, subject headings and thesauri [3] have been used for more than a century in the classification of books, music and other documents. In the Web era, the use of such vocabularies has greatly expanded into the description of personal interests, news categories, blog topics, art and literary styles, music genres and many other semantic attributes.

Such vocabularies are increasingly being published using the Simple Knowledge Organization System (SKOS) standard of describing vocabularies by means of RDF structures [14]. As an example, many library classifications have been published as SKOS vocabularies, allowing various library catalogs using those classifications to be published as Linked Data and then easily integrated using RDF tools [7,13,19], enabling applications such as semantic information retrieval over multiple datasets [8], query expansion and recommendation [16].

However, the benefits of SKOS in data integration are only realizable if the SKOS vocabulary data is structurally valid and makes use of the SKOS entities in a meaningful way. To this end, the SKOS reference [14] defines a number of *integrity conditions* that can be used to detect inconsistencies in a SKOS vocabulary. In addition, *validation tools* are available for verifying that a SKOS vocabulary follows generally accepted best practices for controlled vocabularies which have not been codified in the SKOS reference. Many SKOS vocabularies are currently published by automatically converting vocabularies from legacy formats into SKOS. Structural problems in the resulting SKOS files may be difficult to notice for vocabulary publishers, but may cause problems for users of the vocabularies.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 383–397, 2012.

In this study, we have surveyed the current quality and validity of published SKOS vocabularies and found many examples of structural problems. Our intent is to improve on the current state of standards compliance and quality of SKOS vocabularies published as RDF or Linked Data. We therefore seek answers for the following **research questions**:

1. What criteria can be used to validate a SKOS vocabulary?
2. How well do existing SKOS vocabularies fulfill those criteria?
3. How can SKOS vocabulary quality be improved?

To answer the first research question, we compiled a list of suitable validation criteria for SKOS vocabularies, detailed in Section 3. To answer the second research question, we used an online validation tool to validate thirteen SKOS vocabularies found on the web as well as one vocabulary produced in our own research group. This validation is detailed in Section 4. To answer the third research question, we created the Skosify tool to find and correct many kinds of inconsistencies and problems in SKOS vocabularies. The tool and the problems it can be used to correct are further described in Section 5.

2 Related Work

The SKOS reference specifies a number of *integrity conditions* which must be fulfilled for the vocabulary to be considered valid [14]. Many of these conditions are based on earlier standards for structuring controlled vocabularies and thesauri, including ISO 2788 [1] and the British standard BS8723 Part 2 [2]. These conditions may be considered a minimum set of validation and/or quality criteria for SKOS vocabularies; there are also many vocabulary-related best practices which go beyond the integrity conditions codified in SKOS. A more thorough set of quality criteria (hereafter known as the **qSKOS criteria**) for SKOS vocabularies [12] and a validation tool that can be used to measure vocabularies against these criteria has been developed in the **qSKOS project**[1]. On a more theoretical level, Nagy et al. have explored the various structural requirements of SKOS vocabularies in different application scenarios [16].

The **PoolParty online SKOS Consistency Checker**[2] (hereafter known as the **PoolParty checker**) performs many checks on SKOS vocabularies, including the SKOS integrity conditions. It also indicates whether the vocabulary can be imported into the online **PoolParty thesaurus editor** [18]. The W3C used to host a similar online SKOS validation service, but it was not kept up to date with the evolution of SKOS, and is no longer available.

More general validation services for RDF and Linked Data have also been developed. The **W3C RDF Validation Service**[3] can be used to verify the syntax of RDF documents. The **Vapour** [5] and **RDF:Alerts** [10] systems are

[1] `https://github.com/cmader/qSKOS/`
[2] `http://demo.semantic-web.at:8080/SkosServices/check`
[3] `http://www.w3.org/RDF/Validator/`

online validation tools intended to spot problems in Linked Data. A recent and thorough survey of general RDF and Linked Data validation tools is given in [10]; however, to our knowledge, none of these tools have any specific support for SKOS vocabularies.

3 Validation Criteria

In order to find a suitable set of criteria for checking the validity and quality of SKOS vocabularies, we compared three lists of such criteria: the integrity conditions defined in the SKOS reference [14], the validity checks performed by the PoolParty checker, and the list of quality criteria from the qSKOS project. The results of matching these criteria are shown in Table 1. The last column of the table indicates whether our tool, Skosify, can detect and correct each issue, a feature discussed in more detail in Section 5.

3.1 Selection of Criteria

We excluded several qSKOS criteria that were very context-specific and/or would be difficult to improve algorithmically. For example, the degree of external links (VQC4) and language coverage (VQC11b) in a vocabulary can be measured, but the results have to be interpreted in the context of the intended use of the vocabulary and would not be easy to improve using a computer program.

Table 1. Comparison of Validation Criteria for SKOS Vocabularies

Criterion name	SKOS	PoolParty checker	qSKOS	Skosify
Valid URIs	-	Valid URIs	(VQC5 is stricter)	not checked
Missing Language Tags	-	Missing Language Tags	VQC11a	corrected
Missing Labels	-	Missing Labels	VQC10	corrected partially
Loose Concepts	-	Loose Concepts	(VQC1 is stricter)	corrected
Disjoint OWL Classes	S9, S37	Disjoint OWL Classes	(VQC8)	corrected partially
Ambiguous `prefLabel` values	S14	Consistent Use of Labels	(VQC8)	corrected
Overlap in Disjoint Label Properties	S13	Consistent Use of Labels	VQC12, (VQC8)	corrected
Consistent Use of Mapping Properties	S46	Consistent Usage of Mapping Properties	(VQC8)	not checked
Disjoint Semantic Relations	S27	Consistent Usage of Semantic Relations	(VQC8)	corrected
Cycles in `broader` Hierarchy	-	-	VQC3	corrected
Extra Whitespace	-	-	-	corrected

The different lists of criteria use varying levels of granularity. For example, the PoolParty checker considers both the existence of more than one `prefLabel`[4] per language and the use of disjoint label properties as violating the Consistent Use of Labels check, while qSKOS has a single criterion (VCQ8 *SKOS semantic*

[4] Typographical note: words set in typewriter style that don't include a namespace prefix, such as `Concept` and `prefLabel`, refer to terms defined by SKOS [14].

consistency) which encompasses all kinds of violations of the SKOS integrity conditions. We tried to keep our list of criteria as granular as possible. However, the SKOS integrity conditions S9 and S37 both define related class disjointness axioms, so we chose not to separate between them.

Based on an analysis of the potential severity of quality issues and some previous experience of problems with vocabularies published on the ONKI vocabulary service [21], we settled on eleven criteria, described further below. Nine of these criteria are taken from the PoolParty checker (with the PoolParty Consistent Use of Labels check split into two distinct criteria). Of the remaining criteria, one appears only in the qSKOS list of criteria (Cycles in `broader` Hierarchy) while one criterion, Extra Whitespace, is our own addition which we have found to cause problems in vocabularies published using ONKI.

Valid URIs. The validity of resource URIs can be considered on many levels, from syntactical (the use of valid characters) to semantic (e.g. registered URI schemes) and functional (e.g. dereferenceability of HTTP URIs). The PoolParty checker only appears to perform a syntactic check. The qSKOS criterion VQC5 *Link Target Availability* is a stricter criterion, which requires that links are dereferenceable and the targets reachable on the Web.

Missing Language Tags. RDF literals used for, e.g., concept labels may or may not have a specified language tag. Missing language tags are problematic, especially for multilingual vocabularies. The PoolParty checker counts the number of concepts, which have associated literal values without a language tag. The qSKOS criterion VQC11a *Language tag support* addresses the same issue.

Missing Labels. `Concept`s and `ConceptScheme`s in the vocabulary should carry human-readable labels such as `prefLabel` or `rdfs:label`. The PoolParty checker verifies that this is the case. The qSKOS criterion VQC10 *Human readability* addresses the same issue, though the set of SKOS constructs and label properties to check is longer than in the PoolParty checker.

Loose Concepts. The PoolParty checker defines loose concepts as `Concept` instances that are not *top concepts* (i.e. having incoming `hasTopConcept` or outgoing `topConceptOf` relationships) in any `ConceptScheme` and have no `broader` relationships pointing to other concepts. The checker counts the number of such loose concepts. The qSKOS quality criterion VQC1 *Relative number of loose concepts* is similarly named, but is a stricter criterion: qSKOS defines loose concepts as those concepts that don't link to any other concepts using SKOS properties.

Disjoint OWL Classes. The SKOS specification defines that all the classes `Concept`, `Collection` and `ConceptScheme` are pairwise disjoint, that is, no resource may be an instance of more than one of these classes [14]. The PoolParty checker verifies that this is the case. qSKOS does not have an explicit criterion for this issue, but it is implicitly covered by VQC8 *SKOS semantic consistence*, which addresses all the SKOS consistency criteria.

Ambiguous `prefLabel` Values. The SKOS specification defines that "[a] resource has no more than one value of `prefLabel` per language tag" [14]. The PoolParty checker verifies this as a part of the Consistent Use of Labels check. This issue is also implicitly covered by VQC8 in qSKOS.

Overlap in Disjoint Label Properties. The SKOS specification defines that the label properties `prefLabel`, `altLabel` and `hiddenLabel` are pairwise disjoint, i.e. a concept may not have the same label in more than one of these properties [14]. This is also verified as a part of the Consistent Use of Labels check in the PoolParty checker. The qSKOS criterion VQC12 *Ambiguous labeling* addresses the same issue, but it is also implicitly covered by VQC8.

Consistent Use of Mapping Properties. The SKOS specification defines that the `exactMatch` relation is disjoint with both `broadMatch` and `relatedMatch`; that is, two `Concept`s cannot be mapped to each other using both `exactMatch` and one of the other mapping properties. The PoolParty checker verifies this in the Consistent Usage of Mapping Properties check. qSKOS does not have a specific criterion for this issue, but it is implicitly covered by VQC8.

Disjoint Semantic Relations. The SKOS specification defines the `related` relation to be disjoint with `broaderTransitive`; that is, two concepts cannot be connected by both [14]. The PoolParty checker verifies this in the Consistent Usage of Semantic Relations check. This issue is also implicitly covered by the qSKOS criterion VQC8.

Cycles in `broader` Hierarchy. Cycles in the hierarchies of terminological vocabularies can be simple mistakes that can arise when a complex vocabulary is created, but in some cases the cycles may carry important meaning [17]. Cycles are not forbidden by the SKOS specification and the PoolParty checker does not check for them. However, cycles can cause problems for automated processing such as term expansion in information retrieval systems, where any concept participating in a cycle may be considered equivalent to all the other concepts in the cycle. This issue is addressed by the qSKOS criterion VQC3 *Cyclic Relations*.

Extra Whitespace. SKOS vocabularies may contain surrounding whitespace in label property values such as `prefLabel`, `altLabel` and `hiddenLabel`. While extra whitespace is not forbidden by SKOS, it is unlikely to carry meaning and may cause problems when the vocabulary is, e.g., stored in a database or used for information retrieval, particularly when exact string matching is performed. Such extra whitespace is likely an artifact of conversion from another textual format such as XML or CSV, or it may originate in the text fields of graphical user interfaces used for vocabulary editing, where whitespace is typically invisible.

Table 2. Vocabularies used in validation tests, grouped by size (small, medium and large). When version is not indicated, the latest available SKOS file on 9th January 2012 was used. The columns Conc, Coll and CS show number of concepts, collections and concept schemes, respectively.

Name	Version	Publisher	Description	Conc	Coll	CS
STI Subjects	-	NASA	Subject classification of spacefaring terms	88	0	0
NYT Subjects	-	New York Times	Subject descriptors used in NY Times data	498	0	0
GBA Thesaurus	-	Geological Survey Austria	Thesaurus of geological terms	780	0	2
NYT Locations	-	New York Times	Geographical locations used in NY Times data	1920	0	0
IAU Thesaurus 1993 (IAUT93)	-	IVOA	Legacy astronomical thesaurus	2551	0	1
IVOA Thesaurus (IVOAT)	-	IVOA	Astronomical thesaurus	2890	0	1
GEMET	3.0	EIONET	Environmental thesaurus	5208	79	1
STW Thesaurus	8.08	ZBW	Economics thesaurus	6621	0	12
Schools Online Thesaurus (ScOT)	-	Education Services Australia	Terms used in Australian and New Zealand schools	8110	0	1
Medical Subject Headings (MeSH)	2006 [4]	US NLM	Biomedical vocabulary	23514	0	0
Finnish General Thesaurus (YSA)	2012-01-09	National Library of Finland	General thesaurus used in Finnish library catalogs	24206	61	1
SWD subject headings	07/2011	DNB	Subject headings used in German library catalogs	166414	0	0
LCSH	2011-08-11	Library of Congress	Subject headings used in Library of Congress catalog	407908	0	18
DBpedia Categories	3.7	DBpedia project	Categories from Wikipedia	740362	0	0

4 Validity of SKOS Vocabularies

To gain an understanding of the current quality of SKOS vocabularies published online, we first collected 14 freely-available vocabularies in SKOS format, published by 12 different organizations. Most of the vocabularies were discovered from the SKOS wiki[5]. Among the vocabularies that were available as file downloads, we selected vocabularies based on three criteria: 1) geographical diversity, 2) topical diversity and 3) diversity of vocabulary sizes. In two cases (NY Times and IVOA) we chose two vocabularies per publisher in order to compare vocabulary quality within the same publisher.

The vocabularies, together with some general statistics about the number of concepts, SKOS collections, concept schemes and RDF triples, are shown in Table 2. The vocabularies can be roughly categorized by size: *small* (fewer than 2000 concepts), *medium* (between 2000 and 10000 concepts) and *large* (more than 10000 concepts). We then analyzed most of these vocabularies using the PoolParty checker.

[5] `http://www.w3.org/2001/sw/wiki/SKOS/Datasets`

Table 3. Validation and Correction Results. The first group of columns shows the result of validating the vocabularies using the PoolParty checker. Of these, the last four columns represent mandatory checks that must be passed for the vocabulary to be considered valid by the PoolParty checker. The second group of columns shows the number of problems in each vocabulary that were found and corrected by Skosify.

	Valid URIs	Missing Language Tags	Missing Labels	Loose Concepts	Disjoint OWL Classes	Consistent Use of Labels	Consistent Use of Mapping Properties	Consistent Use of Semantic Relations	Missing Language Tags	Missing Labels	Loose Concepts	Disjoint OWL Classes	Ambiguous `prefLabel` values	Overlap in Disjoint Label Properties	Disjoint Semantic Relations	Cycles in `broader` Hierarchy	Extra Whitespace
STI Subj.	pass	88	pass	1	pass	pass	pass	pass	3134	0	1	0	0	0	0	0	88
NYT Subj.	pass	0	pass	498	pass	pass	pass	pass	0	1	498	0	0	0	0	0	2
GBA	pass	0	pass	0	pass	pass	pass	pass	0	0	1	0	0	0	0	0	30
NYT Loc.	pass	0	pass	1920	pass	**fail**	pass	pass	0	1	1920	0	0	0	0	0	0
IAUT93	pass	358	**fail**	1060	pass	**fail**	pass	**fail**	358	1	1060	0	0	1	10	0	40
IVOAT	pass	2890	pass	926	pass	pass	pass	**fail**	7330	1	926	0	0	0	11	6	0
GEMET	pass	3	**fail**	109	pass	pass	pass	**fail**	3	0	109	0	0	0	2	0	0
STW	pass	2	**fail**	0	pass	pass	pass	**fail**	2	0	0	0	0	0	7	0	2
ScOT	pass	0	pass	0	pass	**fail**	pass	**fail**	0	0	0	0	0	1	26	0	1
MeSH	pass	0	pass	189	pass	pass	pass	**fail**	0	0	189	0	0	0	383	12	22610
YSA	pass	0	**fail**	8614	**fail**	pass	pass	**fail**	0	0	8614	61	0	0	58	6	0
SWD									0	0	65363	0	2	127	108	2	42
LCSH									0	0	423010	0	0	18	200	0	0
DBpedia									0	0	90822	0	0	0	10100	6168	0

4.1 Validation Results

For the first eleven vocabularies, we used the PoolParty checker to look for problems. The results of these checks are summarized in Table 3. Some vocabularies had to be pre-processed before validation[6]. The three largest vocabularies – DNB SWD, LCSH and DBpedia Categories – were too large for the checker.

Only the GBA Thesaurus passed all checks without problems. All the medium-size and large vocabularies that we tested failed at least one mandatory check, meaning that they did not meet some of the SKOS integrity constraints.

5 Correcting Problems

To further analyze structural problems in SKOS vocabularies and to correct as many of them as possible, we created the **Skosify** tool. It is a command line

[6] The GBA Thesaurus used an invalid character encoding, which we corrected. The IVOA Thesaurus used a deprecated SKOS namespace which we corrected. GEMET and MeSH consisted of several RDF files, which we merged into a single file. We converted MeSH into Turtle syntax and removed some vocabulary-specific attributes to keep it below the 20MB limit of the PoolParty checker.

utility that reads one or more SKOS files as input, performs validation checks and structural corrections on the vocabulary, and outputs a corrected SKOS file. Skosify was implemented in Python using the rdflib[7] toolkit. It has been released[8] as open source under the MIT License. An online version of the tool is also available[9].

We processed all the vocabularies listed in Table 2 with Skosify[10]. The number of corrections performed for each vocabulary is shown in the last group of columns of Table 3.

5.1 Addressing the Validity Criteria

Of the eleven validation criteria defined in Section 3, Skosify as of version 0.5 addresses nine, as shown in the last column of Table 1. The criteria for Valid URIs and Consistent Use of Mapping Properties are not addressed in the current version; however, none of the vocabularies we tested violated these according to the PoolParty checker. Corrections performed by Skosify for the remaining criteria are described in the following subsections.

Missing Language Tags. The STI Subjects and both IVOA astronomical thesauri have a large number[11] of concepts with label property values lacking language tags. GEMET and the STW Thesaurus also lack a few language tags. Skosify was able to correct these when given a *default language* setting. However, this approach only works when the language of untagged literals is known and different languages have not been mixed.

Missing Labels. Four vocabularies have either concept schemes or concepts without labels. Skosify can detect unlabeled concept schemes and optionally, when given a label as parameter, add the missing label. However, Skosify does not detect unlabeled concepts.

Loose Concepts. Most of the vocabularies we examined contain loose concepts. The STI Subjects, both NY Times vocabularies, MeSH, SWD and DBpedia Categories do not include any `ConceptScheme` instances, so the existence of loose concepts was a natural consequence. GEMET, YSA and LCSH[12] do include

[7] `http://rdflib.net`

[8] `http://code.google.com/p/skosify/`

[9] `http://demo.seco.tkk.fi/skosify`

[10] The SVN repository of Skosify includes a test suite for processing all the above mentioned vocabularies as well as PoolParty checker reports before and after Skosify processing. The processed vocabularies have been provided as a separate download.

[11] The reported number of missing language tags in Table 3 is sometimes different for the different tools, because the PoolParty checker groups the missing language tags by concept, while Skosify counts every label without language tag separately.

[12] In LCSH, every concept exists in two concept schemes. The number of loose concepts for LCSH in Table 3 is higher than the total number of concepts in LCSH because loose concepts are determined per concept scheme, so the same concept may be counted twice. In total, LCSH has 211505 different loose concepts.

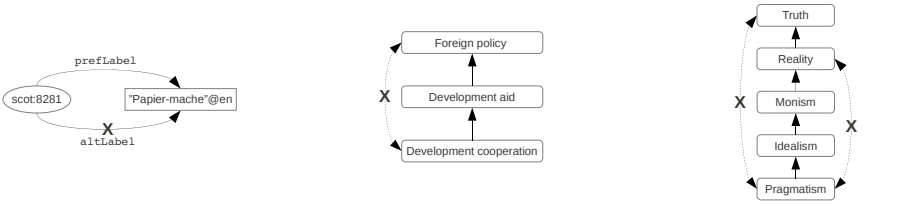
one or more `ConceptScheme` instances, but do not use any `hasTopConcept` or `topConceptOf` relationships to define top concepts, again leading to loose concepts. Both the IVOA vocabularies use `topConceptOf` relationships, but do not mark all of the top level concepts using those properties. In these cases, Skosify identifies the top level concepts (those with no `broader` relationships) and adds `hasTopConcept` and `topConceptOf` relationships to a concept scheme, creating one if necessary.

Disjoint OWL Classes. YSA was the only vocabulary that failed the Disjoint OWL Classes test in the PoolParty checker. In this case, the problem was that some relationships intended for `Concept`s, such as `exactMatch`, were used on `Collection` instances. The RDFS inference capabilities of the PoolParty checker together with `rdfs:domain` specifications of some SKOS properties caused those instances to be marked both as `Concept`s and `Collection`s. Skosify identifies this particular error and corrects it by removing the improper relationship assertions. However, it cannot correct the more general case where a resource is explicitly marked as being of several types that are defined to be disjoint.

Ambiguous `prefLabel` Values. Of the vocabularies we tested, only the SWD subject headings was found by Skosify to contain concepts with more than one `prefLabel` using the same language tag. On a closer look, the two cases appeared to be genuine errors: in one case, the term *Einheitshaus* is used in the same concept with *TURBO C++ für WINDOWS*, and in the other case, *Hämatogene Oxidationstherapie* appears together with *Pikkoloflötenspiel*. In these situations, Skosify arbitrarily selects one of the labels as the real `prefLabel` while the rest are converted into `altLabel`s. By default, Skosify will choose the shortest `prefLabel`, but other options are available for choosing the longest label or not performing any correction at all.

Overlap in Disjoint Label Properties. Four of the vocabularies we tested contain cases where a concept is linked to a label using two different label properties that are defined as disjoint by the SKOS specification. An example from ScOT is shown in Figure 1a. In this situation, Skosify removes the value for the less important property (`hiddenLabel` < `altLabel` < `prefLabel`).

Disjoint Semantic Relations. Ten of the vocabularies we tested (all but the four smallest) contain cases where a concept is linked to another concept using relationships that are defined as disjoint by the SKOS specification. In particular, the `related` relationship is often used to link between concepts that are directly above or below each other in the `broader` hierarchy, as shown in Figure 1b and 1c. In this situation, Skosify removes the `related` relationship assertion, leaving the `broader` hierarchy intact. This correction is performed by default, in order to enforce the SKOS integrity condition S27, but it can be disabled.



a) overlapping labels in ScOT

b) related concepts within the same hierarchy in the STW Thesaurus

c) related concepts within the same hierarchy in LCSH

Fig. 1. Examples of overlapping labels (a) and disjoint semantic relations (b and c). In subfigures b and c, line arrows represent `broader` relationships while dotted arrows represent `related` relationships between concepts. Crosses (X) mark relationships that were eliminated by Skosify.

Cycles in `broader` Hierarchy. In the vocabularies we examined, we found many examples of cycles in the `broader` hierarchy. Some examples of these are shown in Figure 2.

The simplest kind of cycle is a concept which has a `broader` relationship to itself, as in Figure 2a. Another simple case is when a concept has a `broader` relationship to its child, as shown in Figure 2b. In both these cases the relationship causing the cycle is probably an error and can be rather safely eliminated.

More complex cases are cycles where the concepts participating in the cycle are both on the same hierarchy level (i.e. have the same minimum path length to top-level concepts), as in Figures 2c and 2d. In these cases it is difficult to automatically select the relationship to be eliminated. It is still likely that the cycle is a mistake, but properly correcting it may require human intervention.

MeSH 2006 contains several cycles in the SKOS version. They arise because MeSH is structured along several hierarchies (facets) which each have their own criteria for hierarchical relationships. The SKOS version aggregates these hierarchies so that the distinctions between facets are lost. One interesting cycle is the one involving the concepts Morals and Ethics, shown in Figure 2e. This cycle appears to be intentional[13] and it still exists in MeSH 2012.

DBpedia Categories contain thousands of cycles. The DBpedia authors note that the "categories do not form a proper topical hierarchy, as there are cycles in the category system and as categories often only represent a rather loose relatedness between articles" [6]. The cycles arise because the semantics of Wikipedia categories do not match traditional terminological hierarchies.

Skosify can detect and optionally remove cycles in the `broader` hierarchy. It uses a naïve approach based on performing a depth-first search starting from the topmost concepts in the hierarchy. In Figure 2, the relationships removed by Skosify during a test run with cycle elimination enabled have been crossed out.

The depth-first search approach for eliminating cycles is simple, fast and domain independent, but may not produce deterministic results and "cannot ensure that the links ignored during the graph traversal in order to prevent loops from

[13] For some discussion on the issue, see:
`http://lists.w3.org/Archives/Public/public-esw-thes/2011Jul/0005.html`

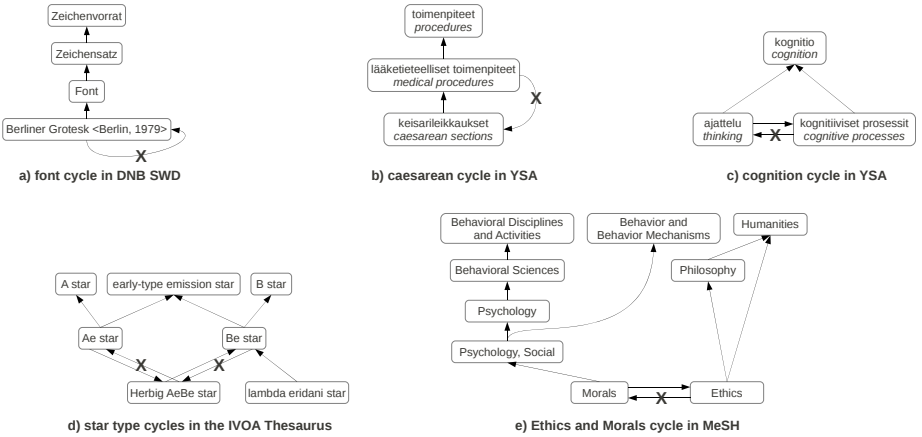


Fig. 2. Examples of cycles from the test vocabularies. English equivalents for Finnish terms are shown in *italic*. Arrows represent `broader` relationships between concepts. Crosses (X) mark relationships that were eliminated by Skosify during a particular run.

happening are actually the appropriate links to be removed" [15]. More accurate formal methods for eliminating cycles in terminological hierarchies exist, but they are more complex and not as general as the naïve approach [15].

Despite the limitations of the naïve cycle elimination approach, it can be useful for alerting vocabulary maintainers of possible problems. For example, the maintainers of YSA were alerted of the existence of cycles detected by Skosify, including those shown in Figures 2b and 2c, and these have since been eliminated both from the original vocabulary and its SKOS version.

Extra Whitespace. Eight of the vocabularies we tested contain SKOS label or documentation property values with surrounding whitespace. MeSH is particularly prone to this, with over 22000 such cases. Skosify removes the extra whitespace from these values.

5.2 Other Transformations

SKOS includes some redundant ways of representing information. Hierarchical relationships can be expressed using either `broader` or `narrower` (or both). Likewise, top-level concepts in a concept scheme can be described using either `hasTopConcept` or `topConceptOf` [14]. Additionally, transitive properties such as `broaderTransitive` and general properties such as `semanticRelation` can in principle be inferred using RDFS/OWL inference. Whether such redundant information is desirable depends on the needs of the application: for example, the Helping Interdisciplinary Vocabulary Engineering tool[14] requires hierarchical relationships to be specified using both `broader` and `narrower`. The qSKOS

[14] `http://code.google.com/p/hive-mrc/`

quality criteria include a *Redundant Relationships* criterion which seeks to measure the amount of redundancy in a SKOS vocabulary.

Skosify has support for generating either a minimally redudant version of the hierarchy of a SKOS vocabulary (using only `broader` relationships), or a version with both `broader` and `narrower` relationships. It can also be used to explicitly assert transitive relationships.

Skosify can also be given arbitrary RDF files as input, together with a mapping configuration specifying how the RDF constructs used in the input correspond to SKOS constructs. We have used this capability to transform many lightweight OWL ontologies created in the FinnONTO projects [21] into structurally valid SKOS versions, which are published alongside the original OWL versions. This capability is described in more detail in the Skosify wiki[15].

We have also found that some FinnONTO vocabularies contain remnants of previously used RDF lists, consisting mostly of blank nodes. Skosify removes such RDF graph elements that are disconnected from the main vocabulary graph.

6 Evaluation

For evaluating how well we attained the goal of improving SKOS vocabularies, we considered 1) improvements in vocabulary validity resulting from its use; 2) the performance and scalability of Skosify when used with large vocabularies.

6.1 Improvements in Validity

We revalidated the SKOS vocabularies processed by Skosify using the PoolParty checker using the same methodology as described in Section 4, again excluding the three largest vocabularies. In most cases, the validation problems shown in Table 3 had indeed been corrected. However, some problems remained.

GEMET, STW Thesaurus and YSA still failed the Missing Labels check. GEMET contains concepts withous labels, and since Skosify did not attempt to correct this issue, the outcome was expected. For the STW Thesaurus and YSA, the problem was caused by a concept scheme being labeled with a `prefLabel`. This property is not recognized by the PoolParty Checker, which only looks for `rdfs:label` properties of concept schemes.

The NY Times Locations vocabulary still did not pass the Consistent Use of Labels check. The vocabulary contains different descriptions of geographical locations, interlinked using `owl:sameAs` relationships. Skosify does not perform OWL inference, so it did not identify cases where the same location was named using different `prefLabels` for different resource URIs. The PoolParty checker performs `owl:sameAs` inference so it was able to detect these inconsistent labels.

Skosify found one loose concept in the GBA Thesaurus, despite it having passed the PoolParty check for loose concepts. This discrepancy is caused by the GBA Thesaurus not using any explicit `inScheme` relationships despite containing

[15] `http://code.google.com/p/skosify/wiki/GettingStarted`

two `ConceptScheme` instances. Skosify is unable to infer which concept scheme(s) concepts belong to and thus it misidentifies one concept (*Lithstrat*) as being loose, even though it is marked as a top concept of one of the concept schemes.

6.2 Performance

On a modern desktop PC (Intel Core i5 CPU, 8GB RAM), processing of the largest vocabularies, LCSH and DBpedia Categories, took 25 minutes and 90 minutes, respectively. Memory usage was below 4GB in each case. Many vocabularies published on the ONKI ontology service [20] are automatically processed with Skosify as a part of the publication process.

7 Discussion

In this study, we first looked at the possible criteria for determining the validity and quality of SKOS vocabularies. We created a list of eleven criteria that we collected and synthetized from several sources.

We then surveyed freely available SKOS vocabularies for different domains and measured their validity, mainly using the PoolParty checker. We found that most vocabularies, particularly the larger ones, violate one or more of the SKOS integrity conditions. This is not surprising, since RDF data published online has been found to contain many errors [9,10,11]. However, earlier studies did not look specifically at the validity of SKOS vocabularies. In particular, we found that the SKOS integrity condition S27, which specifies that the `related` relationship is disjoint with the `broaderTransitive` relationship, is violated by nearly all of the vocabularies we examined. The YSA maintainers specifically declined to remove some relationships violating this constraint from the vocabulary, because they considered them important for vocabulary users. A similar argument could be made for LCSH, which has a very complex `broader` hierarchy with a large amount of multiple inheritance. Thus, the constraint in its current form could be considered overly strict. It could be amended by only forbidding `related` relationships between direct descendants rather than considering the whole transitive hierarchy.

Finally, we created the Skosify tool and used it to improve the validity of fourteen SKOS vocabularies. Our tool was able to correct the great majority of identified structural problems in these vocabularies. If the same processing and validation were performed on a larger selection of SKOS vocabularies, we expect new kinds of problems to be found. Still, the corrections performed by Skosify appear to be useful for many different vocabularies. The implementation is fast enough to be used routinely as a part of the publication process for SKOS vocabularies. Skosify can also be used as a validation tool, particularily for large vocabularies which can be difficult to process using online tools.

In future work, we intend to examine a wider selection of vocabularies and to evaluate the correction results against other validation tools such as the qSKOS tool. The online version of Skosify could be further developed to expose more of the functionalities of the command line version and to better support validation.

Acknowledgments. This work is part of the National Semantic Web Ontology project in Finland FinnONTO[16] (2003–2012), funded mainly by the National Technology and Innovation Agency (Tekes) and a consortium of 38 public organizations and companies. We thank Christian Mader for assistance in relating the qSKOS criteria to other sources, Jouni Tuominen, Matias Frosterus and Eeva Kärki for correcting structural problems in YSA, and Andreas Blumauer and Alexander Kreiser for technical assistance with the PoolParty checker as well as for providing this excellent and free online validation service in the first place.

References

1. ISO 2788: Guidelines for the establishment and development of monolingual thesauri. International Organization for Standardization (ISO) (1986)
2. Structured vocabularies for information retrieval. Part 2: Thesauri (BS 8723-2:2005). British Standards Institution (2005)
3. Aitchison, J., Gilchrist, A., Bawden, D.: Thesaurus Construction and Use: A Practical Manual. Aslib IMI (2000)
4. van Assem, M., Malaisé, V., Miles, A., Schreiber, G.: A Method to Convert Thesauri to SKOS. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 95–109. Springer, Heidelberg (2006)
5. Berrueta, D., Fernández, S., Frade, I.: Cooking HTTP content negotiation with Vapour. In: Proceedings of 4th Workshop on Scripting for the Semantic Web, SFSW 2008 (2008)
6. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 154–165 (2009)
7. Borst, T., Fingerle, B., Neubert, J., Seiler, A.: How do libraries find their way onto the Semantic Web? Liber Quarterly 19(3/4) (2010)
8. Byrne, G., Goddard, L.: The strongest link: Libraries and linked data. D-Lib Magazine 16(11/12) (2010)
9. Ding, L., Finin, T.W.: Characterizing the Semantic Web on the Web. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 242–257. Springer, Heidelberg (2006)
10. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: Proceedings of the 3rd International Workshop on Linked Data on the Web, LDOW 2010 (2010)
11. 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)
12. Mader, C., Haslhofer, B.: Quality criteria for controlled web vocabularies. In: Proceedings of the 10th Europen Networked Knowledge Organisation Systems Workshop, NKOS 2011 (2011)
13. Malmsten, M.: Making a library catalogue part of the semantic web. In: Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications, pp. 146–152. Dublin Core Metadata Initiative (2008)
14. Miles, A., Bechhofer, S.: SKOS simple knowledge organization system reference. World Wide Web Consortium Recommendation (August 2009), `http://www.w3.org/TR/skos-reference/`

[16] `http://www.seco.tkk.fi/projects/finnonto/`

15. Mougin, F., Bodenreider, O.: Approaches to eliminating cycles in the UMLS Metathesaurus: Naïve vs. formal. In: American Medical Informatics Association (AMIA) Annual Symposium Proceedings, pp. 550–554 (2005)
16. Nagy, H., Pellegrini, T., Mader, C.: Exploring structural differences in thesauri for SKOS-based applications. In: I-SEMANTICS 2011, Graz, Austria (September 2011)
17. Nebel, B.: Terminological cycles: Semantics and computational properties. In: Principles of Semantic Networks. Morgan Kaufmann, San Mateo (1991)
18. Schandl, T., Blumauer, A.: PoolParty: SKOS Thesaurus Management Utilizing Linked Data. 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. 421–425. Springer, Heidelberg (2010)
19. Summers, E., Isaac, A., Redding, C., Krech, D.: LCSH, SKOS and Linked Data. In: Proceedings of the International Conference on Dublin Core and Metadata Applications (DC 2008), pp. 25–33. Dublin Core Metadata Initiative (2008)
20. Tuominen, J., Frosterus, M., Viljanen, K., Hyvönen, E.: ONKI SKOS Server for Publishing and Utilizing SKOS Vocabularies and Ontologies as Services. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 768–780. Springer, Heidelberg (2009)
21. Viljanen, K., Tuominen, J., Hyvönen, E.: Ontology Libraries for Production Use: The Finnish Ontology Library Service ONKI. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 781–795. Springer, Heidelberg (2009)

The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation

Silvio Peroni[1], David Shotton[2], and Fabio Vitali[1]

[1] Department of Computer Science, University of Bologna
[2] Department of Zoology, University of Oxford
{essepuntato,fabio}@cs.unibo.it, david.shotton@zoo.ox.ac.uk

Abstract. In this paper we introduce *LODE*, the *Live OWL Documentation Environment*, an online service that automatically generates a human-readable description of any OWL ontology (or, more generally, an RDF vocabulary), taking into account both ontological axioms and annotations, and ordering these with the appearance and functionality of a W3C Recommendations document. This documentation is presented to the user as an HTML page with embedded links for ease of browsing and navigation. We have tested LODE's completeness and usability by recording the success of test users in completing tasks of ontology comprehension, and here present the results of that study.

Keywords: LODE, OWL ontologies, Web tools, ontology documentation, user testing.

1 Introduction

Any strategy that guarantees the broad adoption of Semantic Web technologies must address the need for improved human interaction with semantic models and data. While research has been undertaken on models, theoretical approaches and the development of tools to infer new information from data and ontologies, the Semantic Web will never be really integrated with the everyday Web until semantic information is easily accessible to ordinary users through Web browsers, not limited to Semantic Web practitioners employing specialist tools. This point is even more crucial for Semantic Publishing, since its end-users are by definition publishers, researchers, librarians and general readers, rather than experts in semantic technologies. Thus the Semantic Web / Semantic Publishing communities need to develop user-friendly Web interfaces that mediate between semantic models and end-users.

Of course, work has already been done in this direction. For instance, ontology development editors have been created (e.g. Protégé[1] [12] and the NeOn Toolkit [19]), Web search engines to look for semantic resources launched (e.g.

[1] Protégé: http://protege.stanford.edu

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 398–412, 2012.

Sindice[2] [15] and Watson[3] [5]), and semantic desktop applications released (e.g. SemNotes[4] [7]). However, what is still missing, and what the Semantic Publishing community urgently needs, are tools that assist people who are not expert in semantic technologies in dealing with and publishing semantic data, and in particular in understanding the ontologies that make this possible.

Human interactions with ontologies usually involves the following steps:

1. Once ontologies suitable for the particular domain of interest have been identified, people need to *understand* these models with the minimum amount of effort.
2. Then, if the existing vocabularies/ontologies are not able to fully describe the domain in consideration, people *develop* new models. The development process requires interaction with domain experts and end-users in order to produce a model that address the domain under consideration as fully as possible.
3. Finally, people have to *add* data according to the adopted or developed models and to *modify* those data in the future.

Each of these four operations – understanding, developing, adding and modifying – need to be supported by appropriate interfaces that simplify the work of people who may not be expert in ontology-related formalisms and Semantic Web technologies. In this paper, our focus is on the first point of the above list: *ontology understanding*.

Usually, the first activity performed when someone wants to understand the extent of a particular ontology is to consult its human-readable documentation. A large number of ontologies, especially those used in the Linked Data world, have good comprehensive Web pages describing their theoretical backgrounds and the features of their developed entities. However, problems arise when we look at under-developed models, since natural language documentation is usually only published once an ontology has become stable. This approach is justifiable: writing proper documentation costs effort, and re-writing it every time the developing ontology is modified is not practical.

Thus, the only previous way of getting a sense of existing ontologies was to open them in an ontology editor so as to explore their logical axioms. This approach presents practical and cognitive barriers to a person approaching the ontology world for the very first time. First, (s)he has to download and install an ontology editor. Second, (s)he must learn to use the editor, which typically will have a complex and non-intuitive user interface, presenting the user with novel technical terms such as 'axiom' and 'refactor', or common English words used in novel ways, e.g. 'individual' or 'functional'. Then and only then (s)he can try to get a sense of structure of the ontology itself. Obviously these processes are challenging and time-consuming, presenting a barrier that is too great for the majority of non-specialists.

[2] Sindice: `http://sindice.com`
[3] Watson: `http://watson.kmi.open.ac.uk`
[4] SemNotes: `http://smile.deri.ie/projects/semn`

To address this issue, and following the lines of previous works such as *Parrot* [20] and *Neologism* [1], we have developed the *Live OWL Documentation Environment*[5] (*LODE*), an online service that takes any well-formed OWL ontology or, more generally, an RDF vocabulary, and generates a human-readable HTML page designed for browsing and navigation by means of embedded links. In this paper, we introduce this tool, describe its features, and evaluate its usability by considering the success of test users in completing tasks of ontology comprehension.

The rest of the paper is structured as follows. In Section 2 we introduce relevant works in tools for the automatic production of ontology documentation. In Section 3 we present LODE, highlighting its characteristics and features in detail. In Section 4 we present the setting and the details of the test-user experiments we ran to assess its completeness and usability, and in Section 5 we discuss the outcomes of these experiments. Finally, in Section 6, we conclude the paper by sketching out the future developments of our work.

2 Related Works

The production of natural language documentation for ontologies is an important and crucial part of any ontology development process. Such documentation enables users to comprehend the extent of an ontology without having to concern themselves with the particular formal language used to define its axioms. At the same time, writing such documentation manually is an activity that involves a significant effort. Thus, in order to help authors of ontologies to document them, applications have been developed for the automated creation of a first draft of such documentation starting from labels (i.e. *rdfs:label*), comments (i.e. *rdfs:comment*), and other kinds of annotations (e.g. *dc:description*, *dc:creator*, *dc:date*) within the ontology, and from the logical structure of the ontology itself.

SpecGen[6] is a Python tool for the generation of ontology specifications, available as a stand-alone application. It has been used to prepare the HTML documentation of well-known ontologies such as SIOC[7]. SpecGen generates the documentation by processing a pre-defined HTML template into which it adds the list of ontological classes and properties in specific positions. As a result, we obtain a new HTML document where the natural language description of the ontology comes entirely from the template made by authors, while the software takes care of adding the specific information related to the logical structure of the ontology.

In contrast to SpecGen, that needs its base HTML template to work, *VocDoc*[8] is a small Ruby script that allows one to produce documentation starting from

[5] LODE, the Live OWL Documentation Environment: http://www.essepuntato.it/lode

[6] SpecGen: http://forge.morfeo-project.org/wiki_en/index.php/SpecGen

[7] The Semantically-Interlinked Online Communities (SIOC) project: http://sioc-project.org

[8] VocDoc: http://kantenwerk.org/vocdoc/

RDFS vocabularies and OWL ontologies. It is able to produce both HTML documents and LaTeX files containing the description of the ontology/vocabulary.

Like VocDoc, *OWLDoc*[9] is a fully-automatic generator of a set of HTML pages describing the target ontology. It organises the documentation of each ontological entity in different sections, showing the taxonomy of each entity, the usage of this entity in the context of the ontology, and the formal logical axioms related to the entity (expressed in Manchester Syntax [11]). OWLDoc has been developed as plugin of *Protégé*[10] [12], and as a separate Web application[11].

Oriented to Linked Data applications rather than to ontology documentation, *Paget*[12] is a PHP framework that, upon receipt of an input URL through a browser, dispatches the request according to the particular mime-type specified by the client, and retrieves RDF entities in four different formats: RDF, HTML, Turtle and JSON. It can be used to describe a set of pure RDF statements (*subject-predicate-object*)[13] and, to some extent, to produce a human-comprehensible HTML description from the axioms of an OWL ontology[14].

Neologism[15] [1] is a Web-based editor for the creation of RDFS vocabularies and (very simple) OWL ontologies. Moreover, it implements a publishing system that allows the publication of vocabularies and ontologies on the Web, rendered into natural language HTML pages. Basca *et al.*'s main goal in creating it was to reduce the time needed to create, publish and modify vocabularies for the Semantic Web.

Finally, Parrot[16] [20] is a Web service for the generation of HTML+Javascript documentation of OWL ontologies and RIF rules [3]. This service allows one to specify multiple URLs identifying ontologies in order to produce an HTML summary of them "on the fly", starting from their logical structure and annotations.

3 LODE

LODE, the *Live OWL Documentation Environment*, is a novel online service that automatically generates a human-readable description of an OWL ontology (or, more generally, an RDF vocabulary), taking into account both ontological axioms and annotations, and that orders these with the appearance and functionality of a W3C Recommendation document by use of Cascading Style Sheets (CSS)[17].

[9] OWLDoc: http://code.google.com/p/co-ode-owl-plugins/wiki/OWLDoc
[10] Protégé: http://protege.stanford.edu/
[11] Ontology browser: http://code.google.com/p/ontology-browser/
[12] Paget: http://code.google.com/p/paget
[13] Ian Davis' Linked Data profile, rendered through Paget: http://iandavis.com/id/me.html
[14] A vocabulary for describing whisky varieties, rendered through Paget: http://vocab.org/whisky/terms.html
[15] Neologism: http://neologism.deri.ie
[16] Parrot: http://ontorule-project.eu/parrot/parrot
[17] W3C CSS: http://www.w3.org/TR/CSS/

It automatically extracts classes, object properties, data properties, named individuals, annotation properties, meta-modelling (punning), general axioms, SWRL rules and namespace declarations from any OWL or OWL 2 ontology, and renders them as ordered lists, together with their textual definitions, in a human-readable HTML page designed for easy browsing and navigation by means of embedded links.

LODE is based on an XSLT stylesheet that takes the RDF/XML linearisation of an ontology produced through the OWLAPI[18] [10] as input, and converts it into an HTML representation. If the target ontology is already linearised in that form, it is possible to call the LODE service directly by specifying its URL (i.e. http://www.essepuntato.it/lode/) followed by the complete URL of the ontology, as shown in the following example:

> http://www.essepuntato.it/lode/http://www.essepuntato.it/2008/12/earmark

In the following subsections we introduce the most important features of LODE.

3.1 What Axioms Are Used to Create the Documentation

Primarily, LODE interprets the most common annotation properties used for the description of entities, in particular[19]: *dc:contributor*, *dc:creator*, *dc:date*, *dc:description*, *dc:rights*, *dc:title*, *dcterms:contributor*, *dcterms:creator*, *dcterms:date*, *dcterms:description*, *dcterms:rights*, *dcterms:title*, *owl:versionInfo*, *rdfs:comment*, *rdfs:isDefinedBy*, *rdfs:label*. LODE adopts the following rules when transforming those annotations in HTML documentation:

- in the presence of Dublin Core annotations defined according to both DC Metadata Elements [9] and DC Metadata Terms [8], the former have precedence;
- dates (i.e. *dc:date* and *dcterms:date*) written according to the XML Schema datatype (i.e. *yyyy-mm-dd*) are automatically transformed into *dd/mm/yyyy*;
- agents (i.e. *dc:creator*, *dc:contributor*, *dcterms:creator* and *dcterms:contributor*) are rendered either as strings or as clickable URLs according to their types, i.e. literals or resources, respectively;
- descriptions (i.e. *dc:description* and *dcterms:description*) are rendered either as strings or as media objects according to their types, i.e. literals or resources, respectively;
- comments (i.e. *rdfs:comment*) and descriptions (i.e. *dc:description* and *dcterms:description*) are represented, respectively, as abstracts and as detailed descriptions of entities;

[18] OWLAPI: `http://owlapi.sourceforge.net`

[19] The prefixes *dc*, *dcterms*, *owl* and *rdfs* in the following list respectively refers to "`http://purl.org/dc/elements/1.1/`", "`http://purl.org/dc/terms/`", "`http://www.w3.org/2002/07/owl#`" and "`http://www.w3.org/2000/01/rdf-schema#`".

– labels (i.e. *rdfs:label*) and QNames (when labels are not specified) are used to refer to all the entities of the ontology, instead of using their URLs;
– the nature of each entity is identified by a descriptive abbreviation, according to its type: “c”, “op”, “dp”, “ap” and “ni” being used to identify class, object property, data property, annotation property and named individual, respectively.

In Fig. 1 and Fig. 2 we show how these annotations are rendered for an example ontology, EARMARK [6], an ontology that defines entities describing document markup (such as elements, attributes, comments and text nodes).

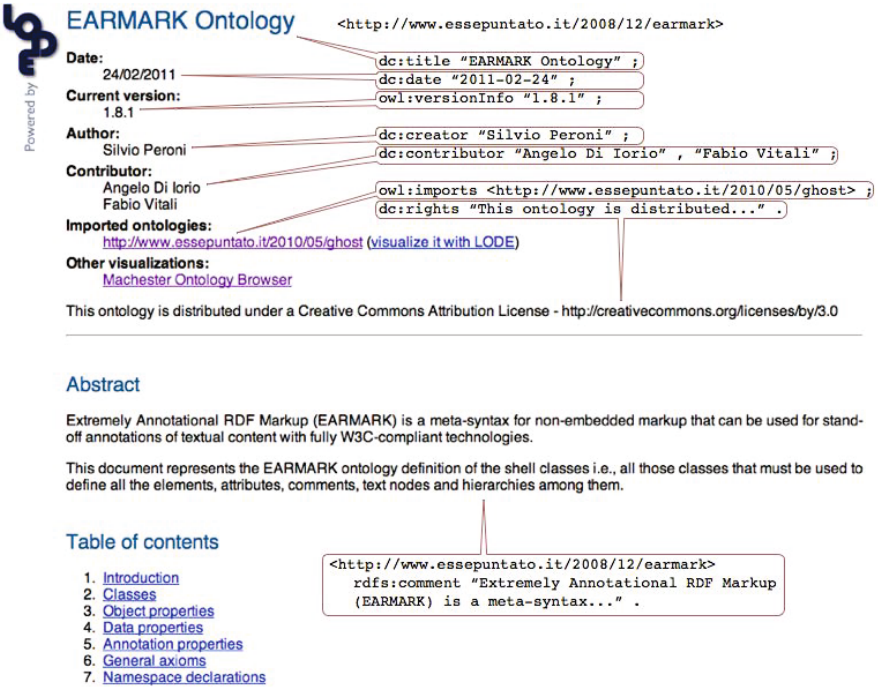


Fig. 1. The beginning of the Web page generated by LODE for the EARMARK Ontology, annotated with OWL assertions in Turtle (not present in the normal LODE web page) illustrating how these assertions are rendered in HTML

LODE converts all the other axioms of the ontology into Manchester Syntax definitions [11], as shown in Fig. 3. We prefer to use this syntax rather than others since it is the most human-comprehensible syntax for ontological axioms, and thus the most helpful for non-specialists.

Ontological axioms are rendered in grey boxes, one for each entity declared in the ontology. The axioms taken into account by LODE refer to: super-class and super-property, equivalent class and property, disjoint class and property,

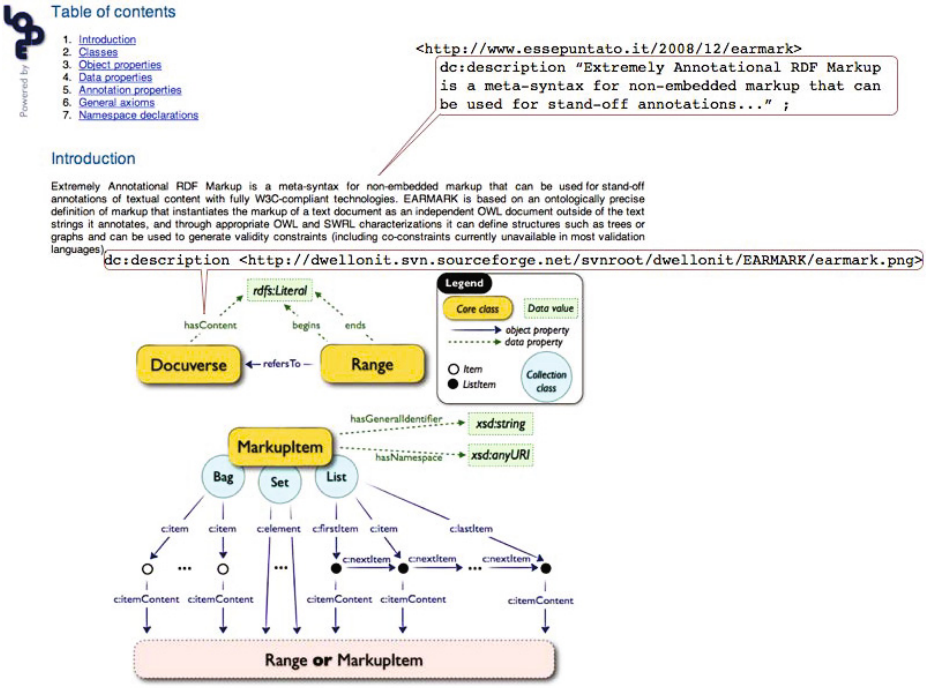


Fig. 2. Two possible kinds of descriptions: pure string (for literals) and media object (for resources)

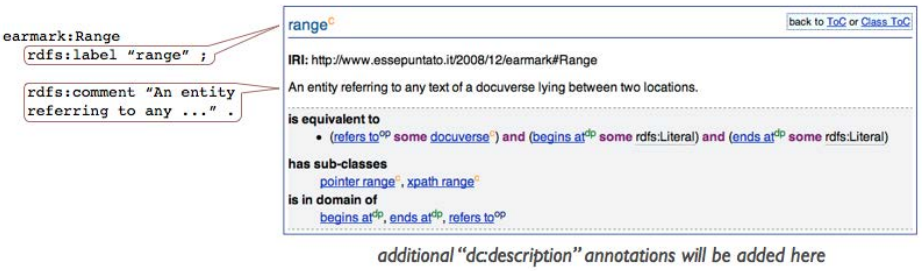


Fig. 3. How entities (classes, properties and individuals) are rendered by LODE

property domain and range, property chain, keys, object/data property assertion, type, imported ontology, generic axiom and SWRL rule. Moreover, LODE automatically enriches those definitions, adding information about sub-classes, domain/range properties of classes, sub-properties and entity meta-modelling.

3.2 Special Parameter Usage When Calling the LODE Service

LODE can be invoked with a number of optional parameters so as to limit or extend the final documentation produced. For instance, it is possible to take into account all the entities in the ontology closure and/or the inferred axioms. The following pseudo-URL describes how to call LODE:

http://www.essepuntato.it/lode/optional-parameters/ontology-url

In particular:

- **www.essepuntato.it/lode** is the URL to call the service;
- **ontology-url** is the full "http://..." URL of the OWL ontology that will be processed by the service. It must be always the last item of the pseudo-URL, and may be preceded by one or more slash-separated parameters.

Fig. 4 illustrates the alternative ways to build the URL to call LODE and the related modules used. The optional slash-separated parameters are described as follows.

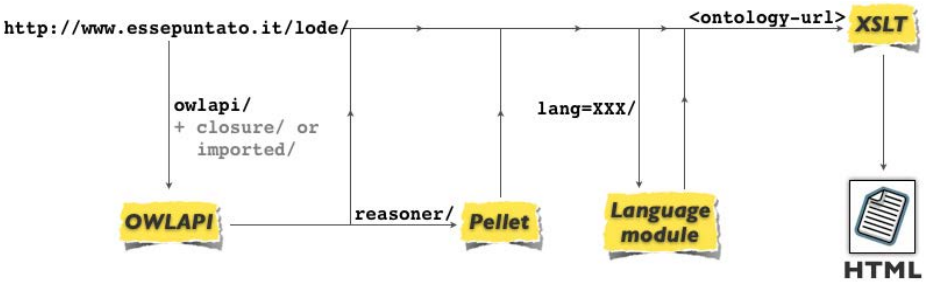


Fig. 4. All the possible ways, according to specific needs, for making a request to LODE

Parameter "owlapi". When this optional parameter is specified, the ontology defined in *ontology-url* will be pre-processed via the OWLAPI [10], in order to linearise it into the RDF/XML format accepted by LODE. This parameter **is always strongly recommended** to process correctly all those ontologies that were not developed through the OWLAPI.

Parameter "imported". When this optional parameter is specified, the axioms in the imported ontologies of *ontology-url* are added to the HTML description of the ontology itself. This parameter implicitly specifies the *owlapi* parameter.

Parameter "closure". When this optional parameter is specified, the transitive closure given by considering the imported ontologies of *ontology-url* is added to the HTML description of the ontology. This parameter implicitly specifies the *owlapi* parameter. If both the parameters *closure* and *imported* are specified (in any order), *imported* will be preferred.

Parameter "reasoner". When this optional parameter is specified, the inferred axioms of *ontology-url* (through the Pellet reasoner [17]) will be added to the HTML description of the ontology. This parameter implicitly specifies the *owlapi* parameter. Note that, depending on the nature of the ontology to process, this computationally intensive function can be very time-consuming.

Parameter "lang". When this optional parameter is specified, the selected language will be used as the preferred language instead of English when showing the documentation of *ontology-url*. It must be followed by an "=" and the abbreviation of the language to use. E.g. "lang=it" for Italian, "lang=fr" for French, etc. (This presupposes that appropriate language descriptions are present in the ontology.)

3.3 URI Fragments

LODE offers intuitive mechanisms to refer to particular ontological entities within the HTML documentation, according to the URL of the entity in consideration. The following extension of the pseudo-URL introduced in Section 3.2 defines how to refer to a particular entity of an ontology:

> http://www.essepuntato.it/lode/optional-parameters/ontology-url **#entity**

For instance, to generate the documentation of EARMARK and then jumping directly to the point where the resource "http://www.essepuntato.it/2008/12/earmark#Element" is described, we need to invoke LODE as follows:

> http://www.essepuntato.it/lode/http://www.essepuntato.it/2008/12/earmark#http://www.essepuntato.it/2008/12/earmark#Element

This request can be simplified if we look for a description of an entity defined as a *fragment* of the ontology URL, such as the entity *Element* in EARMARK. In this particular case, we can use either the entire entity URL, as illustrated previously, or the entity local name only, as shown as follows:

> http://www.essepuntato.it/lode/http://www.essepuntato.it/2008/12/earmark#Element

3.4 Content Negotiation via *.htaccess*

LODE can be used freely by third parties, as described in its documentation. In particular, it may be very usefully employed in conjunction with content negotiation mechanisms to display a human-readable version of an OWL ontology when the user accesses the ontology using a web browser, or to deliver the OWL ontology file itself when the user accesses the ontology using an ontology development tool such as *Protégé* [12] or the *NeOn Toolkit* [19].

LODE can be seen in action by opening, in a Web browser, the EARMARK ontology or any of the SPAR ontologies[20]. For instance, the URL "http://purl.org/spar/fabio" resolves, by content negotiation, to display the LODE HTML version of that ontology with the URL "http://www.essepuntato.it/lode/http://purl.org/spar/fabio". An implementation of such content negotiation is given in [2] by using the *.htaccess* file:

```
AddType application/rdf+xml .rdf
# Rewrite engine setup
RewriteEngine On
# Rewrite rule to serve HTML content
RewriteCond %{HTTP_ACCEPT} !application/rdf\+xml.*(text/html|
    application/xhtml\+xml)
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/.*
RewriteRule ^ontology$ http://www.essepuntato.it/lode/http://
    www.mydomain.com/ontology [R=303,L]
# Rewrite rule to serve RDF/XML content if requested
RewriteCond %{HTTP_ACCEPT} application/rdf\+xml
RewriteRule ^ontology$ ontology.owl [R=303]
# Choose the default response
RewriteRule ^ontology$ ontology.owl [R=303]
```

3.5 Community Uptake

LODE is currently used by the following projects:

- The *Semantic Publishing and Referencing* (*SPAR*) project, to generate the documentation for the entire set of SPAR ontologies (available at http://purl.org/spar) created to describe the publishing domain.
- The *VIVO* project[21], an interdisciplinary network enabling collaboration and discovery among scientists across all disciplines, using an open source Semantic Web application originally developed and implemented at the Cornell University. VIVO uses an ontology for data representation and sharing on the Web, documentation for which[22] is generated using LODE.

In addition, LODE has been listed in the *Lower Level Design Tools* page[23] of the official W3C Semantic Web wiki under the category *OWL ontology browsers*, and has been suggested in the Provenance Working Group wiki[24] as one of the documentation tools to use for the *Provenance Ontology* (*PROV-O*)[25].

[20] Semantic Publishing And Referencing (SPAR) ontologies: http://purl.org/spar
[21] VIVO: http://vivoweb.org
[22] VIVO OWLAPI Documentation: https://sourceforge.net/apps/mediawiki/vivo/ind-ex.php?title=Ontology_OWLAPI_Documentation
[23] LLDtools: http://www.w3.org/2001/sw/wiki/LLDtools
[24] Generating HTML documentation of OWL: http://www.w3.org/2011/prov/wiki/Ge-nerating_HTML_documentation_of_OWL
[25] PROV-O: http://www.w3.org/TR/prov-o

4 Experiments

In order to gather data about the usability of LODE, we undertook user testing. We asked 13 subjects to perform five unsupervised tasks (max. 5 minutes per task), involving ontology navigation through LODE documentation. There were no "administrators" observing the subjects while they were undertaking these tasks. All the subjects were volunteers who responded to personal e-mails or to an invitation sent to the semantic-web@w3.org and public-lod@w3.org lists.

For the tasks, we used a *medium-size* ontology, namely FaBiO, the *FRBR-aligned Bibliographic Ontology*[26], which is composed by 214 classes, 69 object properties, 45 data properties and 15 individuals. FaBiO was chosen because we expected most people involved in the experiments (primarily Semantic Web researchers and practitioners) to have familiarity with the domain it describes, i.e. bibliographic entities such as research papers, journal articles and books. In addition, FaBiO was also chosen because using an ontology larger than FaBiO would have required more time to complete the tasks, potentially reducing the number of users willing to complete the evaluation, and thus reducing the number of useful data for the evaluation.

The tasks given to the subjects are shown in Table 1. This set of tasks was designed to exploring the LODE capabilities in creating a human-readable documentation and in browsing ontologies.

Table 1. The five tasks subjects performed in the user testing session

Task 1	Describe the main aim of the ontology.
Task 2	Describe what the class *doctoral thesis* defines.
Task 3	Describe what the object property *has subject term* describes, and record its domain and range classes.
Task 4	Record the class having the largest number of direct individuals (i.e. individuals that belongs explicitly to that class and that are not inferable from its subclasses)
Task 5	Record all the subclasses and properties involving the class *item*

Task 1 is a pure descriptive activity that involves only the documentation produced by LODE, without using any navigational features such as Web links. Task 2 and 3 are similar to Task 1, but in addition typically requires the user to use some navigational facilities to reach the class *doctoral thesis* and the object property *has subject term*. Finally, Tasks 4 and 5 further assess how easily LODE enables users to browse the ontology and understand its structure. Our interest was to assess how well LODE helped users by producing documentation of an OWL ontology, enabling them more easily to browse and make sense of it.

[26] FaBiO ontology: `http://purl.org/spar/fabio`

The test session was structured as follows. We first asked subjects to complete a short multiple-choice questionnaire about their background knowledge and skills in OWL, ontology engineering and ontology documentation (max. 2 minutes). Then, as a warm-up task, we asked subjects to use LODE to explore the FOAF ontology[27], a relatively simple ontology, in order to become familiar with the structure of the documentation it produced, and with its navigation mechanisms (primarily, internal hypertext links) (max. 5 minutes). Then, as the real test, we asked subjects to complete the five tasks listed in Table 1 using the documentation of the FaBiO ontology created by LODE (ideally 2 minutes, max. 5 minutes, per task). Finally, we asked subjects to fill in two short questionnaires, one multiple choice and the other textual, to report their experience of using LODE to complete these tasks (max. 5 minutes). All the questionaries and all the outcomes of the experiments are available online[28].

5 Evaluation

Out of 65 tasks in total (5 tasks given to each of 13 subjects), 58 were completed successfully (i.e., the right answers were given), while 7 had incorrect answers or were not completed at all, giving an overall success rate of 89%. The 58 successes were distributed as follows: 13 (out of 13) in Task 1, 13 in Task 2, 13 in Task 3, 10 in Task 4 and 9 in Task 5.

The usability score for LODE was computed using the *System Usability Scale* (*SUS*) [4], a well-known questionnaire used for the perception of the usability of a system. It has the advantage of being technology independent (it has been tested on hardware, software, Web sites, etc.) and it is reliable even with a very small sample size [16]. In addition to the main SUS scale, we also were interested in examining the sub-scales of pure *Usability* and pure *Learnability* of the system, as proposed recently by Lewis and Sauro [13]. As shown in Table 2, the mean SUS score for LODE was 77.69 (in a 0 to 100 range), abundantly surpassing the target score of 68 to demonstrate a good level of usability [16]. The mean values for the SUS sub-scales Usability and Learnability were 76.4 and 82.7 respectively. In addition, two sub-scores were calculated for each subject by considering the values of the answers given in the background questionnaire, composed of ten questions about the subject's experience with ontologies and two questions about his/her experience with ontology documentation tools. We compared these sub-scores with the SUS values and the other sub-scales using the Pearson's r. We found a small negative correlation (between -0.34 and -0.14) between the experience sub-scores and the SUS values. This may show that the perceived usability of LODE does not depend upon any particular ability of subjects in the use of ontologies and ontology documentation tools, rather the opposite. However, each correlation measure appears to be not statistically significant and we need to enrich our dataset to come to a more precise conclusion.

[27] FOAF ontology: `http://xmlns.com/foaf/spec/index.rdf`
[28] `http://www.essepuntato.it/2012/04/lodeusertesting`

Table 2. System Usability Scale values and related sub-measures

Measure	Mean	Max. value	Min. value	Standard deviation
SUS value	77.7	92.5	57.5	12.5
Usability	76.4	90.6	56.3	12.8
Learnability	82.7	100	62.5	14.9

Table 3. Terms – three positive (+) and two negative (-) – mentioned by more than one individual in the final questionnaire responses

Term	Description	Frequency
Search (-)	No search function was provided to directly look for and access entities of the ontology. Users acknowledge that since the ontology is on a single web page, they could use (and in fact did use) the search function of the browser, but many still found it a missing feature	7 out of 11
Readability (+)	High praise was given to the clarity of the presentation, the intuitiveness of the organization, and the immediacy of identifying the sought information. The good typographical style of the output is clearly among the best qualities of LODE	5 out of 11
Links within the document (+)	The systematic use of internal links to the various features of the ontology was considered useful and immediately usable	4 out of 11
Scalability (-)	The LODE interface provides no links to entities provided by external, but linked ontologies. A highly modular ontology composed of a multiplicity of independent sub-ontologies is hard to navigate, and similarly the structure on a single page could make really large ontologies quite hard to access	3 out of 11
Single page (+)	Praises were given to the idea of placing all the content on a single Web page, which allows a multiplicity of approaches to accessing and reading the ontology, including visual transitions and scrolling that would not be possible if the ontologies had been presented in separate web pages.	2 out of 11

Axial coding of the personal comments expressed in the final questionnaires [18] revealed a small number of widely perceived issues. Only 11 out of the 13 subjects tested had meaningful comments that were used for the study, and, of the 15 terms that were identified as significant in the comments, only five (three positive and two negative) were mentioned by more than one individual (albeit sometimes with different words), as shown in Table 3.

6 Conclusions

Writing good documentation for ontologies manually is costly in effort, and is impracticable for an ontology that is under active development. In this paper we addressed this issue by introducing LODE, the *Live OWL Documentation Environment*, a Web application that automatically creates human-readable HTML ontology documentation on-the-fly from an OWL file of ontological axioms and annotations. We discussed how it can be used in conjunction with content negotiation that delivers LODE documentation to a browser window or the OWL file itself to an ontology editor. And we assessed LODE's usability and effectiveness in creating browsable ontology documentation by conducting user testing. The results of these tests are encouraging, demonstrating that LODE provides a stable service for the automatic creation of useful ontology documentation.

Some days before the deadline for the camera ready version of this paper, we performed another user testing session that compared LODE with similar tools for the automatic generation of ontology documentation, namely Parrot [20] and the OWLDoc-based Ontology Browser. A preliminary and informal evaluation of the outcomes of this new testing session confirmed the usability results highlighted in this paper. We also performed a comparison of the performances of the tools with users trying to carry out the tasks in Table 1. Early outcomes seem to confirm the usefulness of LODE when dealing with such tasks. This work will be reported at a later date. In future, we plan to extend LODE features to include suggestions highlighted by our users. In addition, we plan to conduct another user testing session to compare LODE with other ontology visualisation and browsing tools such as KC-Viz [14] and OWLViz [12].

Acknowledgements. Aspects of this work have been supported by the *JISC* (*Joint Information Systems Committee*) through grant support to DS. We thank all the people who took part to the user testing session.

References

1. Basca, C., Corlosquet, S., Cyganiak, R., Fernández, S., Schandl, T.: Neologism: Easy Vocabulary Publishing. In: Proceedings of the 4th Workshop on Scripting for the Semantic Web (2008), http://ceur-ws.org/Vol-368/paper10.pdf (last visited June 28, 2012)
2. Berrueta, D., Phipps, J.: Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Group Note (August 28, 2008), World Wide Web Consortium, http://www.w3.org/TR/swbp-vocab-pub/ (last visited June 28, 2012)
3. Boley, H., Hallmark, G., Kifer, M., Paschke, A., Polleres, A., Reynolds, D.: RIF Core Dialect. W3C Recommendation (June 22, 2010), World Wide Web Consortium, http://www.w3.org/TR/rif-core/ (last visited June 28, 2012)
4. Brooke, J.: SUS: a "quick and dirty" usability scale. In: Usability Evaluation in Industry, pp. 189–194. Taylor and Francis, London (1996) ISBN: 978-0748404600
5. d'Aquin, M., Motta, E.: Watson, more than a Semantic Web search engine. Semantic Web – Interoperability, Usability, Applicability 2(1), 55–63 (2011), doi:10.3233/SW-2011-0031

6. Di Iorio, A., Peroni, S., Vitali, F.: A Semantic Web Approach To Everyday Overlapping Markup. Journal of the American Society for Information Science and Technology 62(9), 1696–1716 (2011), doi:10.1002/asi.21591
7. Dragan, L., Handschuh, S., Decker, S.: The semantic desktop at work: interlinking notes. In: Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011 (2011), doi:10.1145/2063518.2063521
8. Dublin Core Metadata Initiative. DCMI Metadata Terms. DCMI Recommendation (2010), http://dublincore.org/documents/dcmi-terms/ (last visited June 28, 2012)
9. Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, Version 1.1. DCMI Recommendation (2010), http://dublincore.org/documents/dces/ (last visited June 28, 2012)
10. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web – Interoperability, Usability, Applicability 2(1), 11–21 (2011), doi:10.3233/SW-2011-0025
11. Horridge, M., Patel-Schneider, P.: OWL 2 Web Ontology Language: Manchester Syntax. W3C Working Group Note (October 27, 2009), World Wide Web Consortium, http://www.w3.org/TR/owl2-manchester-syntax/ (last visited June 28, 2012)
12. Knublauch, H., Horridge, M., Musen, M.A., Rector, A.L., Stevens, R., Drummond, N., Lord, P.W., Noy, N.F., Seidenberg, J., Wang, H.: The Protege OWL Experience. In: Proceedings of the OWLED 2005 Workshop on OWL: Experiences and Directions (2005), http://ceur-ws.org/Vol-188/sub14.pdf (last visited June 28, 2012)
13. Lewis, J.R., Sauro, J.: The Factor Structure of the System Usability Scale. In: Kurosu, M. (ed.) HCD 2009. LNCS, vol. 5619, pp. 94–103. Springer, Heidelberg (2009)
14. Motta, E., Mulholland, P., Peroni, S., d'Aquin, M., Gomez-Perez, J.M., Mendez, V., Zablith, F.: A Novel Approach to Visualizing and Navigating Ontologies. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 470–486. Springer, Heidelberg (2011)
15. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. International Journal of Metadata, Semantics and Ontologies 3(1), 37–52 (2008), doi:10.1504/IJMSO.2008.021204
16. Sauro, J.: A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices (2011) ISBN: 978-1461062707
17. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2), 51–53 (2007), doi:10.1016/j.websem.2007.03.004
18. Strauss, A., Corbin, J.: Basics of Qualitative Research Techniques and Procedures for Developing Grounded Theory, 2nd edn. Sage Publications, London (1998) ISBN: 978-0803959408
19. Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A.: Ontology Engineering in a Networked World. Springer, Berlin (2012) ISBN: 978-3642247934
20. Tejo-Alonso, C., Berrueta, D., Polo, L., Fernández, S.: Metadata for Web Ontologies and Rules: Current Practices and Perspectives. In: García-Barriocanal, E., Cebeci, Z., Okur, M.C., Öztürk, A. (eds.) MTSR 2011. CCIS, vol. 240, pp. 56–67. Springer, Heidelberg (2011)

Key-Concept Extraction for Ontology Engineering

Marco Rospocher, Sara Tonelli, Luciano Serafini, and Emanuele Pianta*

FBK-irst, Via Sommarive 18 Povo, I-38123,Trento, Italy
{rospocher,satonelli,serafini,pianta}@fbk.eu

Abstract. We present a framework for supporting ontology engineering by exploiting key-concept extraction. The framework is implemented in an existing wiki-based collaborative platform which has been extended with a component for terminology extraction from domain-specific textual corpora, and with a further step aimed at matching the extracted concepts with pre-existing structured and semi-structured information. Several ontology engineering related tasks can benefit from the availability of this system: ontology construction and extension, ontology terminological validation and ranking, and ontology concepts ranking.

1 Research Background

The development of ontologies is a crucial task in many fields in which knowledge needs to be shared and reused, such as knowledge management, e-commerce, database design, educational applications, and so on. Building an ontology is a complex task that requires a considerable amount of human effort.

We claim that the developing of ontologies can benefit from the automatic extraction of knowledge from documents related to the ontology domain and to its linking with available structured and semi-structured resources, such as *WordNet*[1] and *Wikipedia*[2].

We describe an on-line environment in which the ontology development process can be performed *collaboratively* in a Wiki-like fashion. To start the construction (or the extension) of an ontology, the user can exploit a domain corpus, from which the terminological component of the system automatically extracts a set of domain-specific key-concepts. These key-concepts are further disambiguated in order to be linked to existing external resources and obtain additional information such as the concept definition, the synonyms and the hypernyms. Finally, the user can easily select through the interface which concepts should be imported into the ontology.

The system support several ontology engineering tasks, as described in Section 3.

2 System Description

We present here the main characteristics of our online environment for key-concept based ontology engineering, focusing on its semantic components. In [1], we presented a preliminary version of the system, in which the evaluation and the disambiguation modules were missing.

* Work partially funded by the European Commission (contract FP7-248594, PESCaDO).

[1] http://wordnet.princeton.edu

[2] http://www.wikipedia.org

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 413–416, 2012.

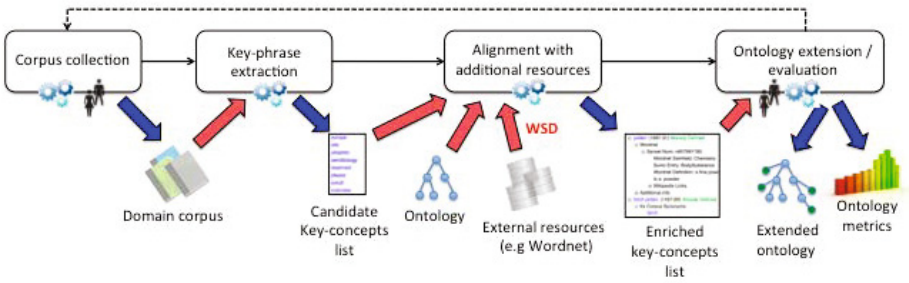


Fig. 1. System workflow for ontology building / extension

Starting from a domain-specific textual corpus, the system first extracts a list of key-concepts, then retrieves additional information from available lexical resources, and finally enables the user, through a user-friendly graphical interface, to (i) add new concepts to the ontology, or (ii) check the terminological coverage of an ontology with respect to the domain described in the text corpus.

The workflow has been made available in an on-line collaborative environment which can be potentially used for any kind of ontology domain. We have taken advantage of the existing infrastructure for conceptual modelling provided by MoKi [2],[3], a collaborative Mediawiki-based[4] tool for modeling ontological and procedural knowledge in an integrated manner. The main idea behind MoKi is to associate a wiki page to each basic entity of the ontology, i.e., concepts, object and datatype properties, and individuals. The environment has been enriched with a terminology-extraction component and with external lexical resources[5].

The overall workflow is displayed in Figure 1. As a preliminary step, the user should collect a domain corpus in one of the languages supported by the tool (currently, either English or Italian). On this corpus, the module for terminological extraction is first run to obtain a set of domain-specific terms that are seen as candidate concepts for the integration into the ontology. Such terms are obtained using KX [3], a robust system for keyphrase-extraction that first extracts n-grams from a corpus, then recognizes the multi-words by combining lexical-based filtering and black lists, and finally ranks the candidate key-concepts by relevance based on different ranking parameters. Hence, the output is a list of key-concepts ranked by relevance. The tool can be easily configured by the user through the online interface in order to define some basic selection criteria for the key-concepts. For instance, longer[6] key-concepts may be preferred over shorter ones. Note that the system is completely unsupervised and can be easily extended to handle multi-lingual documents.

In the following step, the key-concepts are linked to external resources to enrich the information provided to the user while building, extending or evaluating an ontology. To this purpose, the key-concepts are disambiguated by assigning them

[3] See http://moki.fbk.eu

[4] See http://www.mediawiki.org

[5] A demo of the tool is available at http://moki.fbk.eu/moki/tryitout2.0/. After logging in, go to *Import Functionalities*, and then *Extract new concepts from textual resources*.

[6] In terms of number of words that compose a key-concept.

a WordNet synset. Word sense disambiguation (WSD) is performed using WordNet::SenseRelate::WordToSet library[4], which has been developed in order to disambiguate a word given the textual context in which it appears. We apply this disambiguation procedure in a slightly different way: we build a context by considering the 10 top-ranked key-concepts returned by KX, and disambiguate all other key-concepts in the list based on them. The intuition behind this is that the top-ranked concepts should be the most representative for the domain, and therefore they should build the most appropriate context to disambiguate other concepts from the same domain.

Once a key-concept is associated to a unique WordNet sysnet, additional information from WordNet can be retrieved, such as the gloss, the hypernyms and the synonym list for the given key-concept. All this information is displayed to the user after the extraction process. An example is displayed in Fig. 2, showing the top-ranked key-concepts extracted from a corpus about pollens. In addition, we exploit *BabelNet*[5] (an automatic alignment between Wikipedia pages and WordNet synsets), in order to provide for each synset a link to the corresponding Wikipedia page.

Once a set of key-concepts has been extracted from a domain corpus and enriched with additional information from WordNet and Wikipedia, the platform relies on this information to perform different processes. If a pre-defined ontology has been uploaded, each concept and its WordNet synonyms (if any) are matched against the concepts currently defined in the ontology, in order to understand if the concept (or a synonym of it) is already defined in the ontology under development. If a match is recognized, the key-concept is marked with an X (see Fig. 2), thus reducing the chance that unnecessary duplicated content is added to the ontology. In the current version of the system, the matching is performed by applying normalized matching techniques on concepts (and synonyms) labels.

The matching is exploited in two ways: for *ontology extension*, the user can decide whether to automatically add some of the extracted un-matched concepts to the ontology. For *ontology evaluation*, some metrics derived from standard precision, recall, and F1 measures can be computed based on the matching between ontology concepts and corpus key-concepts extracted in the previous step. These measures aim at estimating how well a given ontology terminologically covers the domain described by the corpus, i.e. to which extent the concepts described in the ontology are appropriate and complete with respect to the domain.

3 System Demonstration

We will demonstrate the support provided by our online environment for ontology engineering in the three following tasks:

Boosting of Ontology Construction and Extension. This is the typical situation occurring when users want to build a domain ontology and no structured resources are already available, or when they want to iteratively extend an existing ontology. A sample corpus will be uploaded by the system, together with an ontology (optional). We will show the support provided by the tool in identifying and adding new domain concepts extracted from the corpus to the given ontology.

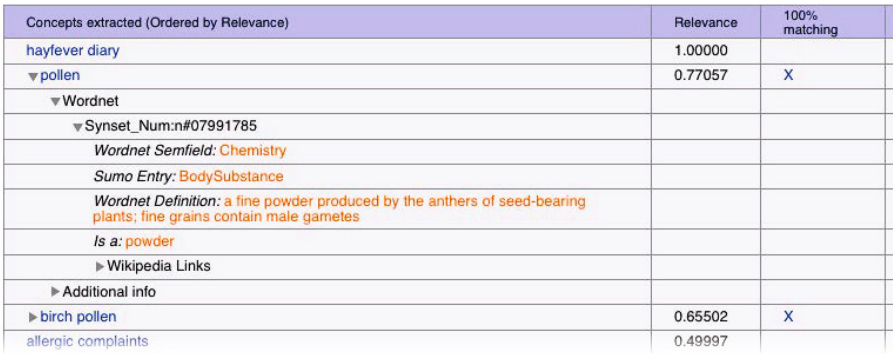


Fig. 2. Screenshot of matching output

Ontology Terminological Evaluation and Ranking. Based on the matching between an ontology and corpus key-concepts, the system can compute a set of metrics inspired by precision/recall/F1 to terminologically evaluate an already existing ontology against the domain described by the corpus, or to rank different candidate ontologies according to how well they terminologically cover the domain. This may be helpful when deciding whether to reuse ontologies already available on the web. We will upload a domain corpus, and compute the ontology metrics for different candidate ontologies, showing how to interpret the results obtained.

Ranking of Ontology Concepts. Finally, we will show how to evaluate the relevance of the concepts in an ontology with respect to the domain described by a corpus. Given that an ontology can include thousands of concepts, it is not always easy to understand if some of them are more relevant than others, thus representing a sort of core knowledge of the domain ontology (information which usually is not explicitly encoded in ontologies). We will upload a domain corpus and an ontology, showing which are the most relevant domain-wise concepts of the ontology according to the tool.

References

1. Tonelli, S., Rospocher, M., Pianta, E., Serafini, L.: Boosting collaborative ontology building with key-concept extraction. In: Proceedings of 5th IEEE International Conference on Semantic Computing (2011)
2. Ghidini, C., Rospocher, M., Serafini, L.: MoKi: A Wiki-Based Conceptual Modeling Tool. In: Proc. of ISWC 2010, Posters and Demonstrations Track, Shanghai, China (2010)
3. Pianta, E., Tonelli, S.: KX: A flexible system for Keyphrase eXtraction. In: Proc. of SemEval 2010, Task 5: Keyword Extraction from Scientific Articles, Uppsala, Sweden (2010)
4. Pedersen, T., Banerjee, S., Patwardhan, S.: Maximizing semantic relatedness to perform word sense disambiguation. Technical Report UMSI 2005/25, University of Minnesota (2005)
5. Navigli, R., Ponzetto, S.P.: Babelnet: Building a very large multilingual semantic network. In: Proc. of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden (2010)

Latest Developments to LODE

Silvio Peroni[1], David Shotton[2], and Fabio Vitali[1]

[1] Department of Computer Science, University of Bologna, Italy
[2] Department of Zoology, University of Oxford (UK)
{essepuntato,fabio}@cs.unibo.it, david.shotton@zoo.ox.ac.uk

Abstract. LODE, the *Live OWL Documentation Environment*, is a service for the generation of human-readable documentation of OWL ontologies and RDFS vocabularies. It automatically extracts classes, object properties, data properties, named individuals, annotation properties, meta-modelling (punning), general axioms, SWRL rules and namespace declarations and renders them as an HTML page designed for easy browsing and navigation by means of embedded links. In this paper, we present an overview of the tool, in particular focusing on the features introduced in the latest available version.

Keywords: LODE, OWL ontologies, Web GUI, Web tools, ontology documentation, structural reasoner.

1 Introduction

Consulting its human-readable documentation is among the first activities to perform on an ontology so as to understand its domain, and whether it describes this domain appropriately. Although most ontologies developed for Linked Data usage are accompanied by a comprehensive set of Web pages describing their theoretical backgrounds and the features of their developed entities, problems arise when we look at partially-developed models[1]. Writing proper documentation manually requires effort, and re-writing it every time an ontology under development is modified is hardly practical and thus very seldom performed. Rather, the natural language documentation of ontologies is normally only published once the ontologies have become stable.

To address this issue, tools have recently been developed for the automatic generation of documentation from the axioms and annotations of an ontology. These include *Neologism*[2] [1], *OWLDoc*[3] and Parrot[4] [4]. Similarly, we have developed *LODE*, the *Live OWL Documentation Environment*[5], an online service that takes any well-formed OWL ontology, and generates a single human-readable HTML page providing for browsing and navigation by means of embedded links. In our

[1] W3C PROV WG ISSUE-270: http://www.w3.org/2011/prov/track/issues/270
[2] Neologism: http://neologism.deri.ie
[3] OWLDoc: http://code.google.com/p/co-ode-owl-plugins/wiki/OWLDoc
[4] Parrot: http://ontorule-project.eu/parrot/parrot
[5] LODE: http://www.essepuntato.it/lode

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 417–420, 2012.

EKAW 2012 in-use paper [3], we described in detail the features of LODE and assessed its usability through an empirical evaluation with test users. During that evaluation, we gathered relevant feedback about usability and missing features which would improve its usefulness. Following these suggestions, we created a new improved version (v. 1.1, dated 1 July 2012) of LODE that is available on the Web and described in this paper.

The rest of the paper is structured as follows. In Section 2 we briefly introduce LODE, while in (SectionNewFeatures) we introduce in detail the new features implemented in the latest version (v. 1.1). Finally, in Section 4 we sketch out some conclusions and some future developments.

2 LODE: The Rationale

LODE is a Web service developed in Java and mainly based on XSLT technology. It takes an RDF/XML linearisation of an OWL ontology created by the OWL API[6] [2], and applies an XSLT transformation that returns a human-readable HTML page containing the ontology documentation. LODE is currently used in different projects such as *SPAR*[7], *PROV-O*[8], *VIVO*[9] and *Tipalo*[10].

The following pseudo-URL describes how to call LODE:

http://www.essepuntato.it/lode/optional-parameters/ontology-url

LODE provides different configurations, actionable through particular parameters (the slash-separated *optional-parameters* in the pseudo-URL), in order to satisfy different needs, e.g. to process *ontology-url* when it is linearised in a format such as Turtle (parameter *owlapi*), to document the axioms of its imported ontologies (parameter *imported*) or of its closure (parameter *closure*), additionally to document the inferred axioms of *ontology-url* (parameter *reasoner*), and to generate the *ontology-url* documentation in a language different form English (parameter *lang*). The usage of all these parameters is introduced in [3].

3 New Features

In what follows, we briefly list the main new features implemented in the latest version of LODE (v 1.1). These features improve the existing documentation and also make possible new user actions.

XSLT structural reasoner. LODE has been extended with an additional XSLT module[11] that implements a preliminary version of an OWL structural reasoner. This infers new axioms directly from the RDF/XML ontology source, in particular the symmetric inferences of the following properties: *owl:disjointWith*,

[6] The OWL API: http://owlapi.sourceforge.net
[7] Semantic Publishing and Referencing Ontologies: http://purl.org/spar
[8] PROV-O: The PROV Ontology: http://www.w3.org/TR/prov-o
[9] VIVO: connect, share, discover: http://vivoweb.org
[10] STLab Tool Tipalo: http://wit.istc.cnr.it/stlab-tools/tipalo
[11] http://speronitomcat.web.cs.unibo.it:8080/LODE/structural-reasoner.xsl

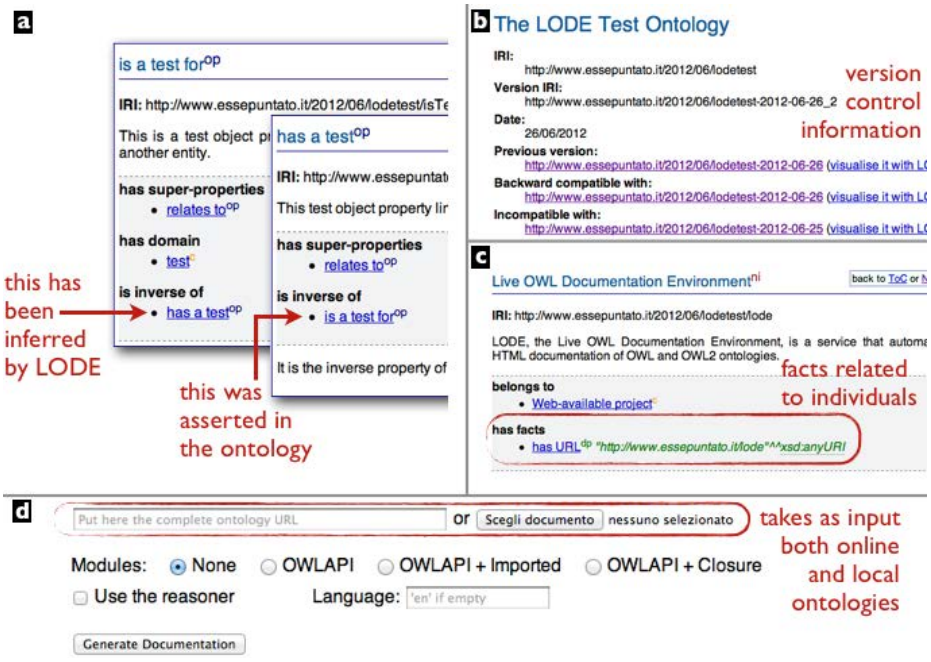


Fig. 1. a) The *owl:inverseOf* axiom inferred by LODE. b) Information about the current and the previous versions of the ontology. c) The facts related to individuals asserted in the ontology. d) The Web GUI to use LODE via its website.

owl:sameAs, *owl:equivalentProperty, owl:equivalentClass* (currently without handling restrictions)and*owl:inverseOf* (the latter shown in Fig. 1 (a)[12]). The XSLT structural reasoner temporarily stores the asserted axioms of the ontology about disjointness, sameness, equivalence and inversion so as to allow the checking and retrieval of both the asserted axioms (e.g. *x owl:inverseOf y*) and the related symmetric ones (e.g. *y owl:inverseOf x*) during the generation of the documentation. These features rely on functions that can be used for more general and parametric purposes, thereby enabling a quick and easy way to extend the structural reasoner to verify additional symmetric inferences.

Additional Information about the Ontology. OWL 2 introduces new built-in annotation properties to record the various versions of an ontology, in particular: *owl:versionIRI*, *owl:priorVersion*, *owl:backwardCompatibleWith*, *owl:incompatibleWith*. As shown in Fig. 1 (b), the new version of LODE shows all this information. In addition, it now also displays *dc:publisher* and *dcterms:publisher* annotations whenever these are present in the ontology.

Facts about Individuals. The documentation produced by LODE has been extended to add assertions that involve individuals (i.e. instances of classes)

[12] The ontology used in the examples is the *LODE Test Ontology*, available at `http://www.essepuntato.it/2012/06/lodetest`

within the ontology. To this end, a new field labelled "has facts" has been added, as shown in Fig. 1 (c), where the statement `lodetest:lode lodetest:hasURL "http://www.essepuntato.it/lode"^^xsd:anyURI` has been documented.

Web GUI. As displayed in Fig. 1 (d), the LODE homepage now contains a simple form that allows users to access the service directly from the web page. In addition to permitting use of all the functions introduced in Section 2 when specifying the URL of an ontology accessible from the Web, the form can also be used to generate documentation of ontologies stored locally, by browsing for them on the local hard drive and then uploading them to the LODE server.

4 Conclusions

In this paper we introduced some features that have been implemented in LODE, a Web application that automatically creates human-readable HTML ontology documentation on-the-fly from an OWL file of ontological axioms and annotations. These new capabilities enable the generation of more complete documentation – now enriched by inferred axioms, version control information and facts about individuals within the ontology – and simplify the use of the tool through a form-based Web interface. In future, we plan to further improve LODE. High on our list of proposed improvements are the inclusion of a search function, inclusion of additional information defined according to other annotation schemas, e.g. *Content Pattern Annotation Schema*[13] and *VANN*[14], and extension of the structural reasoner, e.g. to infer the domain/range classes for inverse properties and the equivalence/subclass relations for classes even in presence of restrictions.

References

1. Basca, C., Corlosquet, S., Cyganiak, R., Fernández, S., Schandl, T.: Neologism: Easy Vocabulary Publishing. In: Proceedings of the 4th Workshop on Scripting for the Semantic Web (2008), http://ceur-ws.org/Vol-368/paper10.pdf (last visited June 27, 2012)
2. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web – Interoperability, Usability, Applicability 2(1), 11–21 (2011), doi:10.3233/SW-2011-0025.
3. Peroni, S., Shotton, D., Vitali, F.: The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation. In: ten Teije, A., et al. (eds.) EKAW 2012. LNCS (LNAI), vol. 7603, pp. 398–412. Springer, Heidelberg (2012)
4. Tejo-Alonso, C., Berrueta, D., Polo, L., Fernández, S.: Metadata for Web Ontologies and Rules: Current Practices and Perspectives. In: García-Barriocanal, E., Cebeci, Z., Okur, M.C., Öztürk, A. (eds.) MTSR 2011. CCIS, vol. 240, pp. 56–67. Springer, Heidelberg (2011)

[13] http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl
[14] http://purl.org/vocab/vann

YAM++ : A Multi-strategy Based Approach for Ontology Matching Task

DuyHoa Ngo and Zohra Bellahsene

Université Montpellier 2, INRIA, LIRMM
Montpellier - France
{firstname.name@lirmm.fr}

Abstract. In this paper, we present the capability of our ontology matching tool YAM++. We show that YAM++ is able to discover mappings between entities of given two ontologies by using machine learning approach. Besides, we also demonstrate that if the training data are not available, YAM++ can discover mappings by using information retrieval techniques. Finally, we show that YAM++ is able to deal with multi-lingual ontologies matching problem.

1 Introduction

There are many challenges in ontology matching task. A matcher tool can be seen as a combination of three sub-matchers such as: Element level matcher, Structural level matcher and Semantical matcher. Generally, element level matcher discovers mappings by comparing annotation (i.e., labels, comments) of entities. It may use many different similarity metrics to handle the high terminological heterogeneity of ontologies. A challenging issue here is how to combine different metrics effectively. Additionally, if labels of entities in ontologies are represented by different languages, the matching process is even more difficult. Structural level matcher discovers mappings of entities based on analyzing their structural information. However, according to [6], most of them don't perform well when the structures of ontologies are different. Moreover, structural matcher is error-prone, since it strongly depends on initial mappings provided by element level matcher. Semantical matcher is mainly used to refine candidate mappings. It exploits the semantic constraints between entities in ontologies in order to remove inconsistent mappings. It is a NP-complete problem [2] to find the global optimization results (i.e., the minimum set of inconsistent mappings). To handle these challenges, we propose our solution as follows:

- If labels of entities are written by different languages, we use a multi lingual translator to translate labels from other languages into English.
- We use machine learning based approach to combine different similarity metrics at element level matcher. In case we don't have training data, we propose a similarity metrics based on information retrieval techniques.
- We use a graph matching, in particular similarity propagation method, which is known as the most stable method dealing with structural information to discover additional mappings.
- In terms of semantic refinement, we use the Global Optimal Diagnosis method [2].

The rest of the paper is organized as follows. In Section 2, we present an overview of YAM++ system. Section 3 contains the demonstration scenarios. In section 4, we summarize our contributions.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 421–425, 2012.

2 YAM++ Overview

Fig. 1 depicts the main components of the YAM++ system. YAM++ discovers mappings between two input ontologies by two matchers: element level matcher and structural level matcher. The combination of the mappings resulting from element level and structural level are then revised by the semantical matcher in order to remove inconsistent mappings.

- At element level, input ontologies are processed in order to extract annotation information for every entity. Based on these information, similarity score between entities are computed by different terminological metrics. Here, similarity metrics can work independently or can be combined by combination methods in order to produce mappings at element level. Currently, YAM++ supports machine learning based combination methods such as Decision Tree, SVM, NaiveBayes[1], etc. In that case, the training data are provided by the user or are taken from knowledge base (KB) resources. Otherwise, by default, YAM++ performs our proposed matching method based on information retrieval technique.
- At structural level, input ontologies are parsed and transformed into graph data structure. Then, YAM++ takes the mappings resulting from element level as initial mappings to run a similarity propagation process. The propagation algorithm here is inspired from the well known Similarity Flooding algorithm [7]. See [5] for more detail about our extension of similarity propagation method.
- In semantical checking module, we make use of global constraint optimization method proposed in Alcomox tool[2]

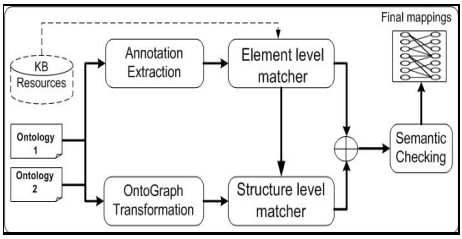


Fig. 1. YAM++ architecture

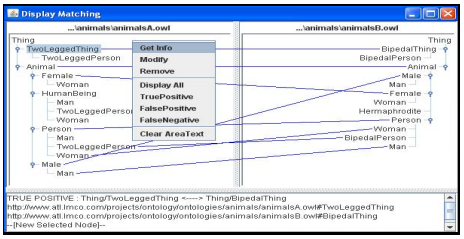


Fig. 2. Graphical User Interface

The resulting mappings of the matching process are displayed in graphical user interface (Fig. 2). The user can judge a mapping as correct or not according to his/her knowledge of ontologies'domain. The user can also modify, remove incorrect mappings or add new mappings with the help of command operations shown in YAM++'s menu.

3 Demonstration Scenarios

YAM++ has been implemented in Java, offering a GUI to select different configuration options and display the matching results. In this demo, we will show the following capabilities of YAM++: (i) matching with machine learning method, (ii) matching with information retrieval method, and (iii) matching with multi-lingual ontologies.

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://web.informatik.uni-mannheim.de/alcomo/

In order to evaluate the matching quality of YAM++, we compute three standard evaluation metrics(i.e., H-mean precision, recall and Fmeasure) on two data sets **Conference**[3] and **Multifarm**[4] in order to compare the matching quality of our system with other participants of OAEI campaign[5].

3.1 Matching with Machine Learning Method

In the first scenario, we assume that the user has several gold standard data sets, which consist of two ontologies and a corresponding alignment provided by experts of the domain. The user may think that he/she can study some matching patterns from the existing data sets to discover new mappings from new matching scenario with to-be-matched ontologies. Obviously, manually finding mappings is not applicable with the big size of ontologies. Therefore, the user would like to use the existing data as training data to train a machine learning model. Then, the learning model will automatically examine every pair of entities from to-be-matched ontologies and classify them into match or not.

Matcher	Prec.	$F_{0.5}$Meas.	Rec.	Prec.	F_1Meas.	Rec.	Prec.	F_2Meas.	Rec.
YAM++	.8	.73	.53	.78	**.65**	.56	.78	.59	.56
CODI	.74	.7	.57	.74	.64	.57	.74	.6	.57
LogMap	.85	**.75**	.5	.84	.63	.5	.84	.54	.5
AgrMaker	.8	.69	.44	.65	.62	.59	.58	**.61**	.62
BaseLine$_2$	**.79**	**.7**	**.47**	**.79**	**.59**	**.47**	**.79**	**.51**	**.47**
MaasMtch	.83	.69	.42	.83	.56	.42	.83	.47	.42
BaseLine$_1$	**.8**	**.68**	**.43**	**.8**	**.56**	**.43**	**.8**	**.47**	**.43**
CSA	.61	.58	.47	.5	.55	.6	.5	.58	.6
CIDER	.67	.61	.44	.64	.53	.45	.38	.48	.51
MapSSS	.55	.53	.47	.55	.51	.47	.55	.48	.47
Lily	.48	.42	.27	.36	.41	.47	.37	.45	.47
AROMA	.35	.37	.46	.35	.4	.46	.35	.43	.46
Optima	.25	.28	.57	.25	.35	.57	.25	.45	.57
MapPSO	.28	.25	.17	.21	.23	.25	.12	.26	.36
LDOA	.1	.12	.56	.1	.17	.56	.1	.29	.56
MapEVO	.27	.08	.02	.15	.04	.02	.02	.02	.02

Fig. 3. Result on Conference 2011 track

Matcher	Threshold	Precision	F0.5-measure	F1-measure	F2-measure	Recall
YAM++	0	0.8	0.78	0.74	0.71	0.69
LogMap	0	0.82	0.75	0.66	0.59	0.55
CODI	0	0.74	0.7	0.64	0.6	0.57
Hertuda	0	0.79	0.7	0.6	0.53	0.49
WeSeE	0.33	0.71	0.65	0.59	0.53	0.5
Baseline2	0	0.79	0.7	0.59	0.51	0.47
LogMapLt	0	0.73	0.67	0.59	0.53	0.5
GOMMA	0	0.84	0.71	0.58	0.49	0.44
AUTOMSv2	0	0.79	0.68	0.56	0.47	0.43
Baseline1	0	0.8	0.68	0.56	0.47	0.43
MaasMtch	0.89	0.74	0.64	0.54	0.46	0.42
MapSSS	0	0.5	0.5	0.5	0.51	0.51
MapPSO	0.67	0.1	0.08	0.07	0.06	0.05
MapEvo	0.82	0.04	0.03	0.03	0.02	0.02

Fig. 4. Result on Conference 2011.5 track

Based on this idea, YAM++ provides different kinds of similarity metrics, which can be used to represent different features of each pair of entities from two to-be-matched ontologies. For example, from the YAM++'s control panel, the user can select string-based metrics such as Levenstein, QGrams[6]; linguistic-based metrics on Wordnet[7] such as Lin, Wu-Palmer, etc. The matching process could be decomposed into three steps: (i) Learning phase. The user will select a set of similarity metrics, a gold standard dataset and a machine learning model. (ii) YAM++ creates training data and performs a training process. (iii) Classification phase. YAM++ generates a classification, which is used to classify and produce mappings between the input ontologies. Furthermore, if the user chooses the option of running structural method from YAM++'s control panel for the next step, these mappings will be passed to input of the similarity propagation process. Finally, the mapping results will be shown in the display for user's judgment. More details about the similarity metrics and their combinations are described in [3,4].

[3] http://oaei.ontologymatching.org/2011/conference/index.html
[4] http://web.informatik.uni-mannheim.de/multifarm/
[5] http://oaei.ontologymatching.org/2011.5/
[6] http://secondstring.sourceforge.net/
[7] http://wordnet.princeton.edu/

Fig. 3 shows the comparison result of YAM++ with other participants on the Conference track in OAEI 2011 campaign[8]. This was the first time we participate in the OAEI competition. At this time, YAM++ stayed in **Top2** among all participants. Especially, in terms of F_1 measure, it achieved the **best** matching tool title.

3.2 Matching with Information Retrieval Method

In this second scenario, we assume that related gold standard data sets are not available. In that case, the method of using machine learning model is not applicable. Instead, YAM++ provides matching methods based on information retrieval techniques. In particular, YAM++ applies information retrieval technique on annotation information of entities to determine amount of informativeness of tokens within the input ontologies. Then, an extended metric of Tversky's theory [11] has been developed to compute similarity between entities' labels. Similar to the first scenario, the user can select similarity propagation to discover more mappings by exploiting structural information of entities.

Fig. 4 shows the comparison result of YAM++ with other participants on the Conference track in OAEI 2011.5 campaign[9]. This was the second time we participate to the OAEI competition with non learning YAM++ system. At this time, YAM++ obtained the **best** matching result and dominated all other participants.

3.3 Matching with Multi-lingual Ontologies

In the last scenario, we show the ability of YAM++ to work withmulti-lingual ontologies matching. When the user provides two to-be-matched ontologies, YAM++ read annotations of entities in order to determine which language is used in each ontology. Once the languages are defined, YAM++ uses Microsoft Bing Translator tool[10] to translate all labels from other languages to English. After that, YAM++ discovers mappings between entities based on their translated labels by using proposed information retrieval methods. The returned mappings are passed as input to similarity propagation process to discover more mappings.

	Different ontologies (type i)				Same ontologies (type ii)			
Matching system	Size	Precision	Recall	F-measure	Size	Precision	Recall	F-measure
YAM++	1838	0.54	0.39	0.45	5838	0.93	0.48	0.63
AUTOMSv2	746	0.63	0.25	0.36	1379	0.92	0.16	0.27
WeSeE	4211	0.24	0.39	0.29	5407	0.76	0.36	0.49
CIDER	737	0.42	0.12	0.19	1090	0.66	0.06	0.12
MapSSS	1273	0.16	0.08	0.10	6008	0.97	0.51	0.67
LogMap	335	0.36	0.05	0.09	400	0.61	0.02	0.04
CODI	345	0.34	0.04	0.08	7041	0.83	0.51	0.63
MaasMtch	15939	0.04	0.28	0.08	11529	0.23	0.23	0.23
LogMapLt	417	0.26	0.04	0.07	387	0.56	0.02	0.04
MapPSO	7991	0.02	0.06	0.03	6325	0.07	0.04	0.05
CSA	8482	0.02	0.07	0.03	8348	0.49	0.36	0.42
MapEVO	4731	0.01	0.01	0.01	3560	0.05	0.01	0.02

Fig. 5. Comparison on Multifarm 2011.5 track

[8] http://oaei.ontologymatching.org/2011/
[9] http://oaei.ontologymatching.org/2011.5/
[10] http://www.microsofttranslator.com/

Fig. 5 shows the comparison result between YAM++ and other participants of OAEI 2011.5 campaign on Multifarm data sets. There are two types of evaluation. In the first type, all matching tools deal with different ontologies with different languages. In this evaluation, YAM++ achieved the **best** matching quality (Fmeasure = 0.45). In the second type, all tools discover mappings of the same ontologies but translated in different languages. In this evaluation, YAM++ obtained the **second** position among all participants.

4 Conclusion

In this paper, we present YAM++ - an ontology matching tool, which supports: (i) discovering alignment of ontologies by machine learning approaches; (ii) discovering alignment of ontologies by generic methods without using learning techniques; (iii) discovering alignment of ontologies represented in different languages. Moreover, our tool produced high matching quality results on Benchmark, Conference and Multifarm tracks in comparison with other participants on OAEI 2011 and OAEI 2011.5 campaigns. The running tool can be found at http://www2.lirmm.fr/∼dngo/.

References

[1] Tversky, A.: Features of similarity. Psychological Review, 327–352 (1977)
[2] Meilicke, C.: Alignment incoherence in ontology matching. PhD Thesis, University of Mannheim, Chair of Artificial Intelligence (2011)
[3] Ngo, D., Bellahsene, Z., Coletta, R.: A Flexible System for Ontology Matching. In: Nurcan, S. (ed.) CAiSE Forum 2011. LNBIP, vol. 107, pp. 79–94. Springer, Heidelberg (2012)
[4] Ngo, D.H., Bellahsene, Z., Coletta, R.: A generic approach for combining linguistic and context profile metrics in ontology matching. In: ODBASE, pp. 800–807 (2011)
[5] Ngo, D.H., Bellahsene, Z., Coletta, R.: Yam++ results for oaei 2011. In: OM (2011)
[6] Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
[7] Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: ICDE, pp. 117–128 (2002)

User-Friendly Pattern-Based Transformation of OWL Ontologies

Ondřej Šváb-Zamazal, Marek Dudáš, and Vojtěch Svátek

Department of Information and Knowledge Engineering,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{ondrej.zamazal,xdudm12,svatek}@vse.cz

1 Motivation for Pattern-Based Ontology Transformation

The high expressivity of the OWL ontology language often allows to express the same conceptualisation in different ways. A simple example is the difference between 'class-centric' and 'property-centric' modelling styles, such that the same notion is modelled as a class in the former (e.g. 'Purchase') and an object property in the latter (e.g. 'bought_from'). Similarly, concept subordination can be expressed via a subclass hierarchy or via individuals connected by a dedicated property (as in SKOS). Such heterogeneity is an obstacle to reusing ontologies in advanced semantic web scenarios. In particular (as mentioned in [1]), two ontologies modelled in different styles are difficult to *match* or to *import* into one another. Furthermore, opting for a style when designing an ontology may have an impact on the applicability and performance of *reasoners*, as some features cause performance problems for certain reasoners. Finally, *human users* may also prefer viewing ontologies in a certain form, possibly 'folding' parts of their complexity. Semi-automatic transformation of the modelling style of existing ontologies, with the help of tools to be presented in the demo, will alleviate such problems.

In the paper we first overview the core framework and then focus on user-oriented tools that allow to perform ontology transformation and edit transformation patterns in a friendly way.

2 *PatOMat* Transformation Framework

The central notion in the *PatOMat* framework[1] is that of *transformation pattern* (TP). A TP contains two *ontology patterns* (the source OP and the target OP) and the description of transformation betweem them, called pattern transformation (PT). The representation of OPs is based on the OWL 2 DL profile, except that *placeholders* are allowed in addition to concrete OWL entities. An OP consists of *entity declarations* (of placeholders and/or concrete entities), *axioms* and *naming detection patterns*; the last capture the naming aspect of the

[1] [4] provides more details about the (earlier version of the) framework, and at http://owl.vse.cz:8080/tutorial/ there is a fully-fledged tutorial for the current version.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 426–429, 2012.

OP, which is important for its detection. A PT consists of a set of *transformation links* and a set of *naming transformation patterns*. Transformation links are either *logical equivalence relationships* or *extralogical relationships* (holding between two entities of different type). Naming transformation patterns serve for generating names for target entities. Naming patterns range from *passive naming operations*, such as detection of a head noun for a noun phrase, to *active naming operations*, such as derivation of verb form of a noun.

The framework prototype implementation is available either as a *Java library* or as three *RESTful services*.[2] The Java library is directly used in the GUIPOT and XD Transformation Wizard tools, see Section 3. The whole transformation is divided into three steps that correspond to the three core services:

- *OntologyPatternDetection* service takes the TP and an ontology on input, and returns the binding of entity placeholders on output, in XML. The naming detection patterns of the source OP are first processed. As a result of applying the naming aspect, bound placeholders arise that are placed to the FILTER component of a SPARQL query (generated according to axioms in the source OP) before its execution.
- *InstructionGenerator* service takes the binding of placeholders and the TP on input, and returns transformation instructions on output.
- *OntologyTransformation* service takes transformation instructions and the original ontology on input, and returns the transformed ontology on output.

The third service is partly based on OPPL[3] and partly on our specific implementation over OWL-API.[4] In contrast to plain OPPL, we use naming constraints and we decompose the process of transformation into parts, which enables user intervention within the whole workflow.

3 User-Oriented Tools for Ontology Transformation

The *GUIPOT* (Graphical User Interface for Pattern-based Ontology Transformation) Protégé plugin, see Figure 1, was developed in order to bring ontology transformation into the standard working environment of a knowledge engineer. The screenshot demonstrates how an ontology fragment expressing the notion of 'paper whose decision is rejection' is transformed into an explicit class RejectedPaper, using a generic transformation pattern (with heuristics for naming detection/transformation included). After loading the transformation pattern, GUIPOT displays a list of pattern instances of the source OP detected[5] in the given ontology. Detected pattern instances can be manually adjusted or even new pattern instances can be created before the transformation. By selecting an

[2] All accessible via the web interface at `http://owl.vse.cz:8080/`

[3] `http://oppl2.sourceforge.net/`

[4] `http://owlapi.sourceforge.net/`

[5] The user can turn on *recursive detection* of pattern occurrences (over a taxonomy), as supported by a reasoner.

instance, the detected entities (placeholder bindings) are highlighted in a classical hierarchy view and also visualized using the OntoGraf plugin[6] on the left part of the plugin window. The right part of the window shows the ontology after transformation, with affected entities indicated by red arrows.

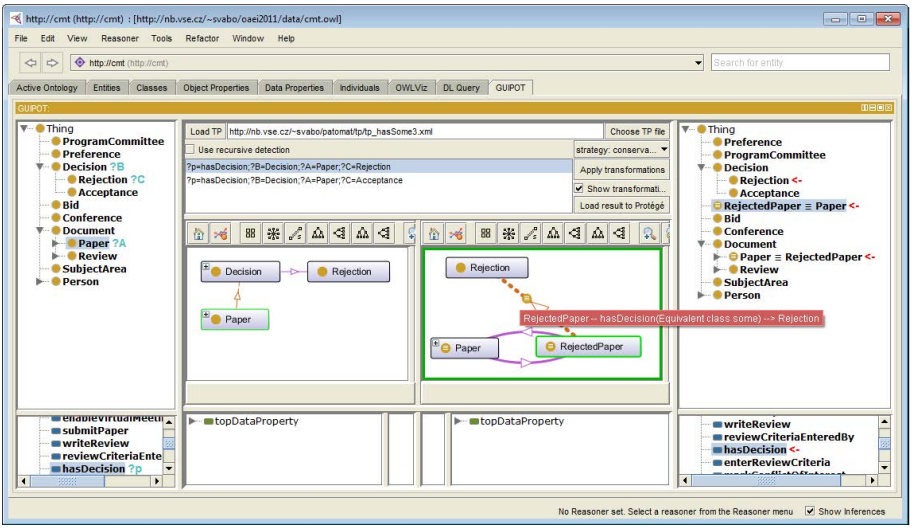


Fig. 1. GUIPOT in action

Aside from GUIPOT as generic graphical interface, we also aimed at support for specific ontological engineering scenarios. One of them is the import of *ontology content patterns* [2] (as small 'best-practice' chunks of ontological knowledge) into legacy ontologies, which, in turn, often have to undergo transformation. For example, when an ontology is to be adapted to the *AgentRole* pattern,[7] often seemingly 'natural' classes have to be changed to 'role' classes. To make such operations rapid and smooth, we decided to closely integrate *PatOMat* with *XDtools*,[8] an ontological engineering environment specifically tailored for ontology content patterns (CPs) exploitation. The tool provides the NeOn toolkit perspective that includes, among others, our *Transformation Wizard*.

When the wizard is invoked, the user chooses the *content pattern* to be imported into an ontology. On the first page of the wizard (see Figure 2) s/he then selects an ontology and a *transformation pattern*. The second page offers the *pattern instances* returned by the detection phase. By clicking on an entity the user can also display its usage within the ontology. The final page of the wizard offers the selection of a finer transformation strategy.

In order to support the *authoring and update* of transformation patterns, we also developed a *Transformation Pattern Editor* (TPE). It allows for their graphical modeling and export/import from/to the (XML-based) TP notation. TPE

[6] http://protegewiki.stanford.edu/wiki/OntoGraf
[7] http://ontologydesignpatterns.org/wiki/Submissions:AgentRole
[8] http://extremedesign.sourceforge.net

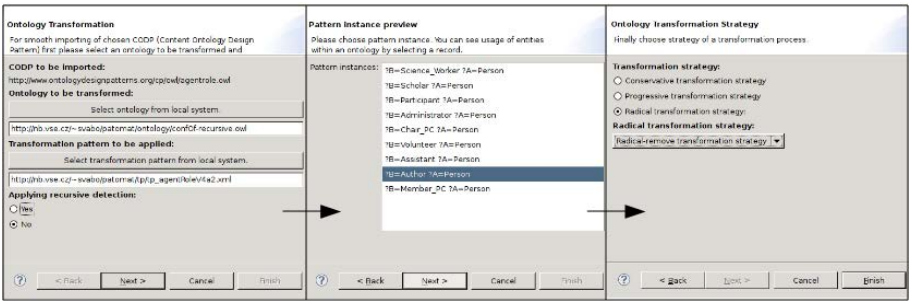


Fig. 2. Step-by-step ontology transformation using XD Transformation Wizard

is available as a plugin for Eclipse and uses the Graphical Editing Framework.[9] A web-based version is also planned in the near future.

Additional information and installation instructions for GUIPOT, XD Transformation Wizard and TPE are at `http://owl.vse.cz:8080/tools.html`.

4 Demo Summary

The demo[10] will feature several variants of *wizard-based transformation* in NeOn wrt. two best-practice content patterns (for role-based modelling and for reified participation). Other TPs can be applied over *GUIPOT* or via *RESTful services* with simple HTML interface. *TPE* can be shown in the design of a new TP or modification of an existing one. Finally, a dedicated HTML interface to *complexity-downgrading*, i.e. a reasoner-targetted setting of the transformation (not described in the paper due to space limitations), can also be demoed.

Acknowledgement. The research has been partially supported by the CSF grant no. P202/10/1825. We thank Enrico Daga for his assistance in including PatOMat into XDTools.

References

1. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
2. Presutti, V., Gangemi, A.: Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 128–141. Springer, Heidelberg (2008)
3. Šváb-Zamazal, O., Daga, E., Dudáš, M., Svátek, V.: Tools for Pattern-Based Transformation of OWL Ontologies. Presented as Demo at ISWC 2011, Bonn (2011)
4. Šváb-Zamazal, O., Svátek, V., Iannone, L.: Pattern-Based Ontology Transformation Service Exploiting OPPL and OWL-API. In: Cimiano, P., Pinto, H.S. (eds.) EKAW 2010. LNCS, vol. 6317, pp. 105–119. Springer, Heidelberg (2010)

[9] `http://www.eclipse.org/gef/`

[10] This demo paper is successor of [3]. The main enhancements since then are the GUIPOT tool and the NeOn version of the wizard.

Guided Semantic Annotation of Comic Panels with Sewelis

Alice Hermann[1], Sébastien Ferré[2], and Mireille Ducassé[1]

[1] IRISA, INSA Rennes
[2] IRISA, Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
{hermanal,ferre,ducasse}@irisa.fr

Abstract. UTILIS (Updating Through Interaction in Logical Information Systems), introduced in a research paper at EKAW'12, is an interactive process to help users create new objects in a RDF graph. While creating a new object, relaxation rules are applied to its current description to find similar objects, whose properties serve as suggestions to expand the description. UTILIS is implemented in Sewelis, a system that reconciles the expressiveness of querying languages (e.g., SPARQL), and the benefits of exploratory search found in faceted search. The same interaction principles are used for both exploration and creation of semantic data. We illustrate the UTILIS approach by applying Sewelis to the semantic annotation of comic panels, reusing the dataset that was used for a user evaluation.

1 Motivation

As discussed in the related EKAW'12 research paper [2], authoring Semantic Web (SW) data is crucial to take into account information that cannot be extracted automatically from existing documents (e.g., multimedia data, subjective judgements). This is, however, tedious and in practice data from the SW are rarely directly produced by users. In the Web 2.0, users, nevertheless, significantly contribute to the production of data, thus, motivation is not a problem. Models exist to bridge the gap between the SW and the Web 2.0, for example by linking tags created by users with SW vocabulary [3]. There are, however, still difficulties to integrate the Web 2.0 data in SW, for example to automatically align users tags and resources of the SW. Moreover, SW data is richer than user tags. SW indeed allows a more complex representation of data, as well as more elaborate queries. It is therefore important to allow users to directly create data in a SW format.

There exists a number of editors that can guide users in the production of semantic data. They can be dedicated tools (e.g., Protégé, OKM, Gino, QuiKey) or integrated into Wiki environments (e.g., SMW (Semantic Media Wiki), KiWI, ACEWiki). Their interface can be based on forms (e.g., Protégé, OKM, SMW), on natural language (e.g., Gino, ACEWiki), or only allow for the creation of one triple (or semantic link) at a time (e.g., KiWI, Quikey). Those editors have

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 430–433, 2012.

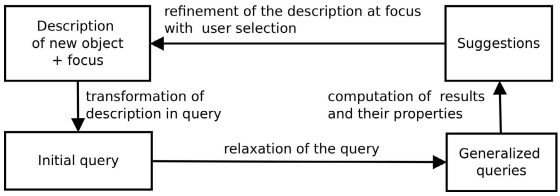


Fig. 1. Interactive refinement of the description of a new object

a number of limits regarding their applicability or their guiding. First, many editors require the definition of a schema or an ontology to make any suggestion (e.g., defining domains and ranges of properties). However, defining an ontology may be even harder than describing instances. Second, most editors only use the ontology axioms, not the facts, to compute suggestions. However, existing instances could serve as models for new instances, possibly in a more precise way than ontology axioms. Other editors hardly use known information about the new instance to compute suggestions. Third, most editors require users to enter at least a few letters to give suggestions, because their suggestions are not fine-tuned to the new instance.

The main contribution of our approach, UTILIS (Updating Through Interaction with Logical Information Systems), is to provide an interaction process that helps users to consistently create new objects by suggesting properties and values finely adapted to the very object being created. The already known properties of a new object are used to suggest others. Let us assume that a user annotates a new comic panel, and has already specified its collection. It is likely that this panel and those of the same collection have characters in common, which should then be suggested before other characters. The process uses a set of relaxation rules, inspired by the work of Hurtado et al. [9], and an efficient algorithm for computing suggestions. An advantage of our approach is that the definition of an ontology is not necessary to compute the suggestions, although UTILIS may use ontology axioms to improve its suggestions when they are available.

2 UTILIS : An Interactive Guidance

UTILIS searches for objects similar to the description of a new object, i.e., objects having common properties and values. Figure 1 illustrates the interactive process to refine the description of a new object. The initial query is obtained from the current description of a new object and from a focus which is an element of the description that the user wants to complete. This description is transformed into a query. That query is used as a starting point to obtain generalized queries using relaxation rules. The initial and generalized queries allow similar objects to be retrieved. Those objects and their properties are used as suggestions to complete the new description. After refinement, the new description will be used to define a new query and so on. For more information on the approach, and a user evaluation, see [2].

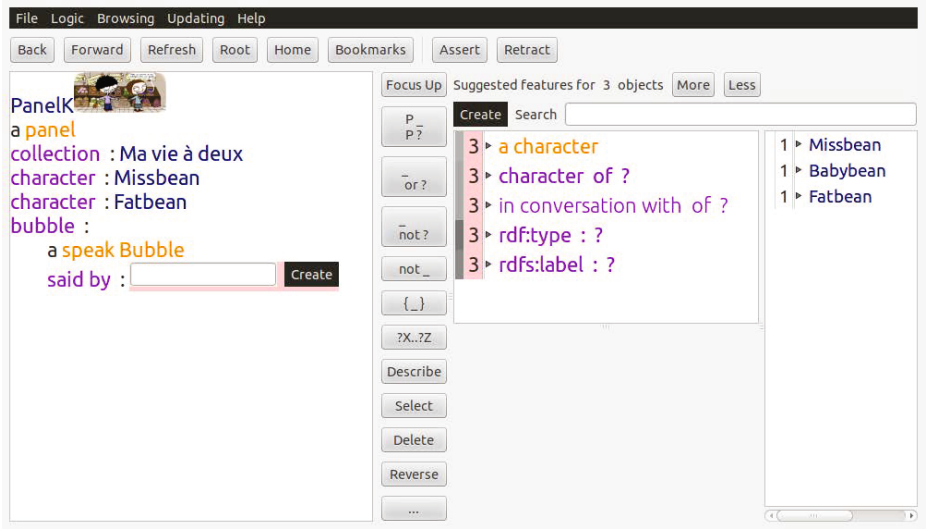


Fig. 2. Screenshot of Sewelis. On the left side, the description of the new object is shown, here panel K is a panel from the *Ma vie à deux* Collection, with *Missbean* and *Fatbean* as characters, it has a speech bubble said by someone. On the right side, suggested resources for the focus are listed, here the speakers.

3 Implementation in Sewelis

UTILIS is implemented in Sewelis[1]. Sewelis has been designed for the exploration of a RDF base, and is based on Query-based Faceted Search (QFS) [1]. It guides users in the construction of complex queries by suggesting query transformations that play the role of navigation links. The guidance avoids that users fall in dead-ends (i.e., empty results), and offers an expressiveness close to SPARQL.

When Sewelis is used to edit of a RDF base, the current query becomes the current description of the new object, and suggested transformations are computed according to the UTILIS approach, as presented in the previous section. To present suggestions to the user and allow him to complete the description of the new object, the interaction mechanisms of Sewelis, initially dedicated to navigation, have been reused to support creation. By default, only the suggestions at the smallest distance are displayed, then the list can be enlarged to bigger distances upon user's request, until all objects compatible with the searched element are proposed. At any time, users can manually enter what they want to add. That is necessary for the new values. An alternative way to find existing values is through auto-completion.

Figure 2 shows the user interface. The current description is on the left side, here panel K is a panel from the *Ma vie à deux* Collection, with *Missbean* and *Fatbean* as characters, it has a speech bubble said by someone. On the right

[1] http://www.irisa.fr/LIS/softwares/sewelis

side, suggested resources for the focus are listed, here the speakers. Suggested classes and properties are in the middle part. Above that area, the number of similar objects is displayed and a *More* button allows them to be expanded to larger distances. From there on, the user can continue the description of the panel and when he decides that it is complete, he adds it to the base with the *Assert* button.

4 Demonstration

The demonstration develops an edition scenario of a comics knowledge base that is initially only partially annotated. It consists of 362 individuals, including 89 panels, divided into 16 classes and connected by 20 properties. The base describes a subset of the panels by their characters, bubbles, places, and other elements; it describes bubbles by their speakers, recipients, and text. In the user evaluation we organized, the subjects had to complete the knowledge base by annotating the not-yet-described panels along the lines of the already-described panels. The demonstration will consist in re-playing this task, and participants will be able to try UTILIS. The various situations that can occur will be illustrated:

- a description element (i.e., a class, a property or a value) is found in the set of suggestions,
- a description element is found by performing a few enlargments of the set of suggestions,
- a description element is found through auto-completion,
- a new value is created,
- a new class is created,
- a new property is created.

The UTILIS suggestions favor the reuse of existing elements upon the creation of new ones, which favors the consistency of the knowledge base by avoiding the creation of duplicates. It also favors the completeness of the base by encouraging users to fill all properties used for previous objects.

References

1. Ferré, S., Hermann, A.: Semantic Search: Reconciling Expressive Querying and Exploratory Search. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 177–192. Springer, Heidelberg (2011)
2. Hermann, A., Ferré, S., Ducassé, M.: An Interactive Guidance Process Supporting Consistent Updates of RDFS Graphs. In: ten Teije, A., et al. (eds.) EKAW 2012. LNCS (LNAI), vol. 7603, pp. 185–199. Springer, Heidelberg (2012)
3. Passant, A., Laublet, P.: Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. In: Workshop Linked Data on the Web (LDOW). CEUR-WS (2008)

I-CAW: Intelligent Data Browser for Informal Learning Using Semantic Nudges

Dhavalkumar Thakker, Vania Dimitrova, and Lydia Lau

University of Leeds, Leeds LS2 9JT, UK
{D.Thakker,V.G.Dimitrova,L.M.S.Lau}@leeds.ac.uk

Abstract. This demonstration will present a system, called I-CAW, which aggregates content from social spaces into a semantic-enriched data browser to promote informal learning. The work pioneers a new way to interact with social content using nudges (in the form of signposts and prompts) that exploit ontologies and semantically augmented content. The results of a user study with I-CAW suggest that semantic nudges are a fruitful way to enhance the interaction in semantic browsers in order to facilitate learning. The demonstration will offer hands-on experience with I-CAW following the settings from the user study.

Keywords: Semantic Data Browser, Social Semantic Web, Semantic Augmentation, Application of Knowledge Management for Informal Learning.

1 Introduction

The dramatic rise of social media leads to the development of a vast amount of user-generated content(UGC) which is radically transforming today's practice in many areas (e.g. policy making, disaster response, open government). Trends and predictions (e.g. [1]) point out that social media will have a strong impact on learning in workplace, providing a huge resource of user-generated content for learning that may be unplanned and driven by circumstances – i.e. informal learning. Social spaces can offer a plethora of digital traces (DTs)[1] of real world experiences: people write reviews about their experiences with staff or services (e.g. in hotels); share their personal stories (e.g. in blogs); leave comments pointing at situations they have experienced (e.g. when watching videos). If selected carefully, these DTs can give broad, authentic and up-to-date digital examples of job activities, and can be a useful source for informal learning. DTs can be particularly appealing as a source for informal learning of soft skills (e.g. communicating, planning, managing, advising, negotiating), highly demanded in the 21st century [2].

We propose a novel approach where UGC is used as a source for learning situations linked to real world experience. For instance, learners can browse through examples with different job situations, share personal stories, read stories and comment, become aware of different perspectives, triggering self-reflection and goal-setting for personal development. This demonstration will present an intelligent content assembly workbench (I-CAW) showing how a semantic data browser can be extended

[1] The term "digital traces" is confined to traces of real world experiences from social media.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 434–437, 2012.

with **semantically-driven 'nudges'** that preserve the freedom for users to explore social spaces with UGC, yet providing guidance to benefit from informal learning.

2 Brief Description of I-CAW

I-CAW is a semantic data browser for learning which combines key semantic technologies - semantic augmentation, semantic query, relatedness, similarity, entity summarisation, and ontology summarisation. Fig. 1 gives the main components.

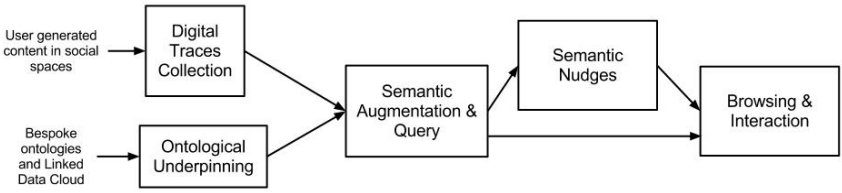


Fig. 1. The main components of the Intelligent Content Assembly Workbench(I-CAW)

Digital Traces Collection. I-CAW supports users to browse user generated content including videos and comments from YouTube (linked within I-CAW) and personal stories. The digital traces are collected from the open social spaces (e.g. YouTube) and the closed social spaces (such as blog-like story telling environment).

Ontological Underpinning. An Activity Model Ontology (AMOn)[2] is developed by a multi-disciplinary team of computer scientists and social scientists [3]. The ontological underpinning for aggregation of DTs in I-CAW utilises AMOn, DBPedia and public ontologies. The ontologies are used by intelligent services for semantic augmentation, query and for designing semantic nudges as described below.

Semantic Augmentation and Query[3]**.** Unstructured or semi-structured user generated content is semantically tagged with ontology concepts, which is implemented using the General Architecture for Text Engineering (GATE). Semantic indexing is developed using semantic repositories converting the annotation sets to RDF triples. The mechanism for querying and browsing the semantically augmented content takes a focus concept (C_f) and outputs other concepts and content using a relatedness algorithm (deriving concepts linked via properties with C_f).

Semantic Nudges. I-CAW proactively suggests areas of exploration to learners in the form of *nudges* based on Sunstein and Thaler's choice architecture [4]. A nudge is any aspect of the choice architecture that alters people's behaviour in a predictable way without forbidding any options, and tries to influence choices in a way that will make choosers better off. In our analysis for mapping the choice architecture to semantic technologies, we narrow down to two elements of the choice architectures that are possible to be implemented using semantic technologies. These two elements are mapped to *signposting* and *prompts* and are the 'semantically-driven nudges' in I-CAW. **Signposting** are related to "default options" which usually most users will

[2] http://imash.leeds.ac.uk/ontology/amon/
[3] Details on services : http://imash.leeds.ac.uk/imreal/wp3/#services

end up with. The semantic data browsers generally have information on a focus concept with a list of known facts from ontological knowledge bases. The presentation (signposting) of these facts influences the navigational path that learners take and facts they can read. "All Facts" show all the facts about a concept from the ontologies. "Key Facts" give a summary of a focus concept with fewer facts, and yet containing sufficient information for learners to quickly identify the concept. This provides immediate exploration space, generated using entity summarisation techniques [5]. "Overview" of the ontological knowledge base provides overall exploration space and is implemented using ontology summarisation techniques [6]. **Prompts** provide non-invasive suggestions based on similar and/or contradictory learning objects. Similarity prompts use the subsumption relations, e.g. if the user explores content on 'nervousness', which is a negative emotion, a similarity prompt directs to content linked to other negative emotion like 'anxiousness'. Contradictory prompts are based on finding mutually disjoint concepts, e.g. if the user explores content on 'aggression', they can be suggested 'empathy' as positive and negative emotions are disjoint.

Browsing and Interaction. I-CAW provides interface that incorporate the services and nudges and allows learners to search and browse relevant digital traces for interpersonal communications. *The demonstration*[4] *will offer hands on experience with I-CAW (an example screen shot is given in Fig. 2).*

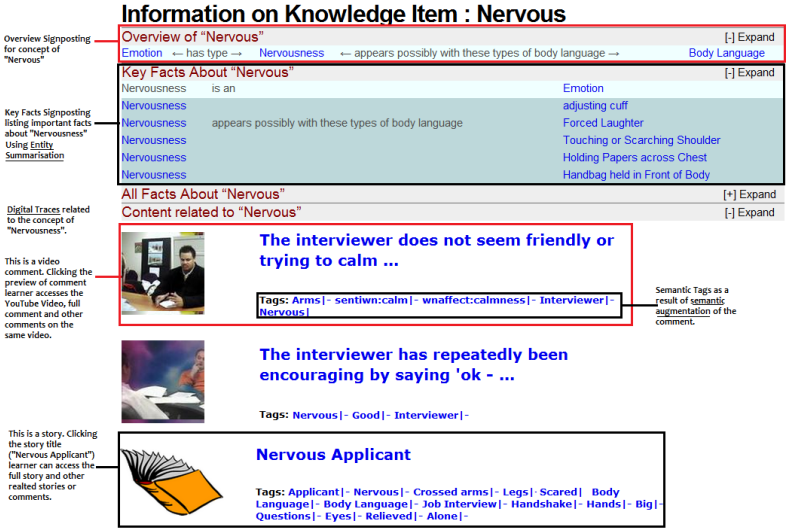


Fig. 2. Sample interaction screen - exploring the concept 'nervous' using signposting nudges

3 Key Contribution

I-CAW illustrates an original knowledge management approach that exploits semantic web technologies to meet the demands of an important domain – informal learning. Semantic data browsers that combine semantically augmented data and ontological

[4] Demo screen cast available at: `http://vimeo.com/user8051533/icawdemo`

knowledge bases are being utilised in various domains (a recent review is given in [7]). Data browsers assume that the users are in charge of what they do when using the browser. This puts the cognitive onus on the user, and is particularly acute in the case of a user being a learner, i.e. not familiar with the conceptual space in the domain and may be unable to decide what is the best course of action for him/her. Hence, directly adopting semantic web browsers in learning contexts would not be sufficient for effective learning environments – new intelligent techniques are needed to extend these browsers with features that facilitate informal learning. I-CAW presents a novel approach to extend semantic browsers with nudges in order to influence the choices users make and benefit learning.

An experimental study was conducted with 10 users who used I-CAW to browse through digital traces related to job interview examples (focusing on body language and emotion). The study, presented in [8], confirmed that the use of *nudges* is a step in the right direction for turning a semantic browsing experience into an informal learning experience. Overall, the signposting *nudges* were considered a fruitful way to provide a quick summary for understanding a concept and for exploration which leads to something new to learn. The prompts were seen as task setting for the learners to browse further to understand the bigger picture. Additional strategies for signposting and prompts were suggested, e.g. adding 'complementary' prompts to suggest complementary content or 'reflection' prompts to ask learners to link their previous experiences to content they have seen.

Using social content brings in new sources for learning, e.g. the diverse range of real-world experiences. I-CAW demonstrates a key step in this direction, which informs further research to exploit the reuse of social content for learning.

Acknowledgements. The research leading to these results has received EU funding (FP7/2007-2013) under grant agreement no ICT 257831 (ImREAL project).

References

1. Redecker, C., Ala-Mutka, K., Punie, Y.: Learning 2.0 - The Impact of Social Media on Learning in Europe, Policy Brief, European Commission, Joint Research Centre (2010)
2. Ananiadou, K., Claro, M.: 21st Century Skills and Competences for New Millennium Learners in OECD Countries. OECD Education Working Papers, No. 41 (2009)
3. Thakker, D., Dimitrova, V., Lau, L., Karanasios, S., Yang-Turner, F.: A Priori Ontology Modularisation in Ill-defined Domains. In: I-Semantics 2011, pp. 167–170 (2011)
4. Sunstein, C., Thaler, R.: Nudge: Improving Decisions about Health, Wealth, and Happiness. Penguin Books, New York (2009)
5. Cheng, G., Tran, T., Qu, Y.: RELIN: Relatedness and Informativeness-Based Centrality for Entity Summarization. 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. 114–129. Springer, Heidelberg (2011)
6. Zhang, X., Cheng, G., Ge, W., Qsssu, Y.: Summarizing Vocabularies in the Global Semantic Web. Journal of Computer Science and Technology 24(1), 165–174 (2009)
7. Popov, I.O., Schraefel, M.C., Hall, W., Shadbolt, N.: Connecting the Dots: A Multi-pivot Approach to Data Exploration. 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. 553–568. Springer, Heidelberg (2011)
8. Thakker, D., Desptakis, D., Dimitrova, V., Lau, L., Brna, P.: Taming Digital Traces for Informal Learning: A Semantic-driven Approach. In: ECTEL (to appear, 2012)

RightField: Scientific Knowledge Acquisition by Stealth through Ontology-Enabled Spreadsheets

Katy Wolstencroft[1], Stuart Owen[1], Matthew Horridge[3], Wolfgang Mueller[2], Finn Bacall[1], Jacky Snoep[4], Franco du Preez[4], Quyen Nguyen[2], Olga Krebs[2], and Carole Goble[1]

[1] School of Computer Science, University of Manchester, Manchester, M13 9PL, UK
[2] HITS gGmbH, Schloss-Wolfsbrunnenweg 35, Heidelberg, Germany
[3] Bio-Medical Informatics Research Group, Stanford University, CA 94305, USA
[4] Manchester Institute for Biotechnology, University of Manchester, M1 7DN
given.family@manchester.ac.uk, given.family@h-its.org, matthew.horridge@stanford.edu

Abstract. RightField is a Java application that provides a mechanism for embedding ontology annotation support for scientific data in Microsoft Excel or Open Office spreadsheets. The result is semantic annotation by stealth, with an annotation process that is less error-prone, more efficient, and more consistent with community standards. By automatically generating RDF statements for each cell a rich, Linked Data querying environment allows scientists to search their data and other Linked Data resources interchangeably, and caters for queries across heterogeneous spreadsheets. RightField has been developed for Systems Biologists but has since adopted more widely. It is open source (BSD license) and freely available from http://www.rightfield.org.uk.

Keywords: ontology annotation, biology, metadata standards, spreadsheets, RDF, linked data, e-science.

1 Introduction

Scientific data search, integration and interpretation rely on accurate and uniform metadata annotation. Consequently many scientific fields have embarked on public initiatives to produce guidelines and associated controlled vocabularies for the collection of the metadata needed for the interpretation and reuse of data. The BioSharing initiative for biology (http://www.biosharing.org) lists over 150 Minimum Information Models (checklists/guidelines) and over 250 ontologies/controlled vocabularies. Minimum information models include MIAME for Microarray Experiment data. Controlled vocabularies include the Gene Ontology for gene products and CHEBI (Chemical Entities of Biological interest) for metabolites and other small molecules. Similar checklists and vocabularies are found in other scientific disciplines.

However, it is a well-recognized concern that although rich metadata annotation is necessary to compare and reuse scientific data there are many barriers to its acquisition: (a) the labour cost is high, being a time-consuming yet undervalued process that can only be partially automated; (b) the Minimum Information Models are merely

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 438–441, 2012.

checklists with many customizations and variants. Tools must cope with heterogeneous data and yet conform as much as possible to community standards and vocabularies; and (c) experimentalists are reluctant to adopt heavyweight tools and procedures, and have little experience of metadata management, ontologies or standardization.

One of the most popular and familiar tools for scientific data capture (and subsequent processing) is the spreadsheet. Microsoft Excel and OpenOffice are key tools for experimental scientists, and Google Refine is gaining popularity for data cleaning. RightField (http://www.rightfield.org.uk) is a lightweight tool for lowering the barrier of manual metadata acquisition by instrumenting spreadsheets. We take advantage of the popularity of spreadsheets to gather ontology-based annotations on experimental data "by stealth". In addition, RDF Linked Data generated from the spreadsheets provides a platform for cross spreadsheet integration and querying.

2 RightField in Three Steps

RightField is an open source (BSD license), freely available Java application that provides a mechanism for embedding ontology annotation support for data in Excel or Open Office spreadsheets. It does not require special macros, visual basic code or platform specific libraries to run it. It enables users to upload Excel spreadsheets along with ontologies from their local file systems, or from the BioPortal repository [1] (http://bioportal.bioontology.org). The tool supports OWL, OBO and RDFS ontologies and RDF vocabularies. Support for SKOS is planned.

Step 1: Definition. We use RightField to define MS Excel or OpenOffice templates for a Minimum Information Model for an experiment. Individual cells, columns, or rows can be restricted to display particular ranges of allowed classes or instances from chosen ontologies (Figure 1). Each spreadsheet can be annotated with terms from multiple ontologies. Once marked-up and saved, the RightField-enabled spreadsheet contains embedded worksheets with information concerning the origins and versions of ontologies used in the annotation. As everything is embedded, scientists do not require any new applications and can complete annotation offline. This also makes the spreadsheets readily exchangeable and enables a series of experiments to be annotated with the same versions of the same ontologies even if the live ontologies change.

Step 2: Acquisition. The resulting RightField-enabled spreadsheet presents selected ontology terms to users as a simple drop-down list, enabling scientists to consistently annotate their data without requiring detailed knowledge of the ontology structures and content. By combining templates for Minimum Information Models and embedded ontology terms we provide an infrastructure that promotes and encourages compliance and standardization. Embedding annotation terms in the spreadsheets ensures that term selection occurs at the time of the experiment within the application already in use. This is ontology annotation by stealth. The experimentalists do not require specialist knowledge of the ontology resources used.

Step 3: Cross-Linking. As RightField supports the definition of ontology classes, instances and properties of each cell, we can automatically generate RDF statements for each cell and therefore an RDF graph, or collection of graphs, for each data set. Extracting this RDF to a triple-store and serving it as Linked Data provides a rich, querying environment that allows the scientists to search their data and other Linked Data resources interchangeably.

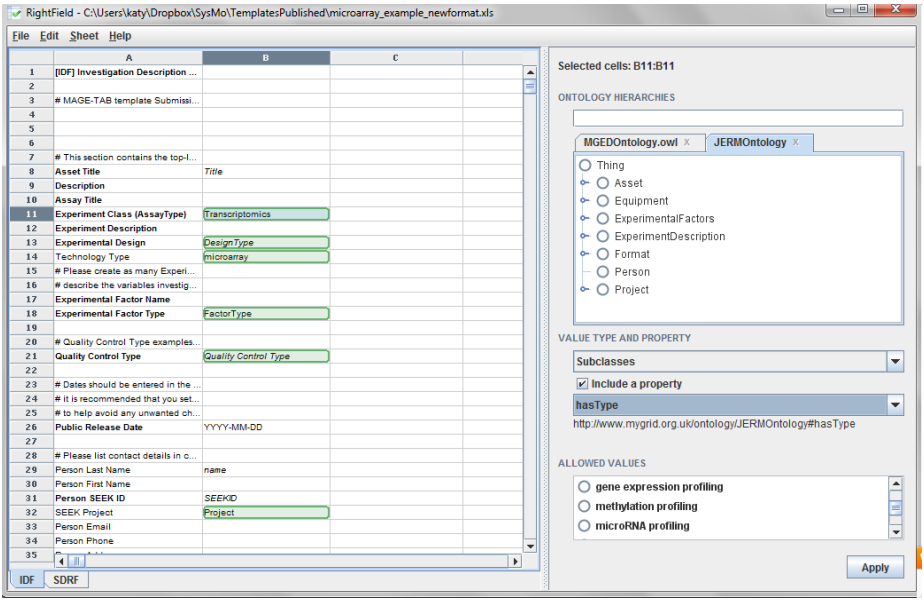


Fig. 1. Building an ontology-enabled template using BioPortal [1]

Established spreadsheet templates can be updated as ontologies are revised using RightField. When, why and how is up to the practice of each laboratory. Deliberately, no automated updating is implemented as each experiment or study has its own protocols for metadata refresh.

Since its release in October 2011 RightField has been adopted by a variety of scientific groups. We estimate the current user base to be around 300 active individuals. We give two examples of areas where scientists are using the tool:

Systems Biology: The SysMO (Systems Biology for MicroOrganisms) Initiative (http://www.sysmo.net) and the Virtual Liver Network (http://www.virtual-liver.de) together cover over 110 institutions, 20 consortiums and 350+ scientists. RightField-enabled SysMO templates are available from the tool website. Adoption has lead to a marked increase in the consistency of data annotation.

Disease Knowledge Bases: The KUPKB (Kidney and Urinary Pathway KnowledgeBase) project uses RightField to annotate datasets from high-throughput experiments on the kidney and the urinary system. The tool has lead to consistent and precise metadata which helps to integrate different experiments [2]. Work has started to build similar knowledge bases for inflammatory bowel disease and Chagas disease.

3 Related Work and Summary

RightField is a tool with a *light-touch* use of semantic web technologies. It is an application that addresses the whole workflow of data collection, annotation, RDF representation and querying. The novel part lies in disguising the use of semantics from the end users. Simplicity and unobtrusive embedding within a widely used application is its strength. ISA Creator (from http://isatab.sourceforge.net) is a bespoke spreadsheet tool

designed for experts, not end users, and is designed to work only with the ISA-TAB specification and associated vocabularies [3], whereas RightField can be configured to use any ontology and metadata schema. Phenote (http://phenote.org), and the PRIDE Proteome Harvest Spreadsheet submission tool (http://www.ebi.ac.uk/pride/ proteome-harvest/) are powerful annotation tools for specific biological disciplines, and are not generically applicable. Google Refine (http://code.google.com/p/google-refine) is a tool designed to help users deal with inconsistent data in spreadsheets. RightField is designed to improve the accuracy of data as it is collected. Both address the same problem of inconsistencies in data. Excel2RDF (http://www.mindswap.org/~rreck/excel2rdf.shtml), XLWrap [4] and RDF123 [5] extract spreadsheet data to RDF but focus on the transformation of spreadsheet content rather than the structure and consistency of that content. Relationships between spreadsheets cells cannot be captured. RightField templates allow the extraction of data to a particular metadata model, allowing the expression of complex relationships between cells and datasheets. The Anzo platform ((http://www.cambridgesemantics.com) is a commercial product with similar goals focusing on spreadsheet coordination and coherency through a common RDF bus.

Compliance with community ontologies means that we can already use Web Services from the BioPortal for term lookup and visualization of data collected using the approach. Extraction into RDF provides further means for searching across the content of spreadsheets and will allow scientific data to be provided as Linked Data. Technically we are tackling the faithful preservation of formats and the handling of large ontologies. The chief challenge is the institution of curator workflows for creating and maintaining the templates. The users of the RightField-enabled spreadsheets have reported favorably, responding well to having their metadata annotation choices restricted.

Acknowledgments. This work was funded by the BBSRC awards SysMO-DB BG0102181 and SysMO-DB2 BB/I004637/1.

References

1. Noy, N.F., et al.: BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37, W170–W173 (2009)
2. Jupp, S., Klein, J., et al.: Ontologies come of age with the iKUP browser. In: Proc. Ontologies Come of Age in the Semantic Web, ISWC 2011. CEUR Proc., vol. (809), pp. 25–28 (2011)
3. Sansone, S.-A., et al.: Toward interoperable bioscience data. Nature Genetics 44, 121–126 (2012)
4. 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)
5. 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)

TrustMe, I Got What You Mean!
A Trust-Based Semantic P2P Bookmarking System

Mustafa Al-Bakri[1], Manuel Atencia[1,2], and Marie-Christine Rousset[1]

[1] University of Grenoble, France
[2] INRIA Grenoble – Rhône-Alpes, France

1 Introduction

Virtual online communities (social networks, wikis...) are becoming the major usage of the web. The freedom they give to publish and access information is attracting many web users. However, this freedom is filling up the web with varied information and viewpoints. This raises important issues that concern privacy and trust. Due to their decentralised nature peer-to-peer (P2P) systems provide a partial solution for the privacy problem: each user (peer) can keep control on her own data by storing it locally and by deciding the access they want to give to other peers. We focus on *semantic* P2P systems in which peers annotate their resources (documents, videos, photos, services) using ontologies.

Indexing resources using the terms of an ontology enables more accurate information retrieval and query answering than indexing by keywords of a textual annotation. In particular, it is a way to overcome the problems raised by homonymy in classical keyword search engines. As an example, the word "Paris" corresponds to (at least) four meanings: a city, a film, a mythological figure, and a genus of flowering plants. Classical keyword search engines such as Google or Yahoo! returns to a keyword query "Paris" a mixture of web pages referring to its different meanings. In contrast, using as query interface an ontology in which a class named "Paris" is a subclass of a class named "Capitals" would remove the ambiguity on the interest of a user if she clicked on that particular class.

In semantic P2P systems, every peer is autonomous and free to organise her local resources as instances of classes of her *own ontology* serving as a query interface for other peers. Alignments between ontologies are required to reformulate queries from one local peer's vocabulary to another. Currently there exists a considerable amount of matchers available for computing alignments automatically. Nonetheless, automatic matchers may fail to compute sound and complete semantic alignments, and thus making query reformulation not fully reliable. Still a trust mechanism can help peers to find the most reliable sources of information within the network.

This demo builds on ATrust, a probabilistic model aimed to assist peers for query answering in semantic P2P systems [1]. Specifically, ATrust allows to estimate the probability that a peer will provide a satisfactory answer to a query posed by another peer. This probability is taken as the *trust* that the addressing peer has towards the addressed peer, and it is subject to the posed query. Unlike other existing approaches to trust, unsatisfactory answers are seen as the result of peers' incapacity to understand each other. Trust values are refined over time as

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 442–445, 2012.

more queries are sent and answers received in a Bayesian learning process in which alignments are used to build initial prior distributions. As a by-product ATrust provides the means to rank, in the scope of a particular class of a peer's ontology, the set of received instances according to their probability to be satisfactory. For this, a function aggregates the trust values of all the peers that returned these instances.

2 Goals of the Demo

TrustMe demonstrates the use of ATrust for guiding query answering in semantic P2P social communities. Our view is that semantic P2P social communities illustrate the evolution of the semantic web towards a social web. We have decided to reproduce this vision via a semantic P2P bookmarking network in which peers exchange URLs of articles (web pages) about the topics they are interested in. Unlike current bookmarking systems (e.g. Delicious), in this ideal P2P bookmarking network, information is no longer centralised, and peers need to query other peers to gather new articles. Each peer uses her own taxonomy of categories for tagging articles. These taxonomies can be seen as ontologies so that each category C in a peer P's taxonomy is a class whose instances are URLs of articles initially indexed by C or by a subcategory (subclass) of C in the taxonomy. Then the set of instances of C can be gradually enriched by adding instances of classes D that belong to the taxonomies of P's acquainted peers and that happen to be aligned with C.

In this demo we highlight the gain in the quality of peers' answers —measured with precision and recall—when the process of query answering is guided by ATrust. In particular, we show how trust overcomes the problem of homonymy, still a constant battle for ontology matchers. Further, a trust-based ranking of instances allows to distinguish those that are relevant to a class from those related to its homonymous classes.

3 Setting up the P2P Bookmarking Scenario

Below we explain the generation of the taxonomies we have used for the semantic P2P bookmarking scenario, the alignments between these taxonomies, and the topology of the network.

Taxonomy Generation. First, we selected a number of homonyms (e.g. "Paris" and "Rome") in different topics (e.g. cinema and cities). These homonyms are the names of some of the most specific classes (the leaves) of the peers' taxonomies. We built up the rest of the taxonomies manually by looking up for Wikipedia categories organised in super-category and sub-category relations. In total, 5 taxonomies were generated. For example, one of the taxonomies is used by peers interested in cinema and music, and contains a category "Paris" which corresponds to a film. Another taxonomy is used by peers interested in cities and music, and contains a category "Paris" corresponding to the capital of France.

Alignment Generation. For generating alignments, we chose four matchers that regularly participate in OAEI [2] and launched them with the peers' taxonomies. From the resulting alignments, we selected the ones with best F-measure values according to manually computed reference alignments.

The selected alignments happened to be both unsound and incomplete. In particular, matchers did not get over the homonymy of the taxonomies. For instance, the class Paris corresponding to a film was aligned with the class.

Network Generation. Social networks are well-known to exhibit small-world characteristics. In the demo we generate small-world topologies that contain up to 1000 peers. Each peer is randomly associated with one taxonomy to reproduce the fact that acquainted peers may or not share the same topics of interest.

Taxonomy Population. For each category/class of each taxonomy we first built a reference set made up of the URLs of all the category's associated articles in Wikipedia, and other related articles returned by Google that are manually checked to be relevant. Then, we populate each peer's "copy" of the category by selecting randomly an initial set that contains 10% of the reference URLs. URLs in this initial set are the only URLs that are known for the peer while the reference set is not known for the peer and it is used only for evaluating the results and to simulate the user feedback. In order to ensure that every leaf had 50 articles at least, we had to duplicate the web pages of the starting set (although they were identified with different URLs). This also guaranteed that each peer's taxonomy had at least 1000 articles.

4 TrustMe

In TrustMe we compare two scenarios: a *naive* scenario in which peers query all their acquaintances through query reformulation based on alignments, and a *trust-based* scenario in which peers compute trust values using ATrust and only accept answers from their trustworthy acquaintances. More specifically, if a peer P_i is interested in gathering new articles about a topic specified by the category C in her taxonomy at time t,

- in the naive scenario, if the correspondence $\langle C, D, = \rangle$ is in the alignment between P_i's and P_j's taxonomies, P_i will send the query $Q = D(X)?$ to P_j, and P_j's answer will be added to P_i's taxonomy, but
- in the trust-based scenario, P_j's answer will not be added unless the trust that P_i has towards P_j w.r.t the translation $\langle C, D \rangle$ at t, is greater than a given trust threshold.

To compare the two scenarios we use precision and recall. Moreover, we evaluate the probability of articles to be relevant to the categories they belong to. This provides the means for ranking articles within categories.

In TrustMe the user can set up the values of several parameters: number of peers, trust threshold and the size of the samples used when performing Bayesian learning in ATrust. Then, run the demo until the correspondences of all the alignments in the network are randomly selected a desired number of times. Figure 1 shows a snapshot of TrustMe when the user chose one peer from the network after running the demo. It shows the returned ranked results of querying the peer's neighbours about a category. More specifically, it presents the added articles to Paris, the capital of France, in the taxonomy of the chosen peer in both scenarios (trust based scenario's results are in the upper left while the

Table 1. Comparing Precision and recall averaged values for Paris homonyms

category/class	Trust Scenario Precision	Trust Scenario Recall	Naive Scenario Precision	Naive Scenario Recall
Paris (city)	88%	64%	48%	96%
Paris (film)	85%	80%	27%	100%
Paris (mythology)	100%	66%	14%	98%
Paris (plant)	80%	97%	16%	100%

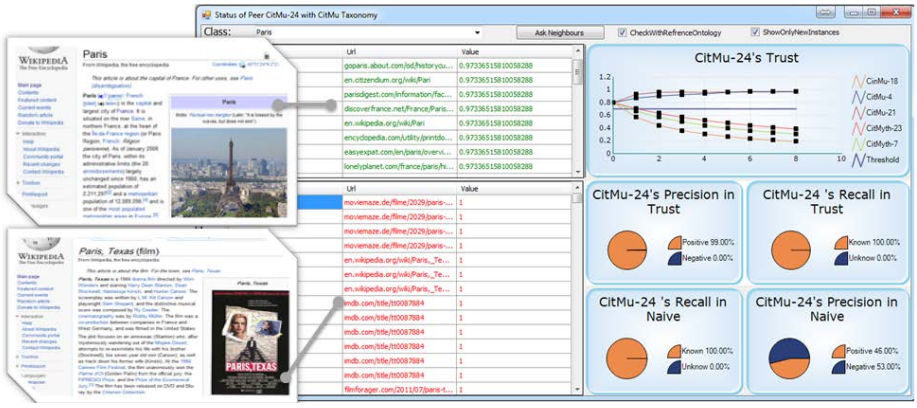


Fig. 1. Snapshot showing the added articles in Paris city class of one peer's taxonomy

naive scenario's results are in the lower left). We clicked on one of the trust based results and one of the naive ones. The former resulted in the Wikipedia article of Paris, and the latter in the Wikipedia article of the film "Paris". Also, the demo shows the trust values over time peer has towards her neighbours in the upper right graphics. The graphics shows 2 trustworthy neighbours against 3 untrustworthy ones. In the lower right of the screen the demo shows the precision and recall for that particular peer in both naive and trust-based scenario.

Table 1 presents the average values of precision and recall for classes about four meanings of Paris in the two scenarios. These means were computed over the set of all peers sharing each of the meanings. To compute them we used the values: number of peers = 100, threshold = 0.7, size of the samples=15%. We ran until the correspondences of all the alignments in the network were selected randomly 5 times. The results show that ATrust guarantees high precision. On the other hand, as one could expect, recall is higher in the naive scenario.

References

1. Atencia, M., Euzenat, J., Pirrò, G., Rousset, M.-C.: Alignment-Based Trust for Resource Finding in Semantic P2P Networks. 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. 51–66. Springer, Heidelberg (2011)
2. 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)

NIF Combinator: Combining NLP Tool Output

Sebastian Hellmann[1], Jens Lehmann[1], Sören Auer[2], and Marcus Nitzschke[1]

[1] Universität Leipzig, IFI/BIS/AKSW, D-04109 Leipzig, Germany
lastname@informatik.uni-leipzig.de
http://aksw.org
[2] Technische Universität Chemnitz, Informatik/ISST, D-09107 Chemnitz, Germany
soeren.auer@informatik.tu-chemnitz.de

Abstract. The NLP Interchange Format (NIF) is an RDF/OWL-based format that provides interoperability between Natural Language Processing (NLP) tools, language resources and annotations by allowing NLP tools to exchange annotations about text documents in RDF. Other than more centralized solutions such as UIMA and GATE, NIF enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. NIF wrappers have to be only created once for a particular tool and can subsequently interoperate with a potentially large number of other tools. We present (1) the currently implemented NIF wrappers, which are available as free web services and (2) a GUI called the NIF Combinator, which allows to combine the output of the implemented NIF web services.

1 Introduction

We are currently observing a plethora of *Natural Language Processing* (NLP) tools and services being available and new ones appearing almost on a weekly basis. Some examples of web services providing *Named Entity Recognition* (NER) are *Zemanta*, *OpenCalais*, *Ontos*, *Evri*, *Extractiv* and *Alchemy*. Similarly, there are tools and services for language detection, Part-Of-Speech (POS) tagging, text classification, morphological analysis, relationship extraction, sentiment analysis and many other NLP tasks. Each of the tools and services has its particular strengths and weaknesses, but exploiting the strengths and synergistically combining different tools is currently an extremely cumbersome and time consuming task, as the programming interfaces and result formats often differ to a great extent. Also, once a particular set of tools is integrated, this integration is usually *not reusable* by others. In order to simplify the combination of tools, improve their interoperability and facilitate the use of Linked Data, we developed the NLP Interchange Format (NIF). NIF is an RDF/OWL-based format that aims to achieve interoperability between *Natural Language Processing* (NLP) tools, language resources and annotations. The NIF specification has been released in an initial version 1.0 in November 2011 and implementations for 8 different NLP tools (e.g. UIMA, Gate ANNIE and DBpedia Spotlight) exist; a public web demo, the NIF Combinator, is available at http://nlp2rdf.lod2.eu/demo.php

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 446–449, 2012.

NIF-aware applications will produce output (and possibly also consume input) adhering to the NIF URI Scheme and the String Ontology as REST services (access layer). Other than more centralized solutions such as UIMA[1] and GATE[2], NIF enables the creation of heterogeneous, distributed and loosely coupled NLP applications, which use the Web as an integration platform. Another benefit is that a NIF wrapper has to be only created once for a particular tool, but enables the tool to interoperate with a potentially large number of other tools without additional adaptations. Ultimately, we envision an ecosystem of NLP tools and services to emerge using NIF for exchanging and integrating rich annotations. This paper describes the accompanying demo for [2].

NIF Example. NIF provides two URI schemes, which can be used to represent strings as RDF resources. In the example below, "Berlin" in the sentence "Berlin has a new mayor!" was annotated by two different tools, a part of speech tagger using an OLiA identifier and an entity linking tool connecting the string with DBpedia. In this case, a simple string offset based URI scheme was used [2].

```
@prefix : <http://prefix.given.by/theClient#> .
@prefix str: <http://nlp2rdf.lod2.eu/schema/str/> .
@prefix sso: <http://nlp2rdf.lod2.eu/schema/sso/> .
#the whole sentence is given a URI and typed as context
:offset_0_23  a     str:Context , sso:Sentence;
    str:isString    "Berlin has a new mayor!" .
#a substring is given a URI and is annotated
:offset_0_6   a     str:StringInContext , sso:Word ;
    str:referenceContext     :offset_0_23 ;
#part of speech annotation
    sso:oliaLink    <http://purl.org/olia/penn.owl#ProperNoun> ;
#link to dbpedia
    sso:oen                  dbpedia:Berlin .
```

2 NIF Wrappers and Combinator

Access via REST Services: The structural and conceptual interoperability layers of NIF are built upon the RDF standard, Linked Data principles and existing ontologies such as OLiA. To improve interoperability and accessibility of NIF components, NIF provides a normative *access layer*, which facilitates easier integration and off-the-shelf solutions by specifying REST parameters. Of special importance is the *prefix* parameter as it enables the client to influence the RDF output. The RDF in Figure 3 is produced by different tools, but can be merged directly under the condition that the URI prefixes and offsets are the same.

NIF can be used for import and export of data from and to NLP tools. Therefore, NIF enables to create ad-hoc **workflows** following a client-server model or the SOA principle. Following such an approach, clients are responsible for implementing the workflow. The NIF Combinator shows one possible implementation of such a workflow. The client sends requests to the different tools either as text or RDF and then receives responses in RDF. This RDF can be aggregated into a

[1] http://uima.apache.org/
[2] http://gate.ac.uk/

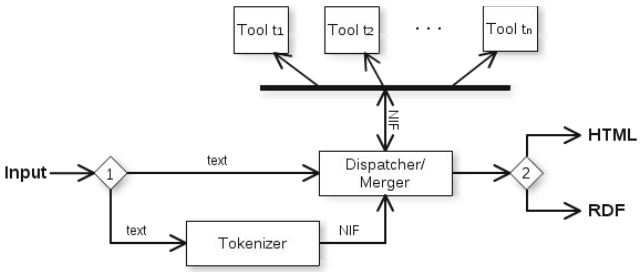


Fig. 1. Overview of the NIF Combinator workflow

local RDF model. Transparently, external data in RDF can also be requested and added without using additional formalisms. For acquiring and merging external data from knowledge bases, existing Semantic Web tools can be used.

The main **interface** are wrappers that provide NIF web services. A NIF web service must be *stateless*, *HTTP method agnostic respective POST and GET* and accept the following *parameters*:

- *Input type* (required, `input-type = text | nif-owl`). Determines the content required for the next parameter input, either plain text or RDF/XML.
- *Input* (required, `input = text | rdf/xml`). Either URL encoded text or URL encoded RDF/XML format in NIF.
- *Compatibility* (optional, `nif = true | nif-1.0`). Enables NIF output for existing web services (i.e. deploy NIF in parallel to legacy output).
- *Format* (optional, `format = rdfxml | ntriples | n3 | turtle | json`). The RDF serialisation format.
- *Prefix* (optional, `prefix = uriprefix`). An URL encoded prefix, which must be used for any URIs that the client will create. If missing, it should be substituted by a sensible default (e.g. the web service URI).
- *URI Scheme* (optional, `urirecipe = offset | context-hash`). The URI scheme that should be used (default is offset).

NIF Combinator Demo: Figure 1 describes the workflow of the NIF Combinator. The given input (normally text) can be forwarded directly to the dispatcher or optionally prepared by a tokenizer (see Diamond 1 in Figure 1). The tokenizer already outputs the results in NIF and provides tokenization for the remaining components. The dispatcher then calls the selected NLP tools (see checkboxes in Figure 2) which can read as well as write NIF. The NIF output from all the tools is then merged (see Figure 3). Merged results can be shown in HTML format for users. Another option (Diamond 2) is to output RDF directly. This way, the NIF Combinator can be used as an aggregator web service itself, by simply executing a GET/POST request with the parameters of the HTML forms.

Conclusions and Future Work: In this demo paper, we briefly presented NIF wrappers and the NIF Combinator tool. All implemented NIF wrappers are able to produce NIF output adhering to the NIF 1.0 specification. The additional

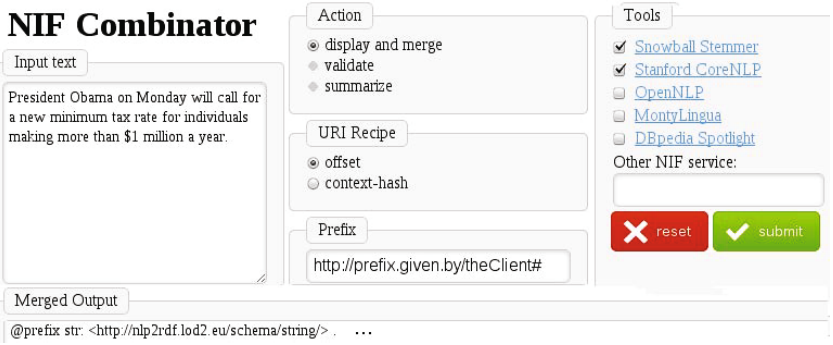


Fig. 2. Screenshot of the NIF Combinator user interface

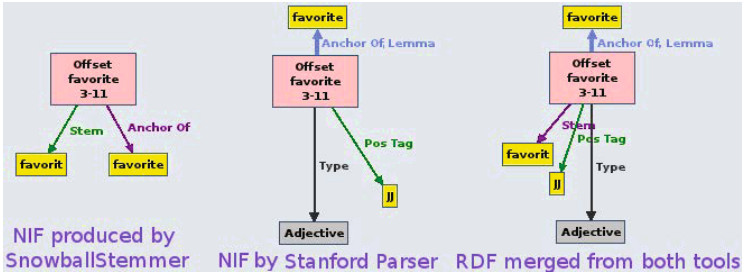


Fig. 3. Example of merged RDF from two NLP tools

integration of a tokenizer has, however, not yet been fully implemented for all wrappers. Tokenization conflicts will be resolved either by providing tokenization for the tools or by implementing resolution strategies for tokenization conflicts [1] in the dispatcher/merger component. Future work comprises the creation of a new version NIF 2.0 based on community feedback. All presented resources are openly available at `http://nlp2rdf.org`

Acknowledgments. We would like to thank our colleagues from AKSW research group and the LOD2 project for their helpful comments during the development of NIF. Especially, we would like to thank Christian Chiarcos for his support while using OLiA and the students that helped implement the wrappers: Markus Ackermann, Martin Brümmer, Didier Cherix, Robert Schulze. This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

1. Chiarcos, C., Ritz, J., Stede, M.: By all these lovely tokens.. merging conflicting tokenizations. LRE 46(1), 53–74 (2012)
2. Hellmann, S., Lehmann, J., Auer, S.: Linked-Data Aware URI Schemes for Referencing Text Fragments. In: ten Teije, A., et al. (eds.) EKAW 2012. LNCS (LNAI), vol. 7603, pp. 175–184. Springer, Heidelberg (2012)

Author Index

Abramowicz, Witold 282
Ahmed, Mohamed Ben 72
Al-Bakri, Mustafa 442
Aroyo, Lora 16
Atencia, Manuel 144, 442
Auer, Sören 175, 302, 353, 446

Bacall, Finn 438
Baumeister, Joachim 363
Bellahsene, Zohra 421
Blomqvist, Eva 216
Bryl, Volha 26, 130
Bucur, Anca 327
Bühmann, Lorenz 57

Campinas, Stéphane 200
Champin, Pierre-Antoine 317
Compton, Paul 97
Curé, Olivier 317

d'Amato, Claudia 26
d'Aquin, Mathieu 337
David, Jérôme 144
Davis, Brian 21
de Boer, Victor 16
Decker, Stefan 373
Delbru, Renaud 200
De Leenheer, Pieter 16
Demter, Jan 353
Di Francescomarino, Chiara 292
Dijkshoorn, Chris 16
Dimitrova, Vania 434
Dividino, Renata 154
Drăgan, Laura 373
Draicchio, Francesco 114
Ducassé, Mireille 185, 430
Dudáš, Marek 426
du Preez, Franco 438
Dzikowski, Jakub 282

Ermilov, Ivan 302

Ferré, Sébastien 42, 185, 430

Gangemi, Aldo 114
Gerber, Daniel 87
Ghezaiel, Lamia Ben 72
Ghidini, Chiara 292
Giuliano, Claudio 130
Goble, Carole 438
Gómez-Pérez, Asunción 267
Gröner, Gerd 154

Harland, Lee 1
Hatko, Reinhard 363
Hellmann, Sebastian 175, 446
Hermann, Alice 185, 430
Hildebrand, Michiel 16
Hogan, Aidan 164
Horridge, Matthew 438
Hyvönen, Eero 383

Joorabchi, Arash 32

Kaczmarek, Monika 282
Karnstedt, Marcel 164
Keet, C. Maria 237, 252
Khalili, Ali 302
Khan, Zubeida 237
Kim, Myung Hee 97
Kitamura, Yoshinobu 227
Kozaki, Kouji 227
Krebs, Olga 438

Latiri, Chiraz 72
Lau, Lydia 434
Łazaruk, Szymon 282
Lehmann, Jens 57, 175, 353, 446

Mahdi, Abdulhussain E. 32
Martin, Michael 353
Mersmann, Stefan 363
Milian, Krystyna 327
Mizoguchi, Riichiro 227
Motta, Enrico 337
Mueller, Wolfgang 438

Ngo, DuyHoa 421
Ngomo, Axel-Cyrille Ngonga 87

Nguyen, Quyen 438
Nikolov, Andriy 337
Nitzschke, Marcus 446

Owen, Stuart 438

Parreira, Josiane Xavier 164
Peroni, Silvio 398, 417
Pianta, Emanuele 413
Poveda-Villalón, María 267
Pradel, Camille 8
Presutti, Valentina 114, 216
Prié, Yannick 317
Puppe, Frank 363

Rospocher, Marco 292, 413
Rousset, Marie-Christine 442
Rudolph, Sebastian 42

Schädler, Dirk 363
Scharffe, François 144
Scheglmann, Stefan 154
Schneider, Jodi 21
Schreiber, Guus 16
Seil Sepour, Azam 216
Serafini, Luciano 26, 130, 413
Shotton, David 398, 417
Snoep, Jacky 438
Suárez-Figueroa, Mari Carmen 267
Suominen, Osma 383
Šváb-Zamazal, Ondřej 426
Svátek, Vojtěch 426

Tarasowa, Darya 302
Tesfa, Binyam 16
Thakker, Dhavalkumar 434
Thimm, Matthias 154
Thomas, Keerthi 337
Tokarchuk, Oksana 282
Tonelli, Sara 130, 413
Tummarello, Giovanni 200

Umbrich, Jürgen 164

van Harmelen, Frank 327
Vitali, Fabio 398, 417

Weiler, Norbert 363
Wolstencroft, Katy 438
Wyner, Adam 21