
Interacting with Linked Data (ILD 2012)

May 28, 2012, Heraklion, Greece

Workshop co-located with the
9th Extended Semantic Web Conference

Preface

While more and more semantic data is published on the Web, the question of how typical Web users can access this body of knowledge becomes of crucial importance. Therefore there is a growing amount of research on interaction paradigms that allow end users to profit from the expressive power of Semantic Web standards while at the same time hiding the complexity behind an intuitive and easy-to-use interface. These paradigms range from keyword search, faceted browsing and exploration to natural language question answering.

Keyword-based semantic search (e.g. provided by Swoogle¹ and Watson²) suffice for simple information lookup. However, keyword queries constitute an ambiguous and impoverished representation of an information need and do not fully exploit the expressive power of Semantic Web datamodels and query languages. *Explicit formal queries*, on the other hand, allow the user to exploit the full power and expressiveness of the Semantic Web standards. But they require schema knowledge and query language expertise, thus are not suitable for naive users. *Faceted browsing interfaces* (e.g. Ontogator [16], mSpace [17], and BrowseRDF [14]), in contrast, do not require domain knowledge or expertise in Semantic Web languages, rather they allow users to navigate through the dataset and thereby explore its content. However, faceted browsing is often domain-dependent. Also, in case the user has a clear information need, it is tedious to search for the answer manually, not knowing where exactly to look for it. Similarly, *graph visualization* (e.g. IsaViz³, [13]) provides an intuitive access to understanding semantic data, but often does not scale to large datasets and is not suitable for searching for an answer to a particular question. In order to precisely answer particular questions, a lot of research has been conducted on *natural language interfaces* (see for instance [4], [5, 6], [7], and [3]), which allow users to express arbitrarily complex information needs in an intuitive fashion. They combine progress in the areas of question answering from textual data [8] and natural language interfaces to databases [9]. The key challenges lie in translating these information needs into a form such that they can be evaluated using standard Semantic Web query processing and inferencing techniques, as well as in scaling question answering approaches to Linked Data.

Despite different goals and different kinds of interaction, the main challenge involved in interacting with Linked Data is the same for all approaches: dealing with a heterogeneous, distributed and very large set of highly interconnected data. The availability of such an amount of open and structured data has no precedents in computer science and approaches that can deal with the specific character of Linked Data are urgently needed.

¹ <http://swoogle.umbc.edu/>

² <http://kmi-web05.open.ac.uk/WatsonWUI/>

³ <http://www.w3.org/2001/11/IsaViz/>

The Workshop *Interacting with Linked Data* (ILD) is the second in a series of workshops exploring approaches towards a powerful and intuitive interaction with Linked Data. While the first workshop⁴ focused on question answering, the scope of ILD 2012 is now broader, including other paradigms for interacting with Linked Data. The goal of ILD is to bring together research and expertise from different communities, including NLP, HCI, Semantic Web and Databases, and to encourage communication across interaction paradigms. To this end, we issued a call for papers on the following topics:

- Question answering and natural language interfaces to Linked Data
- HCI and Linked Data
- Faceted browsing and exploration
- New interaction metaphors for Linked Data
- Multimodal interfaces to Linked Data
- Disambiguation and inferencing across multiple sources and domains
- Natural language generation
- Discovery on the fly of relevant Linked Data sources
- Efficiency and performance aspects
- Dealing with data and schema heterogeneity
- Summarization and aggregation
- Providing justifications of answers and conveying trust
- Personalization in accessing Linked Data
- User feedback and interaction
- Habitability and usability aspects

We received 11 submissions; to each of them three reviewers were assigned. On the basis of their reviews, six full papers were accepted for presentations. In addition, three papers were included as demo presentations.

Accompanying the workshop, we set up the second open challenge on Question Answering over Linked Data (QALD-2). It follows the first challenge (QALD-1) in aiming at facilitating the comparison between different question answering approaches and systems, and at developing the datasets needed for a standard evaluation benchmark for semantic question answering systems that focus on the ability to solve open-ended real life problems over real-world datasets. As part of this challenge, two independent datasets have been provided: DBpedia 3.7 and an updated RDF export of MusicBrainz – together with 100 training questions for each dataset, annotated with SPARQL queries and corresponding answers, and 100 test questions for DBpedia as well as 55 test questions for MusicBrainz. All questions were designed to represent information needs of different complexity that real end users would ask. The datasets as well as all training and test questions can be accessed from the workshop website:

<http://www.sc.cit-ec.uni-bielefeld.de/ild>

⁴ 1st Workshop on Question Answering Over Linked Data (QALD-1) at ESWC 2011:
<http://www.sc.cit-ec.uni-bielefeld.de/qald>

The main goal of the challenge was to get a picture of the strengths, capabilities and current shortcomings of question answering systems, as well as to gain insight into how question answering approaches can deal with the fact that the amount of RDF data available on the Web is huge and that this data is distributed and heterogeneous with respect to the vocabularies and schemas used. Five question answering systems participated in the test phase of the challenge, all reporting on the DBpedia question set (where Alexandria used German translations of the questions and extracted the answers from Freebase, thus suffering from data mismatches). The results of our online evaluation are given in Table 1. Alexandria, SemSeK, and QAKiS are described in papers in these proceedings; MHE was developed by Marek Ciglan⁵ at the Institute of Informatics at the Slovak Academy of Sciences.

	total	answered	right	partially	right	precision	recall	f-measure
SemSeK	100	80	32	7		0.44	0.48	0.46
Alexandria	100	25	5	10		0.43	0.46	0.45
MHE	100	97	30	12		0.36	0.4	0.38
QAKiS	100	35	11	4		0.39	0.37	0.38

Table 1. QALD-2 results on DBpedia

With the ILD workshop and the accompanying open challenge we hope to establish a series of workshops that aim at coupling current research on interaction paradigms for accessing Linked Data with open challenges that benchmark question answering approaches and thereby evaluate their success in providing an easy and intuitive interface to the Semantic Web.

May 2012

Christina Unger
 Philipp Cimiano
 Vanessa Lopez
 Enrico Motta
 Paul Buitelaar
 Richard Cyganiak
 (Organizing Committee ILD 2012)

⁵ <http://ups.savba.sk/~marek/>

References

1. Kaufmann, E., Bernstein, A.: How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users?. In: Proceedings of the Joint International and Asian Semantic Web Conference (ISWC/ASWC), pp. 281–294, 2007
2. Kaljurand, K.: ACE View – An Ontology and Rule Editor based on Controlled English. In: Proceedings of the International Semantic Web Conference (ISWC), Poster and Demo Proceedings, 2008
3. Cimiano, P., Haase, P., Heizmann, J., Mantel, M., Studer, R.: Towards portable natural language interfaces to knowledge bases – The case of the ORAKEL system. *Data Knowl. Eng.* 65:2, pp. 325–354, 2008
4. Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C.: Querying Ontologies: A Controlled English Interface for End-Users. In: Proceedings of the International Semantic Web Conference (ISWC), pp. 112–126, 2005
5. Lopez, V., Nikolov, A., Sabou, M., Uren, V., Motta, E., D’Aquin, M.: Scaling up Question-Answering to Linked Data. In: Proceedings of the 17th International Conference on Knowledge Management and Knowledge Engineering (EKAW), 2010
6. Lopez, V., Uren, V., Sabou, M., Motta, E.: Cross ontology query answering on the semantic web: an initial evaluation. In: Proceedings of International Conference on Knowledge Capture (K-CAP), pp. 17–24, 2009
7. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In: Proceedings of the International Conference on Data Engineering (ICDE), pp. 405–416, 2009
8. Strzalkowski, T., Harabagiu, S.: *Advances in Open Domain Question Answering*. Springer, 2006
9. Minock, M.: C-Phrase: A system for building robust natural language interfaces to databases. *Data Knowl. Eng.* 69:3, pp. 290–302, 2010
10. Damjanovic, D., Agatonovic, M., Cunningham, H.: Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction. In: Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010), Heraklion, Greece, May 31–June 3, 2010, Springer Verlag, 2010
11. Lopez, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: Proceedings of the 3rd European Semantic Web Conference, Budva, Montenegro, 2006
12. Lopez, V., Sabou, M., Uren, V., Motta, E.: = Cross-Ontology Question Answering on the Semantic Web: an initial evaluation. In: Proceedings of the Knowledge Capture Conference 2009, California, 2009
13. Jacobson, K., Sandler, M.: Interacting With Linked Data About Music. In: Proceedings of the WebSci’09: Society On-Line, 18–20 March 2009, Athens, Greece, 2009
14. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for RDF data. In: I. Cruz et al.: ISWC 2006. LNCS 4273, pp. 559–572, 2006
15. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous Semantic Web repositories. In: I. Cruz et al.: ISWC 2006. LNCS 4273, pp. 272–285, 2006
16. Hyvönen, E., Saarela, S., Viljanen, K.: Ontogator: Combining view- and ontology-based search with semantic browsing. In: Proc. of XML Finland, 2003
17. schraefel, m., Wilson, M., Russell, A., Smith, D. A.: mSpace: Improving information access to multimedia domains with multimodal exploratory search. *Comm. of the ACM* 49:4, 2006

Organization

Organizing Committee

- Christina Unger, CITEC, Bielefeld University, Bielefeld, Germany
`cunger@cit-ec.uni-bielefeld.de`
- Philipp Cimiano, CITEC, Bielefeld University, Bielefeld, Germany
`cimiano@cit-ec.uni-bielefeld.de`
- Vanessa Lopez, IBM Research, Dublin, Ireland
`vanlopez@ie.ibm.com`
- Enrico Motta, KMi, The Open University, Milton Keynes, UK
`e.motta@open.ac.uk`
- Paul Buitelaar, DERI, National University of Ireland, Galway, Ireland
`paul.buitelaar@deri.org`
- Richard Cyganiak, DERI, National University of Ireland, Galway, Ireland
`richard@cyganiak.de`

Program Committee

Lora Aroyo	Jorge Gracia	Michael Minock
Sören Auer	Gregory Grefenstette	Guenter Neumann
Abraham Bernstein	Peter Haase	Natasha F. Noy
Kalina Bontcheva	Siegfried Handschuh	Sebastian Rudolph
Mathieu D'Aquin	Andreas Harth	Krystian Samp
Aba-Sah Dadzie	David Karger	Steffen Staab
Maarten De Rijke	Jens Lehmann	Thanh Tran
Tim Finin	Bernardo Magnini	Haofen Wang

Table of Contents

Putting Linked Data to Use in a Large Higher-Education Organisation . . . <i>Mathieu d'Aquin</i>	9
Facets and Pivoting for Flexible and Usable Linked Data Exploration <i>Josep Maria Brunetti, Rosa Gil, and Roberto García</i>	22
Interacting with Statistical Linked Data via OLAP Operations <i>Benedikt Kämpgen, Sean O'Riain, and Andreas Harth</i>	36
SPARTIQLATION: Verbalizing SPARQL queries <i>Basil Ell, Denny Vrandečić, and Elena Simperl</i>	50
Improving Semantic Search Using Query Log Analysis <i>Khadija Elbedweihy, Stuart N. Wrigley, and Fabio Ciravegna</i>	61
Linguistic Modeling of Linked Open Data for Question Answering <i>Matthias Wendt, Martin Gerlach, and Holger Düwiger</i>	75
QAKiS @ QALD-2 <i>Elena Cabrio, Alessio Palmero Aprosio, Julien Cojan, Bernardo Magnini, Fabien Gandon, and Alberto Lavelli</i>	87
A System Description of Natural Language Query over DBpedia <i>Nitish Aggarwal and Paul Buitelaar</i>	96
TypeCraft Collaborative databasing and Resource sharing for Linguists . . <i>Dorothee Beermann and Pavel Mihaylov</i>	100
Author Index	107

Putting Linked Data to Use in a Large Higher-Education Organisation

Mathieu d'Aquin

Knowledge Media Institute, The Open University, Milton Keynes, UK
{m.daquin}@open.ac.uk

Abstract. In this paper, we describe applications built on top of the Open University's linked data platform (data.open.ac.uk), from the point of view of the way they implement particular forms of interactions with linked data. We especially focus on the common advantages and pitfalls in interacting with linked data that these applications illustrate, from both the end-users' and the developer's perspectives. We conclude on suggested steps forwards regarding the ways to facilitate the realisation and adoption of applications interacting with linked data.

1 Introduction

Building and deploying linked data in a large organisation represents a challenge at many different levels. Many of the past research and development works have focused on the publication process for linked data: how to obtain data from legacy information systems; how to model these data according the linked data principles; how to link the organisation's data with external sources; how to expose the data online for wide accessibility. However, it is becoming more and more clear that there is a need to investigate another, possibly more important aspect – the other side of the coin: what are the issues related to interacting with linked data, from an end-user as well as a developer perspective.

In this paper, we rely on our experience in building, deploying and applying the Open University's linked data platform (<http://data.open.ac.uk>) to investigate this perspective. Data.open.ac.uk, developed through the LUCERO project (<http://lucero-project.info>), was the first initiative to expose the public information of a university as linked open data, collecting and providing access to data from across the institution's departments¹. One of the challenges related to pioneering the use of linked data in a particular sector is the need to demonstrate the advantages that it brings to the users of the organisation. Several applications have been developed that are at different stages of their lifecycle, and have been deployed for different audiences (in size, technological

¹ It has since then been followed by a number of other initiatives applying linked data in the higher-education sector, with high potential impact regarding the reuse and interoperability of educational resources from across universities (see <http://linkeduniversities.org>)

awareness, etc.) Through these applications, common benefits as well as the challenges of providing linked data-based functionalities to 'real users' are emerging.

In this paper, we discuss some of these applications from the perspective of the way they provide means to interact with existing linked data sources. We discuss the lessons learnt from our experience regarding the issues, challenges and pitfalls of interacting with linked data. We not only consider here the perspective of the end-users in using the developed applications, but also the ones of the developers having a more direct interaction with linked data with the purpose of providing usable functionalities.

2 The Open University's Linked Data Platform

Data.open.ac.uk is a linked data endpoint that collects data from many different sources within the organisation, using a variety of different vocabularies and linking to external sources such as dbpedia.org or geonames.org (see [4]). Data collection is based on identifying streams of data inside the organisation and, in collaboration with the data owners, re-modelling the data to fit exposure as linked data. The architecture of the platform is based on a triple store providing a SPARQL endpoint, on ad-hoc mechanisms to extract and update data from the considered streams, and on a basic URI delivery mechanism. The process is continuous, with more data being exposed whenever new resources are made available. The current sets of data include:

Course information: This includes information about courses that are currently on offer at the Open University. The information includes a short description of the course, information about the levels and number of credits associated with it, the topics, and the conditions of enrolment (the countries in which it is available, the dates for registration and the student fees).

Example: <http://data.open.ac.uk/course/m366>

Research publications: This includes metadata for the research articles and other publications authored by Open University researchers, as available on the publication repository of the university (ORO, see <http://oro.open.ac.uk>). An article typically includes information about the authors, dates, abstract and venue of the publication.

Example: <http://data.open.ac.uk/oro/29916>

Podcasts: This includes the metadata for audio and video podcasts produced and made openly available by the Open University as open educational resources (see <http://podcast.open.ac.uk>). A typical podcast entity includes a short description, the topics, a link to a representative image and to a transcript if available, as well as information about the course the podcast might relate to and license information regarding the content of the podcast.

Example: <http://data.open.ac.uk/podcast/218dce44a4ed17b36ada50d18b866b03>

Open Educational Resources: This includes metadata about units of Open Educational Resources made available by the Open University through its

OpenLearn system (see <http://openlearn.open.ac.uk>). A typical 'OpenLearnUnit' includes a short description of the units, the topics, tags used to annotate the resource, its language, as well as the course it might relate to, and the license that applies to the content.

Example: http://data.open.ac.uk/openlearn/m366_2

Youtube videos: This includes metadata about videos published by the Open University on Youtube, as promotional videos or open educational resources. Such metadata include a short description of the video, the tags that were used to annotate the video, the collection it might be part of and a link to the related course if relevant.

Example: <http://data.open.ac.uk/page/youtube/A7BA7C1155BE887E/1E5D9A1BA21BDC51>

University buildings: This includes information about the building owned by the University. The Open University being a distance learning education, besides the main campus located in Milton Keynes, it also includes regional centres located in different locations across the UK territory. Building descriptions include their address (including links to the corresponding administrative areas in <http://data.ordnancesurvey.co.uk/>), a picture of the building and the sub-divisions of the building into floors and spaces.

Example: <http://data.open.ac.uk/location/building/rbedrb>

Library catalogue: This includes metadata about items available at the Open University's library that relate to Open University courses (textbooks and setbooks). The description of each item includes information about the topics, the authors, the publisher and ISBN, as well as the course it relates to.

Example: <http://data.open.ac.uk/library/406973>

Other specific data: Other datasets are also included that concern specific research projects (e.g., the Open Arts Archive – <http://openartsarchive.org/>) or specific departments of the Open University (e.g. the FOAF profiles of people from the Knowledge Media Institute – <http://people.kmi.open.ac.uk>).

3 Applying Linked Data at the Open University

Amongst the many applications developed on top of the data.open.ac.uk platform (see e.g. [3]), we choose to describe here the ones that had a concrete deployment and impact on different categories of users of the Open University, focusing on our experience regarding benefits and issues at the level of interacting with the data, both from an end-user perspective, and from a developer's perspective.

3.1 The 'Study at the OU' Mobile Application

"Study at the OU" the website of the Open University that contains the description of the courses and qualifications that can be obtained from the University (see <http://www3.open.ac.uk/study/>). A mobile application was recently developed by the communication services of the University so that this course

catalogue and additional information about the topics covered can be accessed from various types tablets and smartphones (see Figure 1). As part of this application, it is possible to select a topic and obtain information both about the courses available on this topic, and about the related resources such as podcasts, Youtube videos and OpenLearn units. This last feature is implemented using `data.open.ac.uk`, simply querying resources that are directly related to the topic being considered, or for resources attached to courses that are related to this topic.

Since its launch in January 2012, this part of the application has been accessed more than 25,000 times.

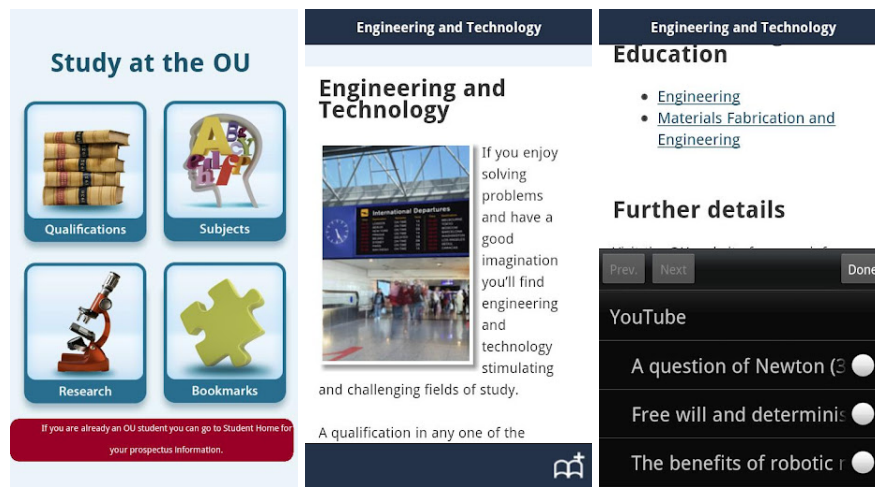


Fig. 1. Screenshots of the 'Study at the OU' application on an Android phone.

From the users' perspective, while connecting to relevant resources could be a very valuable feature, especially for prospective students, there is no indication that this has been realised with linked data. In other terms, without knowing the underlying information architecture, there is no reason not to believe that this functionality was realised using more common technologies. The added value here is however that such a simple and straightforward feature requires the combination of information coming from different, and mostly isolated systems (the course database, the podcast system, etc.), which are seamlessly integrated through linked data.

It is from **the developers' perspective** however, that the benefits of interacting with linked data to build such an application is appearing obvious. Indeed, providing the same feature using the usual information infrastructure of the Open University would have required accessing and connecting to many different systems that use different platforms, technologies, formats, conventions,

etc. It would also have required an ad-hoc integration of the data, with an additional level of complexity.

It is still a problem however for developers used to more common technologies to make use of linked data technologies such as SPARQL (<http://www.w3.org/TR/rdf-sparql-query/>), and to integrate them with their usual development environment. The solution adopted here was to create an intermediary view generating an ad-hoc XML descriptions of the relevant information from the results of pre-established SPARQL queries. This also allowed the inclusion of caching mechanisms, to avoid adding unnecessary overhead to the SPARQL query engine to process identical queries from potentially thousands of users.

3.2 Supporting the ‘Research Excellence Framework’ Activities

The ‘Research Excellence Framework’ (REF) is the process applied to evaluate and assess the quality of research in UK universities (see <http://www.hefce.ac.uk/research/ref/>). As part of this process, each university is required to submit to their corresponding funding body a report summarising the research carried out at the university in various disciplines. In order to achieve this, 18 different panels have been formed at the Open University (to cover the disciplines in which the Open University is carrying-out research), in charge of identifying individual researchers with a selection of their publication to be part of the submission.

To support this work, an application was developed (see Figure 2) to be used by individual researchers to select what they considered to be their ‘best’ publications (since 2008), to indicate to which discipline they are associated, and to annotate their selected publications to include supporting statements for their selection (describing their significance, originality, etc.) This application has already been accessed by about 600 researchers concerned with the REF at the Open University, and a similar application is being developed at the moment to support the work of the 18 panels.

The application uses linked data to obtain for each individual researcher, the list and description of their publications in the recent years, as well as to connect them with information regarding their role in the organisation and the faculty/department they relate to. The captured information (selection of publications and their annotations) is also processed according to linked data principles and technologies, creating another (private) triple store

From the users’ perspective, the main advantage of using linked data in this application is that information about their publications is directly obtained and integrated with other information, without them having to provide any additional input. Their linked data identifier (i.e., their URI) is directly derived from the login name they use to access the application (as well as all other systems on the Open Universities intranet), meaning that the relevant publications are displayed directly as they access the application. The use of linked data is in principle, as in the previous section, hidden from the user. However, some elements can sometimes create confusion due to the different modelling of the data from the original sources. In the original source for example, while each author is

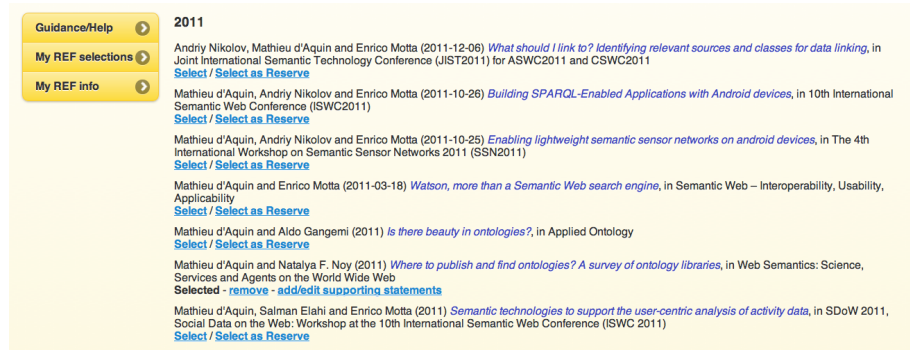


Fig. 2. Partial screenshot of the interface for researchers to annotate their publications for the REF.

associated with a unique ID, their name is recorded separately for each publication. In the linked data version, the names are all aggregated under the person's ID, meaning that each publication might appear slightly differently than in the original source (ORO). This also means that errors that can be very localised in the original source (a typo in the name of an author in one of their publications), might have a larger impact in the linked data version. Similarly, the application relies on the fact that each person and each publication is associated with a stable and unique identifier. This is however a strong assumption which, even considering well curated sources, is often hard to achieve (e.g., people changing user IDs, publications entered multiple times, etc.)

Here again, it is **from the developers' perspective** that the advantages of relying on linked data technologies and principles are the most obvious. Indeed, this allows to build on top of existing data and process various sources in an homogeneous way. Also, in producing new data (selections and annotations of publications), linked data technologies allow more flexibility and agility than with traditional, relational database systems: 'adding a field' in the data is trivial and does not require any database administration task, just adding data according to the newly considered ontological property. This is even more facilitated with the appearance of robust implementations of the SPARQL Update (<http://www.w3.org/TR/sparql11-update/>) language (on which this application relies), allowing the homogeneous use of the HTTP protocol both for querying and updating a triple store. Adding and integrating new sources of data is also made easier, as long as this information is provided using URIs consistent with the ones already in use, and the produced data is naturally reusable to build further applications.

Many issues appear here however that are not usually present when using more traditional technologies. One of them concerns the way to deal with incomplete data. Indeed, in applications like the one considered here that rely on a set of established queries, assumptions are made regarding certain properties of the

data, which are never made explicit. For example, it is expected that every publication is associated with the list of authors. In case this assumption is not valid, and some publications do not have a list of authors, these publications would simply not appear in any result, making it difficult to recognise that a potentially problematic issue have emerged. This adds to the difficulty of knowing whether a problem that is identified at the level of the application originates from the application, the linked data representation or the original sources. Dealing with such issues generates more complexity in the querying process, as well as in the development/maintenance of the application.

3.3 Understanding Research Communities at the Open University

In the continuity of the application described in the previous section, another application was developed to help research managers within the university in understanding and monitoring their research communities. Called RADAR (Research Analysis with DATA and Reasoning), this application makes use of information about research publications from ORO (as above), as well as other sources regarding the positions of researchers, their projects, funding, supervision history, etc. to visualise different indicators of research activities for individuals and groups.

Two parts of the application were developed. The first one relies on a generic framework for the visualisation and exploration of linked data sets, which is parametrised by an ontology of the particular domain of the application. It uses basic ontological reasoning to classify individuals into different classes, and uses automatically generated charts and tag clouds to visualise the distribution of values of the properties in individual classes (see Figure 3). In our case, the classes correspond for example to different categories of academic staff (senior/junior researchers, lecturers, professors, etc.) or different types of projects (internal, national, european, etc.) The indicators being visualised here correspond for example to the amount of funding received, the number of publications, of projects or the number of supervised students.

The second part of the application uses the same data and indicators, but displays graphs specifically conceived for the application in research community analysis, considering for example the distribution of the number of publications per year, or the overlap between the publications of a group of co-authors (see Figure 4).

From the users' perspective, the advantages and issues related to the generic part of the RADAR applications appear very clearly. On the one hand, the application is driven by an ontology, which means that only integrating more knowledge into it is sufficient to make it better structured and more comprehensive. There is an advantage also in having an homogeneous representation for different types of objects, and in having a comprehensive view of all the different indicators and types of data available. It is however a lot more demanding for end-users, as the interface would often include irrelevant elements. Its generic aspect (meaning that it can be applied similarly to other datasets, in other

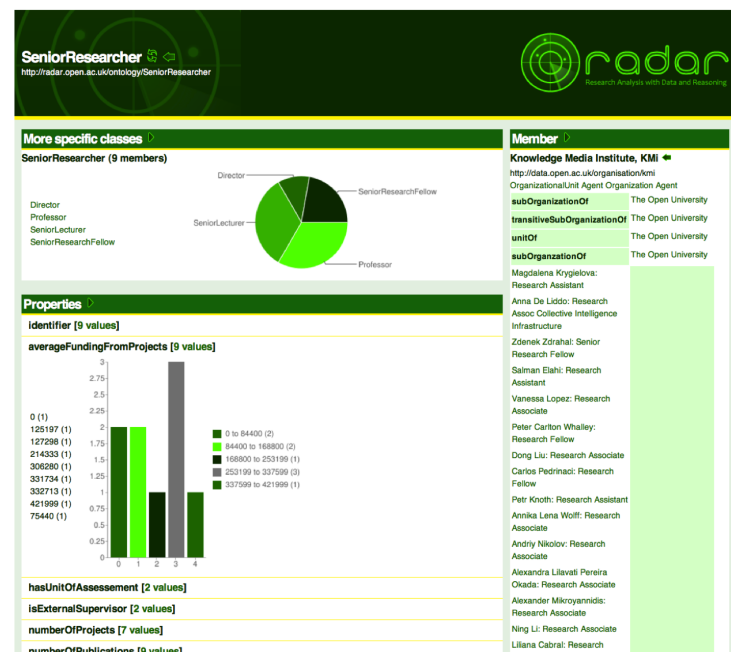


Fig. 3. Screenshot of the ‘generic’ part of the RADAR application, showing distributions of indicators for sub-classes of members of KMi.



Fig. 4. Screenshot of the ‘specific’ part of the RADAR application, showing distributions of indicators for a particular group of researchers.

domains) also means that the organisation and navigational structure of the application is guided by the modelling in the data, which is often not natural to end-users. Finally, it makes the visualisations presented harder to interpret, as they do not necessarily refer to the notions considered in the domain.

The specific part of the application was developed to counter the deficiencies of the generic view, by showing visualisations more directly relevant, more understandable and often more complex than what the generic part could do. While this has clearly been welcomed by users, the main issue with this part is that it is necessarily limited not only to the specific data and domain being considered, but also to the views that the developers of the application implemented.

From the developers' perspective, it is natural to think that the generic version of the application is more difficult to build, as it requires to abstract from the domain and data-specific assumptions that can be made with the specific version. It is however also more customisable, as many changes can be brought into it by modifying the data or the ontology on which it relies. This introduces a number of issues however, at the level of the usability of the application as mentioned above, as, without such assumptions being explicitly made, many of the results being shown to the user cannot be properly interpreted.

The specific part of the application naturally suffers from the inverse issues: while its interface design is guided by the requirements of the specific task, it requires significant efforts to be extended to support other visualisations or tasks. It is in this sense closer to the application presented in Section 3.2.

3.4 Investigating the Presence of the Open University in the Media

This last application is based on data collected in addition to what is available from data.open.ac.uk. It relies on systems used by the Media Relation services to collect clippings from news items mentioning the Open University and its members. The data collected concern the publication/channel where a news item has been issued, general metadata about the news item and possible additional information regarding researchers, lecturers or other members of staff of the Open University cited or who contributed to the news item. Links are also created to data.open.ac.uk (regarding people) and to dbpedia.org (regarding the publications and channels that provided the news items).

An application has been created that allows members of the Open University to create charts and reports on top of this data. A 'chart generation' interface is provided that allows to create filters and identify categories to be visualised based on properties and values in the data. Once configured, the interface creates a linkable and embedable chart that can be customised, and is dynamically updated based on changes in the data (see the example Figure 5).

Here, two different types of users need to be considered. For the **users of the chart**, the results are reasonably straightforward, as the charts can be embedded into dedicated interfaces that implement dashboards and navigation mechanisms that are understandable in the particular context. This application also clearly demonstrates the benefits of linked data, especially through exploiting the links with external sources of data (in the example Figure 5, the information about

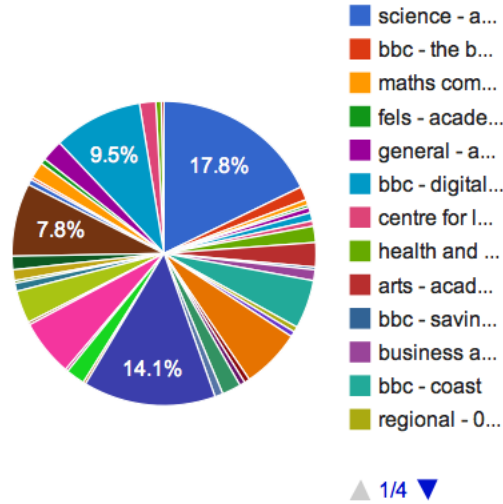


Fig. 5. Charts showing the distribution of topics of news items mentioning the Open University and published through channels owned by the BBC.

topics and number of corresponding news items is provided by the local data, while the information regarding the channels owned by the BBC is provided by dbpedia.org).

The other type of users are the ones **creating the charts** using the configuration interface. This interface requires to be able to understand and follow the properties used in the data for different types of objects. In other terms, while it is generic and can be used to generate charts from any linked data endpoint, it needs for the user to be familiar with the schema used in the data, the particular properties of the data, and with the specificities of their linked data-based modelling.

From the developers' perspective, this application represent an example of a generic application, but which is used to generate specific, customisable visualisations. In this sense, it completely abstract from any specificities of the data being considered, but on the other hand, cannot provide much guidance to the users with respect to the use of the data, and to the interpretation of the results.

4 Conclusions: Challenges and Pitfalls of Interacting with Linked Data, and Steps Fowards

The four applications presented above represent concrete experiences of developments relying on linked data that have been actually deployed and used in

an environment more accustomed to proprietary, corporate software relying on more common information systems and data management solutions. As such, they illustrate common challenges and pitfalls that interacting with linked data can generate, from both the end-users' perspective and the developers' perspectives. In this section, we summarise the general notions that appear from such experiences to require attention, which we believe help identifying important directions for research into interacting with linked data.

First, it appears clearly that linked data should be hidden from the end-users. While this might appear trivial, this is not an easy goal to achieve: from our experience, we can see that most of the advantages of linked data should appear obvious to the application developers, but should as much as possible not need to be understood by the end users. This of course concerns purely technical elements such as URIs, RDF and SPARQL, but also more conceptual considerations, such as the integration of multiple sources of data or the use of reasoning. This is the case of our “Study at the OU” app and of the REF support application, where little issues appear on the end-user side. In RADAR and the media relation applications on the contrary, whenever the technology is too present, and even if it is to provide advanced features, it introduces confusion for the users. In other terms, while it is often still needed to convince stakeholders of the value of linked data, applications that are technological demonstrators tend to have little value to the users, and the technology should be essentially be considered from the point of view of the developers.

Second, there is an elusive trade-off to be found between developing generic, reusable frameworks that can be applied on a large variety of datasets and domains, and specific applications that are meant to work only with certain datasets. Indeed, while the value of reusable components is quite obvious, and making this possible is one of the strong benefits of linked data, the RADAR application clearly shows that achieving an intuitively useable application that relies on a generic template for navigation and presentation of the data is close to impossible. Most applications of linked data nowadays are closer to the “Study at the OU” app or to the REF application: applications working in a close environment with a clearly defined and understood set of datasets and queries. These applications tend to be disappointing as they cannot benefit from the openness of linked data and the possibility to integrate data which might originate from other organisations, possibly at run-time. As a conclusion, rather than a complete, generic application framework like what was attempted with the generic part of RADAR, we believe that what is needed are libraries of reusable and highly customisable interface components that rely on generic linked data resources, but can be flexibly integrated to create specific application interfaces in specific scenarios. The chart creation feature represented by the media relation application can be seen as an example of such a library. Other initiatives exist that provide initial building blocks, such as the SPARK (<http://km.aifb.kit.edu/sites/spark/>) javascript library, the SIM-

ILE timeline and map widgets (<http://simile-widgets.org/>), as well as more general visualisation components, such as Fusion [2] or the Linked Data API².

Third, it appears clearly that one of the obstacles to building reusable interface components based on linked data is the openness and flexibility of the data model on which linked data relies. At a higher level, integrating data from external sources represent a major challenge, as interaction needs to implement a trade-off between control and the potentially infinite possibilities that opening the interface to unknown data can bring. More concretely, many tasks require assumptions related to the data, which are rarely made explicit and formalised. For example, most of the applications we have considered make the assumption that each entity appearing in the interface are associated with a human readable label. It is often the case that, if not requiring that there is only one human readable label for each entity, the application will make a choice between the ones available, either randomly or using domain- and data-specific criteria. Similarly, while operational and functioning in most situations, it is clear that the charts generated by the media relation application can only make sense under certain conditions. For example, if entities can be associated with more than one identifier, this would generally lead to these entities being counted as if they were multiple entities. Also, charts showing distributions of data would be misleading if not all the entities considered have the same number of values for the visualised property or if some of them do not provide values. In other terms, while the fact that the formalisms underlying linked data do not make the closed-world assumption or the unique-name assumptions is at the basis of their openness and flexibility, it makes them less exploitable as part of generic data processes. Our suggestion related to this process would be to create a way to annotate datasets (including such annotations for example in the Void [1] descriptions of the datasets) related to the particular ‘characteristics’ of the datasets that can be exploited by data processing/visualisation mechanisms, including for example “local unique-name assumptions” indicating that the instances of a certain class are non-redundant, as well as expressions similar to integrity constraints (e.g., that there is necessarily a value of a given property for the instances of a given class). Being able to rely on such characteristics would make it more feasible for generic interface components, as previously suggested, to provide some levels of guaranties regarding their interpretability and usability when applied on particular data.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets – On the Design and Usage of void, the ‘Vocabulary of Interlinked Datasets’. In: Proc. of Linked Data on the Web, WWW 2009
2. Araújo, S., Houben, G.-J., Schwabe, D., Hidders, J.: Building Linked Data Applications with Fusion: A Visual Interface for Exploration and Mapping. In: ISWC Posters and Demos (2010)

² <http://code.google.com/p/linked-data-api/>

3. Zablith, F., d'Aquin, M., Brown, S., Green-Hughes, L.: Consuming Linked Data Within a Large Educational Organization. In: Proc. of the Second International Workshop on Consuming Linked Data (COLD), ISWC 2011
4. Zablith, F., Fernández, M., Rowe, M.: The OU Linked Open Data: Production and Consumption. In: ESWC Workshops Proceedings, Linked Learning 2011

Facets and Pivoting for Flexible and Usable Linked Data Exploration

Josep Maria Brunetti, Rosa Gil, and Roberto García

Universitat de Lleida, Jaume II, 69. 25001 Lleida, Spain
{josepmbrunetti, rgil, rgarcia}@diei.udl.cat

Abstract. The success of Open Data initiatives has increased the amount of data available on the Web. Unfortunately, most of this data is only available in raw tabular form, what makes analysis and reuse quite difficult for non-experts. Linked Data principles allow for a more sophisticated approach by making explicit both the structure and semantics of the data. However, from the end-user viewpoint, they continue to be monolithic files completely opaque or difficult to explore by making tedious semantic queries. Our objective is to facilitate the user to grasp what kind of entities are in the dataset, how they are interrelated, which are their main properties and values, etc. Rhizomer is a tool for data publishing whose interface provides a set of components borrowed from Information Architecture (IA) that facilitate awareness of the dataset at hand. It automatically generates navigation menus and facets based on the kinds of things in the dataset and how they are described through metadata properties and values. Moreover, motivated by recent tests with end-users, it also provides the possibility to pivot among the faceted views created for each class of resources in the dataset.

Keywords: Semantic Web, Linked Data, Human-Computer Interaction, usability, interaction

1 Introduction

Though the amount of available following the Linked Data principles [2] is rapidly increasing, from the end-user perspective, the view continues to be that the available datasets are monolithic files completely opaque, or that can only be explored using complex semantic queries. The objective should be now to try to make all this data more usable so users facing a dataset can easily grasp what kind of entities are in there, how they are interrelated, what are their main properties and values, etc.

This way, users that are not experts in Semantic Web technologies can search, browse and analyse the data. This will increase the awareness of the data currently available in the Web and also facilitate the development of new and innovative applications on top of it. The overall outcome is that available data increases its impact and the society as a whole benefits more from data openness.

The best approach to make a dataset more usable to a wider range of users is to use some sort of Web publishing tool. At least, this kind of tools usually provides a HTML rendering for each resource in the dataset. Each HTML page lists all the properties for the corresponding resource. Pages are interlinked based on the connections among resources and the user can follow HTML links to browse through them. However, this feature is only useful if the user has some a priori knowledge about the dataset, especially the URI identifier of a given resource. There is no way to get at least an overview of the kind of resources in the dataset.

Our proposal is to draw from the experience accumulated in the Information Architecture (IA) domain [14] and reuse and adapt existing IA components to provide this kind of guide to users. These components are well known to Web users, as they are present in most web pages: navigation bars, facets, sitemaps, breadcrumbs, etc.

This approach is being applied in Rhizomer, a tool capable of publishing a Linked Data dataset while facilitating user awareness of its content. It is also being evaluated with end-users as part of a User Centred Design development process. Iterative evaluations have motivated and guided the introduction of new features, like pivoting, and validated improvements in the usability.

The rest of this paper is organised as follows. First, the related work is presented in Subsection 1.1. Then, the approach is introduced in Section 2, detailing the Information Architecture components used to implement it. In Section 3, an improvement is presented, motivated by a first round of evaluations with end-users. It is the possibility to use facets to pivot around the resource types in the dataset. An evaluation of this feature is also included in this section. Finally, conclusions and future plans are presented in Section 4.

1.1 Related Work

Publishing and presenting Linked Data in an accessible way for users has been addressed by several projects. The first tool that comes to mind when trying to realize what a dataset is about is a Semantic Web browser. Dadzie et al. [6] analyzed some of those projects finding that most of them are designed only for tech-users and they do not provide an overview of the data.

Browsers are especially useful when dealing with a dataset published as Linked Data because they provide a smooth browsing experience through the graph, e.g. Disco [4] or Tabulator [3]. However, they do not provide additional support for getting a broader view of the dataset being browsed, just a view on the current resource. With Explorator [1] it is possible to browse a dataset available through a semantic queries service. Though Explorator makes it possible to browse the dataset by combining search, facets or operations on sets of resources, it is still difficult to get a broader view on the dataset other than a list of all the classes or properties used.

In some cases it is also possible to get more informative components like facets, e.g. /facet [10]. However, in some cases, facets are pre-computed and just available for a given dataset as in the case of the DBPedia Faceted Browser [9].

Visor [15] is a generic data explorer that allows pivoting to related instances. Despite it provides a hierarchical overview of the dataset, it is complicated to filter resources because there isn't any faceted interface. Parallax [11] also provides the possibility to pivot to related types and a faceted browser, but it works only with the Freebase dataset.

Other alternatives are Content Management Systems (CMS) or Wikis with semantic capabilities. Some mainstream CMSs and wiki systems have started to incorporate semantic technologies. The most significant case is the last version of Drupal¹ that provides features such as semantic metadata storage, querying, importation or rendering. The same applies to semantic wikis, such as Semantic MediaWiki [12], that makes it possible to mix wiki mark-up with semantic annotations. However, semantic CMSs and wikis are intended more for content creation than for importing and exploring existing ones.

There are also specialised tools that publish existing datasets as Linked Data. For instance, Pubby² builds a Linked Data frontend with dereferenceable URIs for the subjects in the served dataset and content negotiation. It also features a metadata extension that provides provenance information. However, in both cases, the frontends provided are just like those Semantic Web browsers have.

To conclude, it is also possible to consider platforms for semantic data storage and publishing like OpenLink Virtuoso³. In addition to the data store, there is an HTML frontend that provides a basic faceted view for the datasets. However, its interface is not intuitive at all and like with previous tools, the support for broader awareness of the dataset structure is very limited.

Consequently, existing tools make it very difficult for non-technical users to explore a dataset, realize what kind of resources there are, what properties they have and how they are related.

2 Approach

Our starting point is the fundamental set of tasks for data analysis proposed by Shneiderman [17]. Next, we present each task associated to the interaction pattern chosen to satisfy it and Information Architecture [14] component selected to implement the pattern:

- **Overview:** get the full picture of the data set at hand. At this stage we propose to apply the Global Navigation interaction pattern⁴. In Information Architecture terms, it corresponds to the navigation bars users are used to see at the top or on the right of web sites.
- **Zoom & Filter:** zoom in on items of interest and filter out uninteresting items. Here the proposal is Faceted Navigation⁵. Once we have zoomed by

¹ Drupal 7, <http://drupal.org/drupal-7.0>

² Pubby, <http://www4.wiwiw.fu-berlin.de/pubby/>

³ <http://virtuoso.openlinksw.com>

⁴ <http://www.welie.com/patterns/showPattern.php?patternID=main-navigation>

⁵ <http://www.welie.com/patterns/showPattern.php?patternID=faceted-navigation>

selecting the kind of things we are interested in from the navigation bar, facets are the Information Architecture components that help users filter out those that are not interesting.

- **Details:** after zooming and filtering the user arrives to the concrete resources of interest. At this point, the user can get the details for those resources, which in the case of Linked Data is to get the properties for the resources plus those properties pointing to them. This is related to the Details on Demand⁶ interaction pattern and can be implemented as a simple list of properties and values of the resource of interest or as a specific visualisation tailored to the kind of resource at hand, e.g. a map for geo-located resources.

Our proposal is to elaborate these interaction patterns in the context of Linked Data. We have chosen them because they are simple and very common so users are very comfortable with them. They are currently part of the culture about how information is structured in the Web so they have been deeply studied in Information Architecture (IA) domain.

We are currently testing all these interaction patterns in a Linked Data publishing tool called Rhizomer⁷. It features navigation bars automatically generated and maintained starting from the underlying thesaurus and ontologies, and how they are structured and instantiated. Navigation menus are described in Section 2.1. A similar approach is followed for generating facets for each kind of entity in the data set. Facets are described in Section 2.2. Fig. 1 is a screen capture of Rhizomer showing at the top the navigation menu and on the left the generated facets.

2.1 Navigation Menus

Navigation menus, in the case of websites, let users navigate through different sections and pages of the site. They tend to be the only consistent navigation element, being present on every page of the site.

Traditionally, user-centred design techniques like Card Sorting [18] are used to develop the navigation menus of web sites. This technique requires a lot of time and effort from developers and most of this effort is wasted as soon as the structure of the menu is established and fixed in a menu that becomes something static. If new kinds of items are introduced or a part of the content becomes more relevant, the Card Sorting should be repeated, at least in part.

The opportunity in the case of web sites build on top of semantic data is to automate part of the process of generation and maintenance of the navigation menus. This is possible because semantic data is structured by thesaurus and ontologies, which hierarchically organise the kinds of things described in the dataset. They specify all the classes or concept⁸ but also which subjects belong to each class or are related to each concept.

⁶ <http://www.welie.com/patterns/showPattern.php?patternID=details-on-demand>

⁷ <http://rhizomik.net/rhizomer/>

⁸ SKOS Simple Knowledge Organization System, <http://www.w3.org/2004/02/skos/>



Fig. 1. Screen capture of Rhizomer, at the top there is the navigation menu and on the left the generated facets.

Consequently, if there are few members of some kind, or there are not at all, it should be less relevant in the menu bar. On the contrary, those that do have a lot of members should be shown prominently in the menu bar. To do this, we obtain the hierarchy of all kinds of entities and apply inference rules to get their members. Then, the hierarchy is flattened to the amount of levels required because this component can generate both global and local menus, i.e. a menu for the whole dataset or for a subset of it. The site administrator can also configure some parameters: the number of levels in the menu, the number of items in each level, the order of items (alphabetical or by number of instances) and a list of classes or concepts to omit.

According to these parameters, this component generates the menu applying a recursive algorithm that mainly performs two operations: Split the concepts or classes with a large amount of members in their narrower related concepts or subclasses. Group those with few members into a broader concept or superclass.

This approach makes it possible to show the user the navigation bar that best fits the data in the dataset at that particular moment. For instance, if the dataset changes from containing mainly data about projects to mainly about publications, the menu would change accordingly to show more prominently the part of the dataset structure about publications. More details about the implementation of navigation menus are available from [7].

2.2 Facets

Users don't always know exactly what they are looking for and, sometimes, they don't even know what its name is. Other times, they are unfamiliar with the domain or they want to learn about a topic. This is particularly true when facing Semantic Web datasets. In these cases, exploratory search is a strategy

that allows users to refine their search by successive iterations. An exploratory interface such as faceted browsing allows users to find information without a priori knowledge of its schema.

With navigation menus we can make the user aware of the hierarchical structure of a dataset but, once they choose the class of things they are interested in, they face the barrier of not knowing how they are described. In other words, what are the main properties that describe them, which ones are the more relevant for that particular kind of things, the range of values they have in that particular case, etc. Faceted navigation is an exploratory technique for navigating a collection of elements in multiple ways, rather than a single and pre-determined order. Facet browser interfaces provide a user-friendly way to navigate through a wide range of data collections.

Traditional facet browsers relied on manual identification of the facets and on previous knowledge of the target domain. When dealing with semantic data, it is possible to automate this process and a semantic faceted browser should be able to handle any RDF dataset without any configuration. Since Linked Data facilitates integrating data from different sources, we can't assume a single fixed schema for all data. Consequently, a Linked Data faceted browser should be scalable and generic, not depending on a particular dataset structure.

In traditional Web, facet browsers are developed to navigate through homogeneous data and facets are fixed. This conflicts with Linked Data, where data is too diverse to use a single set of facets: facets that make sense for one type of entity could be inappropriate for other types. Moreover, when new data is added the system should be able to add new facets at run time.

To build the facets, and to keep them updated, what Rhizomer does is to perform queries for each class in the dataset ontologies that retrieve all the properties their members have, the different values each property has and the cardinality for each value, i.e. how many times that property for that class takes that value.

Facets are calculated and stored in a data structure. They are then updated whenever the dataset is edited through Rhizomer. They are also updated, but just a local copy associated to a user session, when the user starts browsing and selecting values for different facets. In this case, the set of instances used for facets generation is constrained by the choices made so far and the facets are recalculated for that constrained set of results obtained so far. Those facets that are no longer relevant, i.e. no result uses them, are removed from the facets set. More details about the implementation of facets are available from [7].

3 Pivoting

From the point of view of OLAP systems, pivoting or rotation is described as an operation producing a change in the dimensional orientation of data. For instance, if data is initially aggregated by Product, Location and Date, by pivoting, the user can aggregate, for instance, by Location, Date and Product.

However, for richer data models pivot navigation is a way to restart a search from the results of a first search [16]. Usually, the type of resources to be browsed (e.g. book, car, paper...) remains fixed in a faceted browsing application. However, when pivoting is added to faceted navigation, it allows switching the type of displayed entities based on relations to the current result set. For instance, a user who is filtering films using film facets, e.g. director is Woody Allen, then pivots on actors. As a result of this action, the user will see now all actors in the result list, who are related to any film in the previous filtered list and continue filtering but now using actor facets, e.g. country is Spain.

Here, it is possible to establish an interesting analogy between pivoting and natural language. Indeed, the query above can be rephrased as Show actors whose country is Spain, which have acted in films directed by Woody Allen. The idea of pivot is reflected by the fact that the set of Spanish actors in the main sentence also appears in the relative sentence as the relative pronoun which. The relative pronoun point to the facet to browse for a pivot, in this case acted in.

Pivot steps can be repeated, e.g. pivot on countries from actors and filters continent is Europe, after removing the previous country is Spain filter from actors. Each pivot step corresponds to a nested relative sentences, such as Show European countries, where an actor, which has acted in a Woody Allen film, has been born. We have profited from this resemblance to natural language to generate more usable breadcrumbs that help users contextualise their exploration and know why they are getting the list of results that they are looking at as a result of their filtering and pivoting steps so far.

Next subsection illustrates the importance of offering pivoting to users exploring Linked Data, as shown in a preliminary evaluation of Rhizomer with end-users. The implementation details and the results of a new evaluation iteration are then presented in the following subsections.

3.1 Motivation

Pivoting is not a common feature of existing Linked Data exploration tools. To our knowledge, the only active tools capable of providing a faceted view and pivoting on semantic data are Parallax⁹ and Virtuoso Faceted Browser¹⁰. Parallax is constrained to the Freebase dataset though there is an attempt to make it capable to work on top of any SPARQL endpoint called SParallax¹¹. However, in both cases, they lack a mechanism to provide an overview of the dataset and they have some performance issues, especially in the case of SParallax. Facets and pivoting in Virtuoso Faceted Browser are not so evident and quite difficult to use, though it also offers a breadcrumbs system based on the natural language analogy.

⁹ <http://www.freebase.com/labs/parallax/>

¹⁰ <http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtuosoFacetsWebService>

¹¹ <http://sparallax.deri.ie>

However, during the last round of user testing it became quite evident that a mechanism like pivoting was required. Evaluations are conducted at the UsabiliLAB¹² and inspired by the Rapid Iterative Testing and Evaluation (RITE) Method [13], testing with small groups as part of an iterative development process. To register sessions we used Morae Recorder and Morae Observer to analyse user test data. For the usability test metrics we chose effectiveness (percentage of tasks completed, workload) and efficiency (time to complete a task).

For testing, we used a real dataset called the Linked Movie Data Base (LinkedMDB)¹³. We chose this dataset because movies is a well know topic for most people and quite appealing. LinkedMDB is generated from the Internet Movie Database¹⁴ (IMDB).

Six participants were selected, with a unique profile characterized by good knowledge of information technology, limited knowledge about Semantic Web technologies and interest in movies. The reason we chose this profile is because Rhizomer is intended for users interested in working with data, used to information technologies but not necessarily aware of Semantic Web and Linked Data.

The test facilitator proposed users, among others, the following task: Find three films where Woody Allen is director and actor at the same time. The full findings derived from the test are available from [5], while the ones relevant to motivating the inclusion of the pivot operation are summarised next:

- **100%** of participants needed in at least one occasion the guidance of the facilitator to successfully complete the task.
- **100%** of the participants began the navigation from actors instead than from movies (the directors where not so evident in the navigation bar because there are not so many instances than for other classes and they appear as a second level option in the menus).

When analysing the evaluation results, it became evident that the fact users started from actors was the reason why they required assistance, they arrived at a dead-end after filtering actors by name to just Woody Allen. There was no way to switch to his set of films and then filter it using the director facet.

A short path to this problem might be to add for each resource, e.g. Woody Allen (Actor), a link for each facet to the set of resources that can be reached through it, e.g. a link to all the films where Woody Allen has acted. However, this just works for particular instances and the objective is to make it also work for sets of resources, e.g. all the films by a Spanish actor. This motivated the development of a pivot operation that can be performed even before filtering has reduced the set of explored resources to just one.

Moreover, another conclusion from the evaluation was that navigation should be better contextualised. The interface should provide more mechanisms to inform the user where she/he is, where she/he can go and where she/he has been.

¹² UsabiliLAB, <http://griho.udl.cat/en/infraestructures/usabililab.html>

¹³ LinkedMDB by O. Hassanzadeh and M. Consens, <http://linkedmdb.org>

¹⁴ IMDB database, <http://www.imdb.com>

For that, the proposal is to integrate some kind of breadcrumbs that summarise the navigation steps through navigation menus and facets.

The previous findings motivated a next round of work with Rhizomer. The focus was now on incorporating the pivot action and also on improving contextualisation by including breadcrumbs, as detailed in the next section.

3.2 Implementation

To build facets that support pivoting, we first distinguish three types of properties:

- **Datatype properties:** properties whose values are RDF literals or data types from XML Schema. It is not possible to pivot on these properties but recognising them allows displaying them with specialised facet types, e.g. a histogram facet for numbers or a calendar one for dates. These specialised facets are still work in progress.
- **Object properties:** properties whose values reference other resources. These properties were treated, prior to the introduction of pivoting, as facets with literal values, where the values were resources labels. It continues to be possible to filter a set of resources based on the labels of the referenced resources, e.g. filter films through the actor facet based on the actors labels. However, pivots makes also possible to switch to the set of actors and perform a more detailed filtering based on actors facets.
- **Inverse properties:** in some cases, a dataset has a property between resource types modelled just in one way. For instance, each resource of type Film has the property actor, but the resources of type Actor do not have the inverse property to relate them with the films they have appeared in. When inverse properties are not explicit in a dataset, they are detected and facets are generated following the same approach than for explicit object properties. Consequently, it is possible to pivot through explicit object properties and also through implicit inverse object properties. This increases the flexibility of the exploration as more choices are available to the user dead-ends are avoided, like exploring actors and not being able to pivot to films because the property from actors to films is not explicitly stated in the dataset.

Each facet consists of a list of the five most used values and a text search box, which suggests possible matches. There is also the possibility to see the rest of values and choose from them. In the case of object properties and inverse properties we provide the option to pivot to their related types, marked as an arrow icon as shown in Fig. 2. Thus, users can construct complex queries that include different types of resources.

We keep a list of the selected values for each facet and also information for every pivoted step realized. This information is used to build the corresponding SPARQL queries to get results. For example, the SPARQL query when browsing films whose director is Woody Allen is:

```
SELECT DISTINCT ?r1 WHERE {
  ?r1 a <http://data.linkedmdb.org/resource/movie/film> .
  ?r1 <http://data.linkedmdb.org/resource/movie/director>
    <http://data.linkedmdb.org/resource/director/8501> }
```

For those facets that allow pivoting to another type we save the following information: the initial type, the property used to pivot and the target type. In the previous example, when user pivots from Film to Actor:

- Initial type: `http://data.linkedmdb.org/resource/movie/film`
- Pivoting property: `http://data.linkedmdb.org/resource/movie/actor`
- Target type: `http://data.linkedmdb.org/resource/movie/actor`

When pivoting to another type, the restrictions applied to the first one are propagated to the pivoted type through the property used for pivoting and linked using variables, marked in bold in the following SPARQL example after pivoting from films to actors:

```
SELECT DISTINCT ?r2 WHERE {
  ?r2 a <http://data.linkedmdb.org/resource/movie/actor>.
  ?r1 a <http://data.linkedmdb.org/resource/movie/film>.
  ?r1 <http://data.linkedmdb.org/resource/movie/director>
    <http://data.linkedmdb.org/resource/director/8501>.
  ?r1 <http://data.linkedmdb.org/resource/movie/actor> ?r2.
  ?r1 a <http://data.linkedmdb.org/resource/movie/film> }
```

To generate breadcrumbs we use also the restrictions used to build queries. Breadcrumbs show the path that the user has followed to arrive to that set of results. Users can use them to remove filters and pivoted steps. We have used natural language to generate them because we think this is the easiest way to understand the filtering and pivoting steps, as already introduced in Section 3. Fig. 2 shows, in the central part, the breadcrumbs generated for this example.

3.3 Evaluation

This section summarises the results of the first round of user testing after integrating the pivoting functionality into Rhizomer. The evaluation approach is inspired by the Rapid Iterative Testing and Evaluation (RITE) Method [13], testing with small groups as part of an iterative development process, but tests are performed much more frequently.

The aim of the test presented in this section was mainly to validate that the introduction of pivoting solves the problems highlighted in the previous evaluation, described in Section 3.1. Six users that were not involved in the previous evaluation rounds were recruited and one of the tasks, Task 2, was identical to one of the tasks used in previous rounds.

This is the task that is going to be used to test if pivoting has improved efficiency and efficacy. The complete list of tasks is presented in Table 1 and the

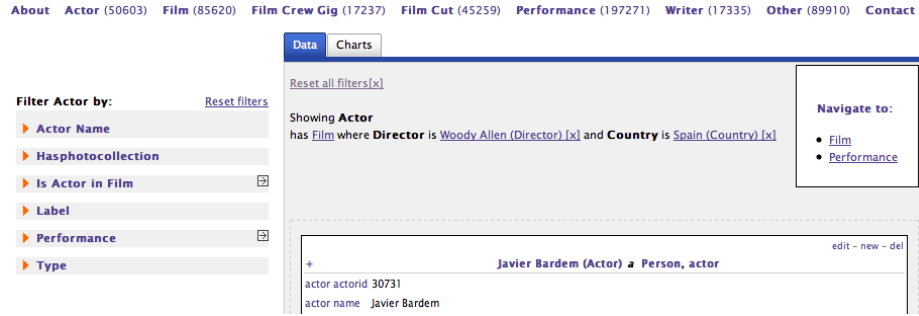


Fig. 2. Capture showing the pivoting enhancements: breadcrumbs as natural language in the middle, pivot-able facets with arrow icon and Navigate to box with pivoting options

Table 1. Tasks posed to users during the pivoting testing round

Task 1	Find 5 films with "Orlando Bloom" as actor.
Task 2	Find 5 films with "Clint Eastwood" both as director and actor.
Task 3	Who has directed more films in countries located in "Oceania".

efficiency results are shown in Table 2. More details about previous evaluations are available from [8].

The first finding has been that the introduction of pivoting has corresponded to a great increase of efficiency, with a 30% reduction in the mean time necessary to complete Task 2. However, the most promising outcome is that the biggest improvement has been in the reduction of the maximum time on tasks, 57% improvement.

Table 2. Efficiency for the tasks in Table 1 plus comparison with pre-pivoting results for Task 2

Time on Task (minutes)	Task 2 with pivoting	Task 2 pre-pivoting	Improvement	Task 1 with pivoting	Task 3 with pivoting
Minimum	0.89	1.05	15%	1.00	1.99
Maximum	2.23	5.23	57%	4.53	4.50
Mean	1.69	2.41	30%	1.61	3.43
Standard Dev.	0.57	1.49	62%	1.21	0.96

This is related with the fact that, thanks to pivoting, all users were able to find their path to solve the task without requiring assistance. Contrary to pre-pivoting tests, where most users got lost when trying to complete the tasks starting from actor or director instead of from film, with pivoting all users were

able to complete the task independently of their starting point without assistance. Consequently, the maximum time was reduced significantly.

However, there is still a lot of room for improvement. The following issues were detected and some proposals to solve them are presented in the conclusions and future work section:

- It was users for users to identify the pivoting button associated to object property of inverse property facets. Moreover, the box labelled Navigate to, that also contained the list of facets that provided pivoting was far from the facets, in the right side while facets are in the left side, and hard to identify. Finally, some users thought that following one of this pivoting links supposed starting from zero the exploration from the target class, loosing all the restrictions done so far through facets.
- Users also experienced many contextualisation problems, not being clear for them what were they seeing at the screen. The breadcrumbs helped solving this once the users realised they were available but this took some time for most of them.

4 Conclusions and Future Work

After some rounds of development and testing with end-users, Rhizomer is capable of publishing Linked Data while facilitating user awareness of what is in the dataset. Awareness is accomplished by components borrowed from the Information Architecture discipline and generated automatically from the dataset structure and ontologies. They are navigation bars, which show the main kinds of items in the dataset, and facets, which show the more significant properties for different kinds of items and their values.

Our preliminary tests with users show that Rhizomer facilitates the exploration of the published datasets, and they have also highlighted many issues. The last functionality addition, motivated during a previous evaluation round, is pivoting. The possibility of allow users move from faceted exploration of a particular class of resource, e.g. Actors, to another class of related resources, e.g. Film, while preserving the previous filters, provides a great level of flexibility to the interaction and avoid dead-ends due to the way the data is structured.

Pivoting has allowed reducing the mean time to complete a particular task, thus improving efficiency, by a 30%. Moreover, the maximum time has been reduced by almost a 60% as a result of the fact that now, contrary to the test prior to the introduction of pivoting, all users were able of completing the task without assistance.

The remaining issues, spotted during the last evaluation, are mainly related with the fact that the interface components providing pivoting are not so evident for users. Moreover, they suppose a conceptual shift that should be mitigated. For instance, some users saw pivoting as restarting exploration for a new class of resources.

One possible way to overcome these limitations of pivoting is to try to integrate it with facets, so users do not need to take their attention from facets, and

also to make it clearer that the filtering done so far is not going to be lost. One way to do that, to be implemented and tested, is to present pivoting as a way of doing advanced filtering on a facet.

This way, users can start doing classical filtering using the labels of the facet values. For instance, filter the actors for a film using the actor facet and the actor labels. However, if they get stuck because they need a more sophisticated filter, make pivoting available as a way of attaining advanced filtering. This got a promising way to go as a result of Task 3 used in the evaluation presented in Section 3.3. Users looked for films from countries in Oceania but got stuck because the country facet showed just country names. Some of them pointed out that they needed to filter the countries by continent but did not see pivoting as the solution to their problem. Their idea is that we can link their need of performing advanced filtering on properties of the entities associated to a facet to pivoting.

Finally, another issue is related with contextualisation. Though natural language-inspired breadcrumbs have been seen by users as very useful, they should get more prominent in the user interface because it took too much time to users to spot them. The idea in this case is to increase breadcrumbs size and use colours that highlight the breadcrumbs and also the entities involved, i.e. the names of the classes and facets involved.

References

1. Araujo S., Schwabe D., Barbosa S.: Experimenting with Explorator: a Direct Manipulation Generic RDF Browser and Querying Tool. In: Proc. of VISSW 2009
2. Berners-Lee, T.: Linked Data. Design Issues (2009) <http://www.w3.org/DesignIssues/LinkedData.html>
3. Berners-Lee, T. Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D. Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: Proc. of SWUI06 at ISWC 2006
4. Bojars, U., Passant, A., Giasson, F., Breslin, J. G.: An Architecture to Discover and Query Decentralized RDF Data. In: Proc. of SFSW 2007, CEUR Workshop Proceedings 248 (2007)
5. Brunetti, J.M., García, R.: Information Architecture Automatization for the Semantic Web. In: Proc. of the 13th IFIP TC 13 International Conference on Human-Computer Interaction (2011) 410–413
6. Dadzie, A.-S., Rowe, M.: Approaches to Visualising Linked Data: A Survey. The Semantic Web Journal – Special Call for Semantic Web surveys and applications **2**(2) (2011) 89–124
7. García, R., Brunetti, J.M., López-Muzás, A., Gimeno, J.M., Gil, R.: Publishing and interacting with linked data. In: Proc. of the International Conference on Web Intelligence, Mining and Semantics (2011) 18:1–18:12
8. García, R., Gimeno, J.M., Perdrix, F., Gil, R., Oliva, M., López, J.M., Pascual, A., Sendín, M.: Building a Usable and Accessible Semantic Web Interaction Platform. In: Proc. of WWW 2010 143–167
9. Hahn, R., Bizer, B., Sahnwaldt, C., Herta, C., Robinson, S., Bürge, M., Düwiger, H., Scheel, U.: Faceted Wikipedia Search. In: Proc. of BIS10

10. Hildebrand, M., Ossenbruggen, J., and Hardman, L.: /facet: A Browser for Heterogeneous Semantic Web Repositories. In: Proc. of The Semantic Web at ISWC 2006, 272–285
11. Huynh, D. F., Karger, D. R.: Parallax and companion: Set-based browsing for the data web. In: WProc. of WWW 2009
12. Krötzsch, M., Vrandečić, D., Völkel, M.: Semantic MediaWiki. In: Proc. of ISWC 2006, LNCS 4273 (2006) 935–942
13. Medlock, M.C., Wixon, D., Terrano, M., Romero, R.L., Fulton, B.: Using the RITE method to improve products: A definition and a case study. In: Proc. of the Usability Professionals Association (2002)
14. Morville, P., Rosenfeld, L: Information Architecture for the World Wide Web. O'Reilly Media (2006)
15. Popov, I., schraefel, m., Hall, W., Shadbolt, N.: Connecting the Dots: A Multi-pivot Approach to Data Exploration. In: Proc. of ISWC 2011, 23–27
16. Sacco, G.M., Tzitzikas, Y. (eds.): Dynamic taxonomies and faceted search: theory, practice, and experience. Springer (2009)
17. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: Proc. of the 1996 IEEE Symposium on Visual Languages (VL'96) 336–343
18. Spencer, D.: Card Sorting. Rosenfeld Media (2009)

Interacting with Statistical Linked Data via OLAP Operations

Benedikt Kämpgen¹, Sean O’Riain², and Andreas Harth¹

¹ Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany
{benedikt.kaempgen,harth}@kit.edu

² Digital Enterprise Research Institute, National University of Ireland, Galway
sean.oriain@deri.org

Abstract. Online Analytical Processing (OLAP) promises an interface to analyse Linked Data containing statistics going beyond other interaction paradigms such as follow-your-nose browsers, faceted-search interfaces and query builders. As a new way to interact with statistical Linked Data we define common OLAP operations on data cubes modelled in RDF and show how a nested set of OLAP operations lead to an OLAP query. Then, we show how to transform an OLAP query to a SPARQL query which generates all required facts from the data cube. Both metadata and OLAP queries are issued directly on a triple store; therefore, if the RDF is modified or updated, changes are propagated directly to OLAP clients.

Keywords: OLAP, query, operation, Linked Data, statistics, XBRL

1 Introduction

Linked Data provides easy access to large amounts of interesting statistics from many organizations for information integration and decision support, including financial information from institutions such as the UK government³ and the U.S. Securities and Exchange Commission.⁴ However, interaction paradigms for Linked Data such as follow-your-nose browsers, faceted-search interfaces, and query builders [10, 12] do not allow users to analyse large amounts of numerical data in an exploratory fashion of “overview first, zoom and filter, then details-on-demand” [24]. Online Analytical Processing (OLAP) operations on data cubes for viewing statistics from different angles and granularities, filtering for specific features, and comparing aggregated measures fulfil this information seeking mantra and provide interfaces for decision-support from statistics [2, 4, 25]. However OLAP on statistical Linked Data imposes two main challenges:

- OLAP requires a model of data cubes, dimensions, and measures. Automatically creating such a multidimensional schema from generic Linked Data is

³ <http://data.gov.uk/resources/coins>

⁴ <http://edgarwrap.ontologycentral.com/>

difficult, and only semi-automatic methods have proved applicable [16,18,21]. The RDF Data Cube vocabulary (QB)⁵ is a Linked Data vocabulary to model statistics in RDF. Several publishers have already used the vocabulary for statistical datasets.⁶

- OLAP queries are complex and require specialised data models, e.g., star schemas in relational databases, to be executed efficiently [9]. The typical architecture of an OLAP system consists of an ETL pipeline that extracts, transforms and loads data from the data sources into a data warehouse, e.g., a relational or multidimensional database. OLAP clients such as JPivot allow users to build OLAP queries and display results in pivot tables. An OLAP engine, e.g., Mondrian, transforms OLAP queries into queries to the data warehouse, and deploys mechanisms for fast data cube computation and selection, under the additional complexity that data in the data warehouse may change dynamically [8,14].

In this paper, we assume that statistical Linked Data has been modelled using QB and focus on efficiently executing OLAP queries on statistical Linked Data. In previous work [11] we have presented a proof-of-concept to interpret Linked Data reusing QB as a multidimensional model and to automatically load the data into a data warehouse used by common OLAP systems. The fact that OLAP queries are executed not on the RDF directly but by a common OLAP engine after automatically populating a data warehouse result in two drawbacks: first, although the relational star schema we adopted is a quasi-standard logical model for data warehouses, our approach requires a ROLAP engine to execute OLAP queries; second, if statistical Linked Data is updated, e.g., if a single new row is added, the entire ETL process has to be repeated, to have the changes propagated. Figure 1 shows our new data flow of having an OLAP engine issuing SPARQL queries directly to a triple store.

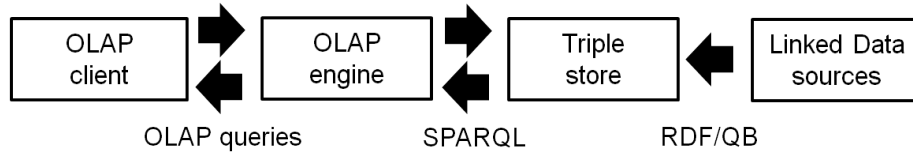


Fig. 1. Data flow for OLAP queries on statistical Linked Data in a triple store

The current work presents a new way to interact with statistical Linked Data:

- We define common OLAP operations on data cubes as Linked Data reusing QB and show how a nested set of OLAP operations lead to an OLAP query.
- We show how to transform an OLAP query to a SPARQL query which generates all required facts from the data cube.

⁵ <http://www.w3.org/TR/2012/WD-vocab-data-cube-20120405/>

⁶ <http://wiki.planet-data.eu/web/Datasets>

In the remainder of the paper, we first present a motivational scenario from the financial domain in Section 2. As a prerequisite for our contribution, in Section 3, we formally define a multidimensional model of data cubes based on QB. Then, in Section 4, we introduce OLAP operations on data cubes and present a direct mapping of OLAP to SPARQL queries. We apply this mapping in a small experiment in Section 5 and discuss some lessons learned in Section 6. In Section 7, we describe related work, after which, in Section 5, we conclude and describe future research.

2 Scenario: Analysing Financial Linked Data

In this section we describe a scenario of analysing Linked Data containing financial information. The Edgar Linked Data Wrapper⁷ provides access to XBRL filings⁸ as Linked Data reusing QB. Those filings disclose balance sheets of a large number of US organizations, for instance that *RAYONIER INC* had a *sales revenue net* of 377,515,000 USD from 2010-07-01 to 2010-09-30.⁹

Using LDSpider, we crawled Linked Data from the Edgar wrapper and stored a data cube *SecCubeGrossProfitMargin* into an Open Virtuoso triple store. The data cube contains single disclosures from financial companies such as *RAYONIER INC*. Each disclosure either discloses cost of goods sold (*CostOfGoodsSold*) or sales revenue net (*Sales*) as measures. The two measures have the unit USD and an aggregation function that returns the number of disclosures, or - if only one - the actual number. Any disclosure is fully dependent on the following dimensions: the disclosing company (*Issuer*), the date a disclosure started (*Dtstart*) and ended (*Dtend*) to be valid, and additional segment information (*Segment*).

In our scenario, a business analyst wants to compare the number of disclosures of cost of goods sold for two companies. He requests a pivot table with issuers *RAYONIER INC* and *WEYERHAEUSER CO* on the columns, and the possible periods for which disclosures are valid on the rows, and in the cells showing the number of disclosed cost of goods sold, or - if only one - the actual number. Figure 2 shows the needed pivot table.

3 A Multidimensional Model Based on QB

In this section, as a precondition for OLAP queries on Linked Data, we formally define the notion of data cubes in terms of QB. The definition is based on the multidimensional models of Gómez et al. [7], Pedersen et al. [20] and the Open

⁷ <http://edgarwrap.ontologycentral.com/>

⁸ <http://www.xbrl.org/Specification/XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2008-07-02.htm>

⁹ <http://edgarwrap.ontologycentral.com/archive/52827/0001193125-10-238973#ds>

Columns (issuer)		RAYONIER INC	WEYERHAEUSER CO
Rows (dtstart, dtend)			
2009-01-01	2009-3-31	1,100,335 USD	0 values
2009-04-01	2009-06-30	2 values	2,300,800 USD
...

Filters: CostOfGoodsSold

Fig. 2. Pivot table to be filled in our scenario

Java API for OLAP¹⁰ as well as on the Linked Data vocabularies of QB, SKOS¹¹ and skosclass.¹²

Definition 1 (Linked Data store with terms and triples). *The set of RDF terms in a triple store consists of the set of IRIs \mathcal{I} , the set of blank nodes \mathcal{B} and the set of literals \mathcal{L} . A triple $(s, p, o) \in \mathcal{T} = (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ is called an RDF triple, where s is the subject, p is the predicate and o is the object.*

Given a triple store with statistical Linked Data, we use basic SPARQL triple patterns on the store to help us defining sets of multidimensional elements. Given a multidimensional element x $iri(x) \in (\mathcal{I} \cup \mathcal{B})$ returns its IRI or blank node:

Member defines the set of members as $Member = \{?x \in (\mathcal{I} \cup \mathcal{B}) \mid (?x \text{ a } skos:Concept)\}$. Let $\mathcal{V} = 2^{Member}$, $V \in \mathcal{V}$, $ROLLUPMEMBER \subseteq Member \times Member$, $rollupmember(V) = \{(v_1, v_2) \in V \times V \mid (iri(v_1) \text{ skos:broaden } iri(v_2) \vee iri(v_2) \text{ skos:narrower } iri(v_1))\}$

Level defines the set of levels as $Level = \{(?x, V, rolluplevel(V)) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{V} \times ROLLUPMEMBER \mid (?x \text{ a } skosclass:ClassificationLevel \wedge \forall v \in V (iri(v) \text{ skos:member } ?x))\}$. Let $\mathcal{L} = 2^{Level}$, $L \in \mathcal{L}$, $ROLLUPLEVEL \subseteq Level \times Level$, $rolluplevel(L) = \{(l_1, l_2) \in L \times L \mid (iri(l_1) \text{ skosclass:depth } x) \wedge (iri(l_2) \text{ skosclass:depth } y) \wedge x \leq y)\}$

Hierarchy defines the set of hierarchies as $Hierarchy = \{(?x, L, rolluplevel(L)) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{L} \times ROLLUPLEVEL \mid (?x \text{ a } skos:ConceptScheme) \wedge \forall l \in L (iri(l) \text{ skos:inScheme } ?x)\}$. Let $\mathcal{H} = 2^{Hierarchy}$.

Dimension defines the set of dimensions as $Dimension = \{(?x, H) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{H} \mid (?x \text{ a } qb:DimensionProperty) \wedge \forall h \in H (?x \text{ qb:codeList } iri(h))\}$. Let $\mathcal{D} = 2^{Dimension}$.

¹⁰ <http://www.olap4j.org/>

¹¹ <http://www.w3.org/2004/02/skos/>

¹² http://www.w3.org/2011/gld/wiki/ISO_Extensions_to_SKOS

Measure defines the set of measures as $Measure = \{(?x, aggr) \in (\mathcal{I} \cup \mathcal{B}) \times \{UDF\} \mid (?x \text{ a } qb:MeasureProperty) \}$ with UDF a default aggregation function since QB so far does not provide a standard way to represent typical aggregation functions such as SUM, AVG and COUNT: if only one value is given, the value itself else the number of values is returned. Conceptually, measures are treated as members of a dimension-hierarchy-level combination labelled “Measures”. Let $\mathcal{M} = 2^{Measure}$.

DataCubeSchema defines the set of data cube schemas as $\{(?x, D, M) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{D} \times \mathcal{M} \mid (?x \text{ a } qb:DataStructureDefinition \wedge \forall d \in D (?x \text{ qb:componentProperty } ?comp \wedge ?comp \text{ qb:dimensionProperty } iri(d)) \wedge \forall m \in M (?x \text{ qb:componentProperty } ?comp \wedge ?comp \text{ qb:measureProperty } iri(m)))\}$.

Fact defines the set of possible statistical facts as $Fact = \{(?x, ?c_0, \dots, ?c_i, ?e_0, \dots, ?e_j) \in (\mathcal{I} \cup \mathcal{B}) \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) \times \dots \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) \times \mathcal{L} \times \dots \times \mathcal{L} \mid (?x \text{ a } qb:Observation) \wedge (?x ?d_0 c_0 \wedge ?d_0 \text{ a } qb:DimensionProperty) \wedge \dots \wedge (?x ?d_i c_i \wedge ?d_i \text{ a } qb:DimensionProperty) \wedge (?x ?m_0 e_0 \wedge ?m_0 \text{ a } qb:MeasureProperty) \wedge \dots \wedge (?x ?m_j e_j \wedge ?m_j \text{ a } qb:MeasureProperty)\}$. Let $\mathcal{F} = 2^{Fact}$.

DataCube defines the set of data cubes as $DataCube = \{(cs, F) \in DataCubeSchema \times \mathcal{F} \mid cs = (?x, D, M) \wedge D = \{D_0, \dots, D_{|D|}\} \wedge M = \{m_0, \dots, m_{|M|}\} \wedge \forall c \in F (c = (?obs, c_0, \dots, c_{|D|}, e_0, \dots, e_{|M|}) : (?obs \text{ qb:dataSet } ?ds \wedge ?ds \text{ qb:structure } ?x) \wedge \forall D_i \in D (?x iri(D_i) c_i \wedge iri(D_i) \text{ qb:codeList } ?h \wedge ?l \text{ skos:inScheme } ?h \wedge ?v \text{ skos:member } ?l \wedge (?v \text{ skos:notation } ?c_i \vee ?v \text{ skos:exactMatch } ?c_i)) \wedge \forall m_i \in M (?x iri(m_i) e_i \vee e_i = null))\}$.

We distinguish *metadata queries* and *OLAP queries* on data cubes. Whereas metadata queries return multidimensional objects such as the cube schema, the dimensions, and the measures, OLAP queries return facts directly contained in or derived from (e.g., by aggregation of several facts) the data cube.

Definition 1. According to Gray et al. [8] a materialised data cube (cs, F) with $cs = (x, D, M)$ contains a set of facts $MF = (?x, ?c_0, \dots, ?c_{|D|}, ?e_0, \dots, ?e_{|M|})$ with $?c_i \in C_i$ with $C_i : 0 \leq i \leq |D|$ all possible members of a dimension $D_i \in D$ or the special ALL value, and with $e_j \in T : 0 \leq j \leq |M|$ with T a numeric domain including the special null value in cases of cube sparsity. A data cube can be materialised by a union of $2^{|D|}$ sub-queries, each grouping the result on the members of a subset of dimensions, replacing the values of the other dimensions by a special ALL value, and computing each measure value by the measure’s aggregation function. The number of facts from a materialised data cube is given by $|MF| = (|C_0| + 1) \cdot \dots \cdot (|C_{|D|}| + 1)$.

Subqueries and aggregation functions in SPARQL 1.1 make easily possible the concept of Gray et al. [8] to fully materialise a data cube represented as Linked Data reusing QB. However, due to its exponential size w.r.t. the number of dimensions, such a SPARQL query is inefficient. OLAP queries may require only a small subset of all possible facts of a data cube; therefore, in the next section, we show how to evaluate OLAP queries using a single SPARQL query.

4 Mapping OLAP Operations to SPARQL on QB

In this section we show how to issue OLAP queries on a multidimensional model. We define common OLAP operations on single data cubes [19,20,22,23]. A nested set of OLAP operations lead to an OLAP query. We describe how to evaluate such an OLAP query using SPARQL on QB. Figure 3 illustrates the effect of common OLAP operations, with inputs and outputs.

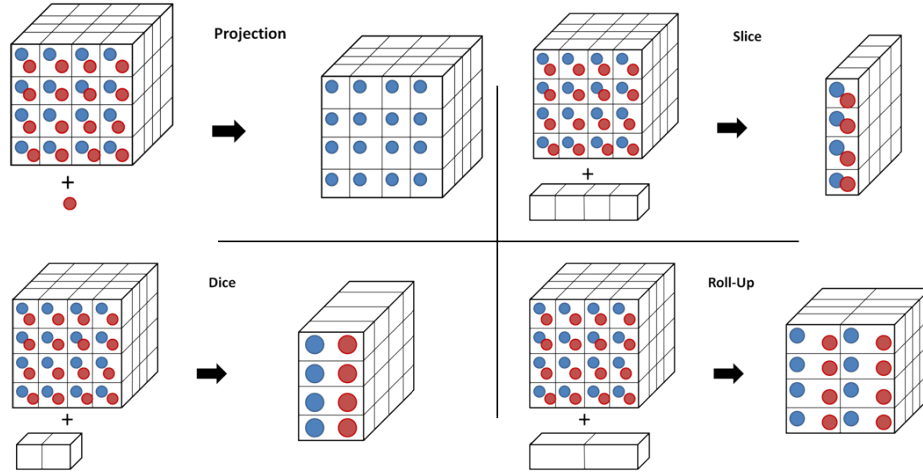


Fig. 3. Illustration of common OLAP operations with inputs and outputs (adapted from [23])

Note, this paper focuses on direct querying of single data cubes, the integration of several data cubes through *Drill-Across* or set-oriented operations such as union, intersection, difference is out-of-scope. Multiple datasets can already be queried together if they are covered by one *qb:DataStructureDefinition*.

Each OLAP operation has as input and output a data cube. Therefore, operations can be nested. A nested set of OLAP operations lead to an OLAP query. For interpreting a set of OLAP operations as an OLAP query and evaluate it using a SPARQL query on QB, we adopt the notion of *subcube queries* [13].

Definition 2. We define an OLAP query as a subcube query [13] on a certain cube (cs, C) with $cs = (?uri, D, M)$, represented as a tuple with as many elements as dimensions and measures: $(q_0, \dots, q_{|D|}, m_0, \dots, m_{|M|})$ with $\text{dom}(q_i) = \{?, ALL, x\}$ with $?$ for an inquired dimension, with ALL for an aggregated dimension, with x for one or more members to fix a dimension, and with m_i a measure to query for.

As examples, we describe three distinguishable subcube queries:

- A *full-cube query* returns exactly the tuples from a *DataCubeInstance* and inquires all dimensions: $(?, ?, ?, \dots, m_0, \dots, m_{|M|})$.

- A *point query* asks for a data cube comprising one specific instance tuple: $(a_0, a_1, \dots, a_{|D|}, m_0, \dots, m_{|M|})$ with a_i a member of a dimension D_i .
- A *fully-aggregated query* asks for the measures aggregated over all dimensions: $(ALL, ALL, \dots, ALL, m_0, \dots, m_{|M|})$.

In the following we describe how to evaluate each OLAP operation in terms of this query model and how a nested set of OLAP operations results in one specific OLAP subcube query. We assume that names and schemas of input and output data cubes are implicitly given and we focus on the data cube facts which will be queried for. An input data cube is represented as a full-cube query $(?, ?, ?, \dots, m_0, \dots, m_{|M|})$.

Projection is defined as $Projection : DataCube \times Measure \rightarrow DataCube$ and removes a measure from the input cube and allows to query only for specific measures. We evaluate *Projection* by removing a measure from the subcube query tuple.

Slice is defined as $Slice : DataCube \times Dimension \rightarrow DataCube$ and removes a dimension from the input cube and aggregates over the members of a dimension. We evaluate *Slice* by setting the tuple element of that dimension to *ALL*.

Dice is defined as $Dice : DataCube \times Dimension \times \mathcal{V} \rightarrow DataCube$ and allows to filter for and aggregate over certain dimension members. We evaluate *Dice* by setting the tuple element of that dimension to this particular member or set of members and aggregate over the set. Note, dice is not a selection operation but a combined filter and slice operation.

Roll-Up is defined as $Roll-Up : DataCube \times Dimension \times Level \rightarrow DataCube$ and allows to create a cube that contains instance data on a higher aggregation level. We evaluate roll-up by replacing the inquired members of a dimension with members of a higher level. Note, we do not define *Drill-Down*, since it can be seen as an inverse operation to *Roll-Up*.

As an example, consider an OLAP query on our *SecCubeGrossProfitMargin* cube for the cost of goods sold (CostOfGoodsSold) for each issuer and each date until when each disclosure is valid (dtend), filtering by disclosures from two specific segments. A nested set of OLAP operations that queries the requested facts can be composed as follows. In all our queries, we use prefixes to make URIs more readable:

```

Slice (
  Dice (
    Projection (
      edgar : SecCubeGrossProfitMargin ,
      edgar : CostOfGoodsSold ) ,
    edgar : segment ,
    { edgar : segmentAHealthCareInsuranceCompany ,
      edgar :
        segmentAResidentialRealEstateDeveloperMember } )
  ,

```

```
edgar:dtstart)
```

This query can then be represented as a subcube query with dimensions Issuer, Dtstart, Dtend, Segment:

```
(?, *, ?, {edgar:segmentAHealthCareInsuranceCompany,
edgar:segmentAResidentialRealEstateDeveloperMember},
edgar:CostOfGoodsSold)
```

Next, we describe how to evaluate such an OLAP query using a SPARQL query on QB. Since OLAP hierarchies add considerable complexity to QB and since a *Roll – Up* has a similar effect to a *Slice* operation, in this paper, we assume data cubes with only one hierarchy and level per dimension. A subcube query $Q = (q_0, \dots, q_{|D|}, m_0, \dots, m_{|M|})$ can be translated into a SPARQL query using the following steps:

1. We initialise the SPARQL query using the URI of the data cube. We query for all instance data from the data cube, i.e., observations linking to datasets which link to the data structure definition.
2. For each selected measure, we incorporate it in the SPARQL query by selecting additional variables for each measure and by aggregating them using the aggregation function of that measure, using OPTIONAL patterns in case we query for several measures.
3. For each inquired dimension, we query for all the instances of *skos:Concept* in a level of a hierarchy of the dimension and for the represented values used (from members either linked via *skos:notation* or *skos:exactMatch*) in the observations. We query for the observations showing property-value pairs for each of these variables. To display inquired dimensions in the result and correctly aggregating the measures, we *group by* each dimension variable.
4. For each fixed dimension, we filter for those observations that exhibit for each dimension one of the listed members.

We transform our example from above to the following SPARQL query. Note, “UDF” represents the standard aggregation function from our scenario:

```
select ?dimMem0 ?dimMem1 UDF(?measureValues0) where {
?obs qb:dataSet ?ds.
?ds qb:structure edgar:SecCubeGrossProfitMargin.
?obs edgar:issuer ?values0.
    ?dimMem0 skos:member ?level0.
    ?level0 skos:inScheme ?hierarchy0.
    edgar:issuer qb:codeList ?hierarchy0.
    ?dimMem0 skos:exactMatch ?values0.
?obs edgar:dtend ?values1.
    ?dimMem1 skos:member ?level1.
    ?level1 skos:inScheme ?hierarchy1.
    edgar:dtend qb:codeList ?hierarchy1.
    ?dimMem1 skos:notation ?values1.
```

```

?obs edgar:segment ?values2.
    ?slicerMem0 skos:member ?level2.
    ?level2 skos:inScheme ?hierarchy2.
    edgar:segment qb:codeList ?hierarchy2.
    ?slicerMem0 skos:exactMatch ?values2. Filter(?
        slicerMem0 = edgar:
        segmentAHealthCareInsuranceCompany
    OR ?slicerMem0 = edgar:
        segmentAResidentialRealEstateDeveloperMember)
?obs edgar:CostOfGoodsSold ?measureValue0.
} group by ?dimMem0 ?dimMem1

```

5 Experiment

In this section we demonstrate in a small experiment the applicability of our OLAP-to-SPARQL mapping to our scenario from the financial domain. The *SecCubeGrossProfitMargin* cube contains 17,448 disclosures that either disclose cost of goods sold or sales revenue net. The values of the measures fully depend on one of 625 different issuers, the date a disclosure started (27 members of dimension Dtstart) and ended (20 members of Dtend) to be valid, and additional information (21,227 members of Segment). The two measures have the unit USD and an aggregation function that returns the number of disclosures, or - if only one - the actual number. If fully materialised according to Definition 1, the cube contains $626 \cdot 28 \cdot 21 \cdot 21,228 = 7,813,772,064$ facts. To compute all of its facts, $2^4 = 16$ SPARQL subqueries would be needed.

OLAP interface allow users to interactively combine OLAP operations into an expression of an OLAP query language. Results of the query shall be visualised using a pivot table, a compact format to display multidimensional data [5]. As far as we know, MDX is the most widely used OLAP query language, adopted by OLAP engines such as Microsoft SQL Server, the Open Java API for OLAP, XML for Analysis (XMLA), and Mondrian. Therefore, we show that an MDX query can be transformed into an OLAP subcube query according to Definition 2 and evaluate the subcube query using a SPARQL query. The result is a subset of all possible facts from a data cube. The pivot table determines what dimensions to display on its columns and rows.

In order to answer the OLAP question of our scenario, we created the following MDX query. Multidimensional elements are described there using URIs.¹³ For an introduction to MDX, see its website.¹⁴ A more detailed description of how to transform an MDX query into an OLAP query due to space constraints we leave for future work when we evaluate our OLAP-to-SPARQL mapping more thoroughly.

¹³ Note URIs need to be translated to an MDX-compliant format that does not use reserved MDX-specific characters.

¹⁴ <http://msdn.microsoft.com/en-us/library/aa216770%28v=sql.80%29.aspx>

```

SELECT
{edgar:cik1417907idConcept , edgar:cik106535idConcept} ON
  COLUMNS,
CrossJoin(edgar:dtstartRootLevel.Members, edgar:
  dtendRootLevel.Members) ON ROWS
FROM [edgar:SecCubeGrossProfitMargin]
WHERE {edgar:CostOfGoodsSold}

```

A nested set of OLAP operations to compose our OLAP query is as follows:

```

Slice(Projection(
  edgar:SecCubeGrossProfitMargin ,
  edgar:CostOfGoodsSold) ,
edgar:segment)

```

This query can then be represented as a subcube query with dimensions Issuer, Dtstart, Dtend, Segment: (?, ?, *, *, *CostOfGoodsSold*). The resulting SPARQL query is as follows:

```

select ?dimMem0 ?dimMem1 ?dimMem2 count(xsd:decimal(?
  measureValue0)) sum(xsd:decimal(?measureValue0))
where {
?obs qb:dataSet ?ds.
?ds qb:structure edgar:SecCubeGrossProfitMargin.
?obs edgar:issuer ?values0.
  ?dimMem0 skos:member ?level0.
  ?level0 skos:inScheme ?hierarchy0.
  edgar:issuer qb:codeList ?hierarchy0.
  ?dimMem0 skos:exactMatch ?values0.
?obs edgar:dtstart ?values1.
  ?dimMem1 skos:member ?level1.
  ?level1 skos:inScheme ?hierarchy1.
  edgar:dtstart qb:codeList ?hierarchy1.
  ?dimMem1 skos:notation ?values1.
?obs edgar:dtend ?values2.
  ?dimMem2 skos:member ?level2.
  ?level2 skos:inScheme ?hierarchy2.
  edgar:dtend qb:codeList ?hierarchy2.
  ?dimMem2 skos:notation ?values2.
?obs edgar:CostOfGoodsSold ?measureValue0.
} group by ?dimMem0 ?dimMem1 ?dimMem2

```

The aggregation function used is a non-standard one, therefore, we had to compute the SUM and COUNT for the measure. We run the query after a reboot of the triple store. The query took 18sec and returned 58 facts to be filled into the requested pivot table. The number of 7,813,772,064 potential facts in the cube does not have a strong influence on the query since the cube is very sparse, for instance, the triple store contains observations only for a fraction of segment members.

6 Discussion

Data from the data cube is queried on demand, and no materialization is done. We correctly aggregate data on one specific granularity, defined by the mentioned inquired and fixed dimensions. Dimensions that are not mentioned will be automatically handled as having an *ALL* value [8], representing all possible values of the dimension. The aggregation results in correct calculations, since we assume only one hierarchy-level per dimension in this work. Only if observations would be defined on different granularities, e.g., gender *male*, *female*, and *total*, aggregating over them would result in incorrect numbers.

Filling the pivot table with measure values from the SPARQL result requires matching of the member values for each fact for the following reasons: first, if the data cube is sparse, i.e., not for every possible combination of members a value is given, then, for non-occurring combinations the SPARQL query does not return a value; second, all member combinations of inquired dimensions are calculated, even though only specific combinations might be selected, as in the case of the two issuers in the OLAP query of our scenario. Indexing of either the pivot table or the SPARQL result table may allow faster population of the pivot table.

In summary, though our OLAP algebra to SPARQL mapping may not result in the most efficient SPARQL query and require additional efforts for populating the pivot table, it correctly computes all required facts from the data cube without the need for explicitly introducing the non-relational *ALL* member or using sub-queries [8].

7 Related Work

Kobilarov and Dickinson [12] have combined browsing, faceted-search, and query-building capabilities for more powerful Linked Data exploration, similar to OLAP, but not focusing on statistical data. Though years have passed since then, current literature on Linked Data interaction paradigms does not seem to expand on analysing large amounts of statistics.

OLAP query processing in general has long been a topic of research [27]. OLAP operations have been defined on a logical level [1, 20, 26] or on a conceptual level [3, 22, 25]. Execution of OLAP operations mainly is concerned with the computation of the data cube and with storing parts of the results of that computation to efficiently return the results, to require few disk or memory space, and to remain easy to update if data sources change [14]. Approaches mainly depend on the type of data structure on which to perform the computations and in which to store the results. Data structures can roughly be grouped into ROLAP, using relational tables and star or snowflake schemas, and MOLAP, using multidimensional arrays for direct storing and querying of data cubes.

Specific approaches regarding OLAP on Linked Data seem to have concentrated so far on multidimensional modelling from ontologies [6, 15–17]. For instance, Nebot et al. [16] recognise the potential of OLAP to analyse RDF data,

but do not provide a dedicated query engine and require a multidimensional database that needs to be updated if RDF data changes. Entity-centric object databases [20] show some resemblance to OLAP querying, however, have so far not been applied to Linked Data.

In this work we use the graph-based RDF data model for querying and storing of multidimensional data reusing QB. Here, both schema information and actual data is accessed using Linked Data principles and managed using an RDF store. Our approach focuses on OLAP queries that can be composed by common OLAP operations and can be represented as a subcube query. Our mapping allows translating OLAP queries into one SPARQL query to be run on the RDF without storage of intermediate results. In our small experiment the produced SPARQL query showed sufficiently fast, but queries are expected to become insufficient for larger datasets. Since no materialization is done, only few extra space is required for a hashmap to fill the pivot table with the SPARQL result set, and updates to the RDF are propagated directly to OLAP clients. Although there may be more efficient querying approaches such as special indexing and caching, to the best of our knowledge, this is the first work on computing and querying of data cubes represented in RDF.

8 Conclusions and Future Work

We have presented an approach to interact with statistical Linked Data using common Online Analytical Processing operations of “overview first, zoom and filter, then details-on-demand”. For that, we define common OLAP operations on single data cubes in RDF reusing the RDF Data Cube vocabulary, map nested sets of OLAP operations to OLAP subcube queries, and evaluate those OLAP queries using SPARQL. Both metadata and OLAP queries are issued directly on a triple store; therefore, if the RDF is modified or updated, changes are propagated directly to OLAP clients. Though, our OLAP-to-SPARQL mapping may not result in the most efficient SPARQL query and require additional effort in populating resulting pivot tables, we correctly calculate requested facts of a data cube without the need for explicitly introducing the non-relational *ALL* member or using subqueries.

Future work may be conducted in three areas: 1) extending our current approach with OLAP hierarchies and Drill-Across queries; 2) implementing an OLAP engine to more thoroughly evaluate our current approach and to investigate more efficient OLAP query execution plans; 3) investigating possible OLAP clients that map OLAP operations to intuitive user interactions.

Acknowledgements

This work was supported by the German Ministry of Education and Research (BMBF) within the SMART project (Ref. 02WM0800) and the European Community’s Seventh Framework Programme FP7/2007-2013 (PlanetData, Grant 257641).

References

1. Agrawal, R., Gupta, A., Sarawagi, S.: Modeling Multidimensional Databases. In: Proc. of the Thirteenth International Conference on Data Engineering (1997)
2. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. *ACM SIGMOD Record* **26** (1997) 65–74
3. Chen, L., Ramakrishnan, R., Barford, P., Chen, B., Yegneswaran, V.: Composite Subset Measures. In: Proc. of the 32nd International Conference on Very Large Databases (2006)
4. Codd, E.F., Codd, S.B., Salley, C.T.: Providing OLAP to User-Analysts: An IT Mandate. (1993)
5. Cunningham, C., Galindo-Legaria, C.A., Graefe, G.: PIVOT and UNPIVOT: optimization and execution strategies in an RDBMS. In: Proc. of the Thirtieth International Conference on Very Large Databases (2004)
6. Diamantini, C., Potena, D.: Semantic enrichment of strategic datacubes. In: Proc. of the ACM 11th international workshop on Data warehousing and OLAP (2008)
7. Gómez, L.I., Gómez, S.A., Vaisman, A.A.: A Generic Data Model and Query Language for Spatiotemporal OLAP Cube Analysis Categories and Subject Descriptors. In: Proc. of EDBT 2012
8. Gray, J., Bosworth, A., Lyaman, A., Pirahesh, H.: Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. In: Proc. of the Twelfth International Conference on Data Engineering (1995) 152–159
9. Harinarayan, V., Rajaraman, A.: Implementing data cubes efficiently. *ACM SIGMOD Record* (1996)
10. Harth, A.: VisiNav: A system for visual search and navigation on web data. *Journal of Web Semantics* **8**(4) (2010) 348–354
11. Kämpgen, B., Harth, A.: Transforming Statistical Linked Data for Use in OLAP Systems. In: Proc. of I-Semantics 2011
12. Kobilarov, G., Dickinson, I.: Humboldt: Exploring Linked Data. In: Proc. of Linked Data on the Web Workshop (LDOW 2008) at WWW 2008
13. Li, X., Han, J., Gonzalez, H.: High-dimensional OLAP: a minimal cubing approach. In: Proc. of the Thirtieth International Conference on Very Large Databases (2004)
14. Morfonios, K., Konakas, S., Ioannidis, Y., Kotsis, N.: ROLAP implementations of the data cube. *ACM Computing Surveys* **39** (2007) 12–es
15. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. *Decision Support Systems* **52** (2012) 853–868
16. Nebot, V., Berlanga, R., Pérez, J.M., Aramburu, M.J., Vej, S.L.: Multidimensional Integrated Ontologies : A Framework for Designing Semantic Data Warehouses. *Journal on Data Semantics* (2009) 1–36
17. Niinimäki, M., Niemi, T.: An ETL Process for OLAP Using RDF/OWL Ontologies. *Journal on Data Semantics XIII* **5530** (2009) 97–119
18. Pardillo, J., Mazón, J.-N.: Using Ontologies for the Design of Data Warehouses. *International Journal of Database Management Systems* **3** (2011) 73–87
19. Pardillo, J., Mazón, J.-N., Trujillo, J.: Bridging the semantic gap in OLAP models: platform-independent queries. In: Proc. of the ACM 11th international workshop on Data warehousing and OLAP (2008)
20. Pedersen, T.B., Gu, J., Shoshani, A., Jensen, C.S.: Object-extended OLAP querying. *Data Knowl. Eng.* **68** (2009) 453–480
21. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: Proc. of the ACM tenth international workshop on Data warehousing and OLAP (2007)

22. Romero, O., Abelló, A.: On the Need of a Reference Algebra for OLAP. In: Proc. of DaWaK 2007
23. Romero, O., Marcel, P., Abelló, A., Peralta, V., Bellatreche, L.: Describing analytical sessions using a multidimensional algebra. In: Proc. of the 13th international conference on Data warehousing and knowledge discovery (2011)
24. Shneiderman, B.: The Eyes Have It : A Task by Data Type Taxonomy for Information Visualizations. *Information Visualization* (1996) 336–343
25. Trujillo, J.: Bridging the Semantic Gap in OLAP Models : Platform-independent Queries Categories and Subject Descriptors. *Computing Systems* (2008) 89–96
26. Vassiliadis, P.: Modeling Multidimensional Databases, Cubes and Cube Operations. In: Proc. of the 10th International Conference on Scientific and Statistical Database Management (1998)
27. Vassiliadis, P., Sellis, T.: A survey of logical models for OLAP databases. *ACM Sigmod Record* **28** (1999) 64–69

SPARTIQUULATION: Verbalizing SPARQL queries

Basil Ell, Denny Vrandečić, and Elena Simperl

KIT, Karlsruhe, Germany

{basil.ell,denny.vrandecic,elena.simperl}@kit.edu

Abstract. Much research has been done to combine the fields of Databases and Natural Language Processing. While many works focus on the problem of deriving a structured query for a given natural language question, the problem of query verbalization – translating a structured query into natural language – is less explored. In this work we describe our approach to verbalizing SPARQL queries in order to create natural language expressions that are readable and understandable by the human day-to-day user. These expressions are helpful when having search engines generate SPARQL queries for user-provided natural language questions or keywords and enable the user to check whether the right question has been understood. While our approach enables verbalization of only a subset of SPARQL 1.1, this subset applies to 85% of the 209 queries in our training set. These observations are based on a corpus of SPARQL queries consisting of datasets from the QALD-1 challenge and the ILD2012 challenge.

Keywords: SPARQL, natural language generation, verbalization

1 Introduction

Much research has been done to combine the fields of Databases and Natural Language Processing to provide natural language interfaces to database systems [22]. While many works focus on the problem of deriving a structured query for a given natural language question or a set of keywords [10, 21, 27], the problem of query verbalization – translating a structured query into natural language – is less explored. In this work we describe our approach to verbalizing SPARQL queries in order to create natural language expressions that are readable and understandable by the human day-to-day user. The verbalized form of the generated query is helpful for users since it allows them to understand how the results have been retrieved and whether the right question has been asked to the queried knowledge base.

In this paper we describe the current state of our SPARTIQUULATION system¹ which allows verbalization of a subset of SPARQL 1.1 SELECT queries.

¹ The name is derived from joining *SPARQL* and *articulation*.

The remainder of this paper is structured as follows. Section 2 gives an overview of our query verbalization approach in terms of our system’s architecture and the tasks that it performs. Section 3 relates it to existing work, and in Section 4 conclusions are drawn and an outlook is provided.

2 Query Verbalization Approach

2.1 Introduction

Our approach is inspired by the pipeline architecture for natural language generation (NLG) systems and the set of seven tasks performed by such systems as introduced by Reiter and Dale [19]. The input to such a system can be described by a four-tuple (k, c, u, d) – where k is a knowledge source (not to be confused with the knowledge base a query is queried against), c the communicative goal, u the user model, and d the discourse history. Since we neither perform user-specific verbalization nor do we perform the verbalization in a dialog-based environment, we omit both the user model and the discourse history. The communicative goal is to communicate the meaning of a given SPARQL query q . However, there are multiple options. Three basic types of linguistic expressions can be used: i) statements that describe the search results where this description is based on the query only and not on the actual results returned by a SPARQL endpoint (e.g. *Bavarian entertainers and where they are born*), ii) a question can be formulated about the existence of knowledge of a specified or unspecified agent (e.g. *Which Bavarian entertainers are known and where are they born?*), and iii) a query can be formulated as a command (e.g. *Show me Bavarian entertainers and where they are born*). In this work we decided to explore how to verbalize queries as statements. Therefore, the communicative goal is to verbalize a query as a statement – more precisely in English.

2.2 Components and Tasks

In this section we present our approach along the seven tasks involved in NLG according to Reiter and Dale [19]. This work is the first step towards the verbalization of SPARQL queries. So far we put a focus on *document structuring*, but not on *lexicalization*, *aggregation*, *referring expression generation*, *linguistic realisation*, and *structure realisation*.

The pipeline architecture is depicted in Figure 1. Within the Document Planner the content determination process creates messages and the document structuring process combines them into a document plan (DP) which is the output of this component and the input to the Microplanner component. Within the Microplanner the processes lexicalization, referring expression generation and aggregation take place, which results in a text specification (TS) that is made up of phrase specifications. The Surface Realizer then uses this text specification to create the output text.

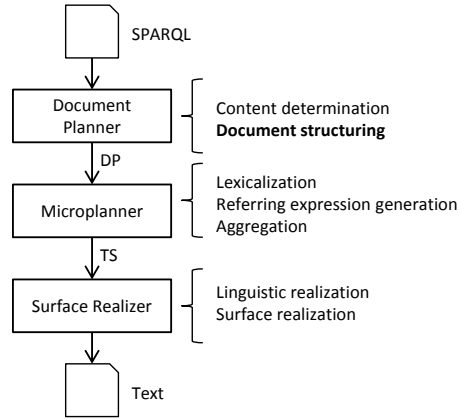


Fig. 1. Pipeline architecture of our NLG system

Content determination is the task to decide which information to communicate in the text. In the current implementation we decided not to leave this decision to the system. What is communicated is the meaning of the input query without communicating which vocabularies are used to express the query. Therefore, in this task no action is performed.

Document structuring is the task to construct messages from the input query and to decide for their order and structure. These messages are used for representing information in the domain, such as the class to which the selected entities belong to or the number to which the result set is limited. We present the set of message types after introducing the notion of the main entity and the graph transformation. Our observations are based on a corpus of SPARQL queries consisting of datasets from the QALD-1 challenge² and the ILD2012 challenge.³ The full dataset contains 263⁴ SPARQL SELECT queries and associated manually created questions. In order to leave parts of this dataset for future evaluation we only regarded 80% of each dataset as training data. Since in our approach we cannot yet handle all features of the SPARQL 1.1 standard, we had to exclude some queries. Within this training set of 209 queries we excluded the queries with the features UNION (22), GROUPBY (7), and those where the triple patterns within the WHERE clause do not form a connected graph (3). This means that this subset covers 85% of the queries within the training set.

We perform a transformation of the query graph, since it reduces the number of necessary message types which are shown in Table 1. Thus it simplifies the verbalization. This transformation is based on the observation that in most queries one variable can be identified that is rendered as the subject of a sen-

² <http://www.sc.cit-ec.uni-bielefeld.de/qald-1>

³ <http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/>

⁴ For nine questions no query is given since they are out of scope regarding the datasets provided for the challenge. 28 queries are ASK queries.

tence. For example when querying for mountains (bound to variable `?mountain`) and their elevations (bound to variable `?elevation`), then `?mountain` is verbalized as the subject of the verbalization `mountains and their elevations`. We refer to this variable as the *main entity* of a query. However, for some queries no such element exists. Consider for example the query `SELECT * WHERE { ?a dbpedia:marriedTo ?b . }`. Here a tuple is selected and in a possible verbalization *Tuples of married entities*⁵ no single variable appears represented as a subject. In order to identify the main entity we define Algorithm 1 that applies the ordered list of rules shown in Figure 2. These rules propose the exclusion of members from a candidate set. We derived them by looking at queries within the training set having multiple candidates. The candidate set C is initialized with variables that appear in the select clause and the algorithm eliminates candidates step by step. Q denotes the set of triples within the WHERE clause of a query, R_t is the property `rdf:type` and R_l is a labeling property from the set of 36 labeling properties identified by [6]. The application of an exclusion rule R_i to a candidate set C , denoted by $R_i(C)$, results in the removal of the set E proposed by the reduction rule.

Rule 1 $E := \{x \in C \mid "x \text{ appears in OPTIONAL only"}\}$
Rule 2 $E := \{z \in C \mid \neg \exists(z, R_t, u) \in Q\}$
 if $\exists c_1 \in C : \neg \exists(c_1, R_t, x) \in Q \wedge \exists c_2 \in C : \neg \exists(c_2, R_t, y) \in Q$
Rule 3 $E := \{z \in C \mid \neg \exists(z, R_l, u) \in Q\}$
 if $\exists c_1 \in C : \neg \exists(c_1, R_l, x) \in Q \wedge \exists c_2 \in C : \neg \exists(c_2, R_l, y) \in Q$

Fig. 2. Exclusion rules

The rules can be described as follows where the numbers show how often a rule was successful in reducing the candidate set for the 209 queries within our training set. *Rule 1* (85, 40.67%) proposes removing candidates that appear within the WHERE clause only within OPTIONAL blocks. *Rule 2* (12, 5.74%) proposes removing candidates that represent subjects that are not constrained via `rdf:type` in the case that there are candidates that are constrained via `rdf:type`. *Rule 3* (48, 22.97%) proposes removing candidates for which no label is constrained or requested in the case that there are candidates for which this is the case. In some cases (64, 30.62%) no rule was applied since the candidate set contained only a single variable. For all queries given these rules the main entity has been identified. While our actual list of exclusion rules contained more rules these were never applied for the given training data and thus omitted here.

We transform, as shown in Algorithm 2, queries in a way that the query graph is converted into a graph where the main entity is the root and all edges point away from the root if the query does not come in that shape already. Therefore the algorithm maintains three sets of edges: edges that are already processed (P), edges that need to be followed (F), and edges that need to be transformed

⁵ DBpedia provides no `rdfs:domain` and `rdfs:range` information, such as `foaf:Person` for this property. Therefore here we give a generic verbalization to demonstrate the problem.

Algorithm 1 Applying reduction rules to candidate set.

```

if  $|C| = 1$  then
  return  $C$ 
while  $|C| > 1$  do
  for all  $R_i \in R$  do
    if  $|R_i(C)| > 0$  then
       $C \leftarrow R_i(C)$ 
      if  $|C| = 1$  then
        return  $C$ 
return  $\emptyset$ 

```

(T) which means reversed. An edge is reversed by exchanging subject and object and by marking the property (p) as being reversed (p^r).

Algorithm 2 Graph transformation

```

 $P \leftarrow \emptyset, F \leftarrow \{(s, p, o) \in Q \mid s = m\}, T \leftarrow \{(s, p, o) \in Q \mid o = m\}$  (init)
while  $F \neq \emptyset$  or  $T \neq \emptyset$  do
  for all  $(s_i, p_i, o_i) \in F$  do
    for all  $(s_j, p_j, o_j) \in Q \setminus (P \cup F \cup T)$  do
      if  $o_i = s_j$  then
         $F \leftarrow F \cup \{(s_j, p_j, o_j)\}$ 
      else if  $o_i = o_j$  then
         $T \leftarrow T \cup \{(s_j, p_j, o_j)\}$ 
    Move  $(s_i, p_i, o_i)$  from  $F$  to  $P$ 
  for all  $(s_i, p_i, o_i) \in T$  do
    for all  $(s_j, p_j, o_j) \in Q \setminus (P \cup F \cup T)$  do
      if  $s_i = s_j$  then
         $F \leftarrow F \cup \{(s_j, p_j, o_j)\}$ 
      else if  $s_i = o_j$  then
         $T \leftarrow T \cup \{(s_j, p_j, o_j)\}$ 
     $T \leftarrow T \setminus \{(s_i, p_i, o_i)\}$ 
     $P \leftarrow P \cup \{(o_i, p_i^r, s_i)\}$ 
return  $P$ 

```

We identified the set of 14 message types (MT), shown in Table 1 that allow us to represent the 209 queries from our training set. The first 9 MTs represent directed paths in the query graph which means that for each directed path that begins at the main entity, we represent this path with a message. Each path starts at the main entity (M) and consists of none to many pairs ($((RV)^*$) of a resource (R) followed by a variable (V). Moreover, they may contain a labeling property (R_l) or the `rdf:type` property (R_t). *VAR* represent all information about a variable, such as its name, whether it is the main entity, whether it is selected, distinct, optional, counted, or whether any filter is specified for this

variable. The MTs *ORDERBY*, *LIMIT*, *OFFSET* and *HAVING* represent the respective SPARQL features.

The document plan (DP), which is the output of the Document Planner and input to the Microplanner, structures the content as follows: in the first part, which can later be verbalized as one or more sentences, the main entity and its constraints are described, followed by a description of the requests (the variables besides the main entity that appear in the select clause) and their constraints. In a second part, if available and not already communicated in the first part, the selection modifiers are verbalized. According to these 3 categories – abbreviated with *cons*, *req*, and *mod* – we classify the MTs as follows. The MTs (1), (2), (4), (6), (7), and (9) from Table 1 belong to the class *cons*, the MTs (3), (5), and (8) belong to the class *req*. MTs (1), (2), (4), (6), (7) and (9) may also belong to class *req* if they contain a variable besides the main entity that appears in the select clause. MTs (10) – (14) belong to the class *mod*. While this set of message types is sufficient for the given training set, which means that all queries can be represented using these message types, we extended this list with 7⁶ more types in order to be prepared for queries such as `SELECT ?s ?p ?o WHERE { ?s ?p ?o. }` and `SELECT ?p WHERE { ?s ?p ?o. }` where instead of generating text, canned text is used, such as *All triples in the database* and *Properties used in the database*.

nr name	nr name	nr name
(1) $M(RV)^*RR$	(2) $M(RV)^*RL$	(3) $M(RV)^*RV$
(4) $M(RV)^*R_lR$	(5) $M(RV)^*R_lV$	(6) $M(RV)^*R_lL$
(7) $M(RV)^*R_tR$	(8) $M(RV)^*R_tV$	(9) $M(RV)^*R_tL$
(10) <i>VAR</i>	(11) <i>ORDERBY</i>	(12) <i>LIMIT</i>
(13) <i>OFFSET</i>	(14) <i>HAVING</i>	

Table 1. Message types

As an example the SPARQL query in Listing 1 is represented using the 6 messages shown in Figure 3. Note that due to the graph transformation the property `onto:author` is reversed which is denoted by `rev: yes` within the data structure stored in the messages. This query can be verbalized as: *Authors of books with English name "The Pillars of the Earth" and if available their English names*. Note that plural (*authors*, *books* and *names* instead of *author*, *book*, and *name*) is used per default. The filter for English labels is stored within the message representing the variable `string`.

⁶ Given that all three variables can either be selected or not selected and at least one variable needs to be selected, this results in 7 combinations.

```

SELECT ?uri ?string WHERE {
  ?books rdf:type onto:Book .
  ?books onto:author ?uri .
  ?books rdfs:label "The Pillars of the Earth"@en .
  OPTIONAL {
    ?uri rdfs:label ?string .
    FILTER (lang(?string) = 'en')
  }
}

```

Listing 1. Who wrote the book The pillars of the Earth? – SPARQL query

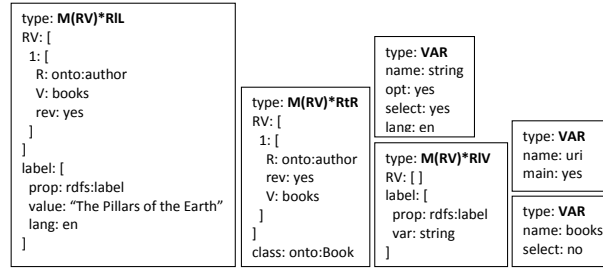


Fig. 3. Messages for the SPARQL query in Listing 1.

Lexicalization is the task of deciding what specific words to use for expressing the content. For each entity we dereference its URI and in case that RDF data is returned, we check if an English label is provided using one of the 36 labeling properties defined in [6]. Otherwise, we derive a label from the URI's local name using the patterns introduced by Hewlett et al. in [11].

Referring expression generation is the task of deciding how to refer to an entity. Considering the example *Entertainers born in Bavaria and where they are born*. Here, *they* is the expression that refers to the Bavarian entertainers.

Aggregation is the task to decide how to map structures created within the document planner onto linguistic structures such as sentences and paragraphs. For messages of type *cons* and *req* sentence parts are created that are joined into a single sentence. Messages of type *mod* are verbalized in further sentences. Aggregation is indispensable for concise verbalization. Since we split a query graph into (overlapping) paths where each path is represented by a message, aggregation would exploit these overlappings.

Linguistic realization is the task of converting abstract representations of sentences into real text. Text parts are generated for each of the message types (1) – (9) from Table 1. For each such type a rule is invoked that produces a sentence fragment, for example for the MT $MRVR_lL$ – which is an instance of the MT $M(RV)^*R_lL$ – the rule `article(lex(prop1)) + lex(prop1) + L` produces for two triples `?uri dbpedia:producer ?producer` and `?producer rdfs:label "Hal Roach"` the text `a producer Hal Roach`. The function `article` chooses an appropriate article (`a` or `an`) depending on the lexicalization `lex(prop1)` of the property. This fragment is added to the part of the verbalization where the constraints for the main entity are described and may be joined by the word `and` with other constraint fragments.

Structure realization is the task to add markup such as HTML code to the generated text in order to be interpreted by the presentation system, such as a web browser. While this could be helpful to enhance the readability of a complex verbalization, which is the case in [2], we do not currently exploit this opportunity.

3 Related Work

While to the best of our knowledge no work is published on the verbalization of SPARQL queries, related work comes from three areas: verbalization of RDF data [5, 16, 24, 25, 29], verbalization of OWL ontologies [1, 3, 4, 7–9, 11, 12, 14, 20, 23, 26, 28], and verbalization of SQL queries [13, 17, 18]. Although the first two fields provide techniques that we can apply to improve the lexicalization and aggregation tasks, such as the template-based approach presented in [5], the document structuring task, on which we focus here, is rarely explored. Compared to the SQL verbalization work by Minock [17, 18], where they focus on tuple relational queries, our problem of verbalizing SPARQL queries is different in the sense that we strive for having a generic approach that can be applied to any datasource without being tied to any schema. Patterns need to be manually created to cover all possible combinations for each relation in the schema whereas in our work we defined a set of message types that are schema-agnostic. Koutrika et al. [13] annotate query graphs with template labels and explore multiple graph traversal strategies. Moreover, they identify a main entity (the *query subject*), perform graph traversal starting from that entity, and distinguish between *cons* (*subject qualifications*) and *req* (*information*).

4 Conclusions and Outlook

For the task of verbalizing SPARQL queries we focused on a subset of the SPARQL 1.1 standard which covers 88% of the queries in a corpus of 209 SPARQL queries. Evaluation will have to show the representativeness of this

corpus compared to real-life queries and the qualities of the verbalizations generated using our SPARTIQUULATION system. While in our architecture 6 tasks are needed to generate verbalizations, our main focus has been the task of *document structuring* which we described in this work. In order to realize the full verbalization pipeline, 5 other tasks need to be explored in future work. Since the current approach is mostly schema-agnostic – only terms from the RDFS vocabulary are regarded during document structuring – we believe that this approach is generic in terms of being applicable to queries for RDF datasources using any vocabularies. However, in the future the tasks of lexicalization can be improved by regarding schemas such as FOAF since persons are treated differently in verbalizations than non-persons, genders can be regarded etc. Having message types designed for specific vocabularies allows to tailor the verbalization to a specific use case and may lead to more concise verbalizations. In the current implementation message types are hard-coded thus limiting the flexibility of the approach. Having the possibility to load a set of message types into the system would add the possibility to integrate automatically learned or application-specific message types.

Acknowledgements

The work presented in this paper is supported by the European Union's 7th Framework Programme (FP7/2007-2013) under Grant Agreement 257790.

References

1. Aguado, G., Bañón, A., Bateman, J. A., Bernardos, S., Fernández, M., Gómez-Pérez, A., Nieto, E., Olalla, A., Plaza, R., Sánchez, A.: ONTOGENERATION: Reusing domain and linguistic ontologies for Spanish text generation. In: Proc. of the Workshop on Applications of Ontologies and Problem Solving Methods, ECAI 1998
2. Bontcheva, K.: Generating tailored textual summaries from ontologies. In: Proc. of ESWC 2005 531–545
3. Bontcheva, K., Wilks, Y.: Automatic Report Generation from Ontologies: the MI-AKT approach. In: Proc. of NLDB 2004
4. Cregan, A., Schwitter, R., Meyer, T.: Sydney OWL Syntax - towards a Controlled Natural Language Syntax for OWL 1.1. In: Proc. of OWLED 2007
5. Davis, B., Iqbal, A., Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Handschuh, S.: RoundTrip Ontology Authoring. In: Proc. of The Semantic Web, ISWC 2008 50–65
6. Ell, B., Vrandečić, D., Simperl, E.: Labels in the Web of Data. In: Proc. of ISWC 2011
7. Fliedl, G., Kop, C., Vöhringer, J.: Guideline based evaluation and verbalization of OWL class and property labels. *Data Knowl. Eng.* **69**(4) (2010) 331–342
8. Galanis, D., Androutsopoulos, I.: Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system. In: Proc. of the Eleventh European Workshop on Natural Language Generation (2007) 143–146

9. Gareva-Takasmanov, L., Sakellariou, I.: OWL for the Masses: From Structured OWL to Unstructured Technically-Neutral Natural Language. In: Proc. of BCI 2009 260–265
10. Haase, P., Herzig, D., Musen, M., Tran, D. T.: Semantic Wiki Search. In: Proc. of ESWC 2009 445–460
11. Hewlett, D., Kalyanpur, A., Kolovski, V., Halaschek-Wiener, C.: Effective NL Paraphrasing of Ontologies on the Semantic Web. In: Proc. of the End User Semantic Web Interaction Workshop at the 4th International Semantic Web Conference (2005)
12. Kaljurand, K., Fuchs, N. E.: Verbalizing OWL in Attempto Controlled English. In: Proc. of OLWED 2007
13. Koutrika, G., Simitsis, A., Ioannidis, Y. E.: Explaining Structured Queries in Natural Language. In: Proc. of ICDE’10 (2010)
14. Liang, S. F., Stevens, R., Rector, A.: OntoVerbal-M: a Multilingual Verbaliser for SNOMED CT. In: Proc. of Multilingual Semantic Web (2011)
15. McKay, B. D.: Practical graph isomorphism. *Congressus Numerantium* **30** (1981) 45–87
16. Mellish, C., Sun, X.: The semantic web as a Linguistic resource: Opportunities for natural language generation. *Knowl.-Based Syst.* **19**(5) (2006) 298–303
17. Minock, M.: A Phrasal Approach to Natural Language Interfaces over Databases. In: Proc. of NLDB 2005 333–336
18. Minock, M.: C-Phrase: A system for building robust natural language interfaces to databases. *Data Knowl. Eng.* **69**(3) (2010) 290–302
19. Reiter, E., Dale, R.: *Building Natural Language Generation Systems*. Cambridge University Press (2000)
20. Schütte, N.: Generating natural language descriptions of ontology concepts. In: Proc. of the 12th European Workshop on Natural Language Generation (2009) 106–109
21. Shekarpour, S., Auer, S., Ngonga Ngomo, A.-C., Gerber, D., Hellmann, S., Stadler, C.: Keyword-driven SPARQL Query Generation Leveraging Background Knowledge. In: Proc. of International Conference on Web Intelligence (2011)
22. Simitsis, A., Ioannidis, Y. E.: DBMSs Should Talk Back Too. In: Proc. of CIDR 2009
23. Stevens, R., Malone, J., Williams, S., Power, R.: Automating class definitions from OWL to English. In: Proc. of Bio-Ontologies 2010: Semantic Applications in Life Sciences SIG at the 18th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB 2010)
24. Sun, X., Mellish, C.: Domain Independent Sentence Generation from RDF Representations for the Semantic Web. In: Proc. of Combined Workshop on Language-Enabled Educational Technology and Development and Evaluation of Robust Spoken Dialogue Systems, ECAI 2006
25. Sun, X., Mellish, C.: An experiment on ”free generation” from single RDF triples. In: Proc. of the Eleventh European Workshop on Natural Language Generation (2007) 105–108
26. Third, A., Williams, S., Power, R.: OWL to English: a tool for generating organised easily-navigated hypertexts from ontologies. In: Proc. of ISWC 2011
27. Tran, D. T., Wang, H., Haase, P.: Hermes: Data Web search on a pay-as-you-go integration infrastructure. *Journal of Web Semantics* **7**(3) (2009)
28. Wilcock, G.: Talking OWLs: Towards an Ontology Verbalizer. In: Proc. of Human Language Technology for the Semantic Web and Web Services, ISWC 2003 109–112

29. Wilcock, G., Jokinen, K.: Generating Responses and Explanations from RDF/XML and DAML+OIL. In: Proc. of IJCAI 2003

Improving Semantic Search Using Query Log Analysis^{*}

Khadija Elbedweihiy, Stuart N. Wrigley, and Fabio Ciravegna

Department of Computer Science, University of Sheffield, UK
{k.elbedweihiy, s.wrigley, f.ciravegna}@dcs.shef.ac.uk

Abstract. Despite the attention Semantic Search is continuously gaining, several challenges affecting tool performance and user experience remain unsolved. Among these are: matching user terms with the search-space, adopting view-based interfaces in the Open Web as well as supporting users while building their queries. This paper proposes an approach to move a step forward towards tackling these challenges by creating models of usage of Linked Data concepts and properties extracted from semantic query logs as a source of collaborative knowledge. We use two sets of query logs from the USEWOD workshops to create our models and show the potential of using them in the mentioned areas.

Keywords: semantic search, query logs, linked data, usability

1 Introduction and Problem Statement

The proliferation of structured data on the web is driving innovation in both ‘conventional’ search (information retrieval – IR) as well as in semantic search. For instance, the use of semantic markup (such as Microformats and RDFa) within existing HTML/XHTML webpages has helped mainstream web search engines such as Google and Bing to enhance their result pages by providing additional and related information to that which would normally have formed the query results.

In contrast to web search engines, Semantic Web search engines such as Swoogle [10], Watson [8] and Sindice [29] index data on the Semantic Web and act more as gateways to Semantic Web documents or data. The results of such systems are intended for Semantic Web professionals rather than end-users. In a more user-friendly approach, mashups like Sig.ma [28] and VisiNav [14] integrate data from different sources to provide rich descriptions about the searched-for concepts. At the heart of these technologies is the use of Linked Data providing the opportunity to exploit explicit and implicit knowledge.

Little work has been conducted on how to exploit Semantic Web search for end-users with potentially complex information needs and thus complex queries. One approach is to provide a natural language interface to such a search tool

^{*} This work was partially supported by the European Union 7th FWP ICT based e-Infrastructures Project SEALS (Semantic Evaluation at Large Scale, FP7-238975). Copyright © 2012 K. Elbedweihiy, S. N. Wrigley and F. Ciravegna. All rights reserved.

which allows a user to express their query in a (near) natural manner. Tools which have followed this approach include Querix [18], PowerAqua [19] and Freya [9]. However, such approaches suffer from a significant problem: a high degree of abstraction between the words which may be used by the user in formulating their query and the underlying semantically-corresponding terms in the ontology. Indeed, this problem is equally applicable to other approaches including a conventional keyword-based approach. Formal evaluations of semantic search technologies [17,30,32] have shown that it is very helpful (both in terms of user experience but also for response precision and recall) for users – especially who are unfamiliar with the underlying data – to explore the search space while building their queries.

In an attempt to provide context within the underlying data, a number of tools have adopted a visual approach to query construction (e.g., Semantic Crystal [4], K-Search [5], Corese [7]). However, such approaches tend to be focussed on relatively small data set sizes (especially when compared to the wider Linked Data context). The outstanding challenge is to adapt this approach to Linked Data in which there are multiple data sets, of widely varying size and spanning many domains. Recently, initial work in this vein demonstrated a visual query interface that helps users build a subgraph of interest from the underlying data through exploration and navigation [6]. However, even this can result in unwieldy lists of triples which are not conducive to a positive user experience.

The difficulties in Semantic Web search are not confined to abstraction, query construction or data visualisation. An additional problem focuses on the results of the query execution: what to return to the user and how to display it. We have shown previously [32] that semantic search tools should go a step further and augment the direct answer with associated information in order to provide a ‘richer’ experience for the user. Additionally, returning information related to entities and concepts found in a query might also be of interest to users [22,23].

In this paper we present a new approach which uses *collaborative knowledge* to address these problems. In a similar way in which traditional search engine logs record information about search histories including queries submitted by the users and their subsequent interactions (see [2,15,27,31] for early examples of analyses on such logs), the logs from interactions with Semantic Web search engines can be used to extract a rich picture of data use and user behaviour.

Whilst a number of studies have used log analysis to investigate the high level use and characterisation of Linked Data [13,23], [24] was the first to study its use through analysing semantic query logs issued to SPARQL endpoints including DBpedia and SWDF. The analysis considered ‘low-level’ factors such as the type of requesting agent (human or software) and the structure of the query. Subsequent studies on the same query logs identified the most common query types, SPARQL features and types of triple patterns [1]; proposed a method to derive more useful labels for LD entities based on the variable names used in the queries [12]; and showed how the analysis of semantic query logs can detect errors and weaknesses in LD ontologies and in turn support their maintenance [20].

In our previous work [11], we introduced a new approach for analysing semantic query logs and found that a small set of concepts and relations in a data set often account for a large proportion of the queries and thus may be of more interest to Linked Data users. In the current paper, we extend this approach in order to demonstrate that careful log analysis can be viewed as a proxy for information need and be used to enhance the search process at a number of different stages from query construction, through search engine optimisation to result presentation. To achieve this, we use three different models, each of which captures information regarding different aspects of the patterns present in the multi-user query logs. We will show how combinations of these three models allow us to address the matching of user search terms to the underlying data vocabulary; the creation of data subgraphs for visualising the underlying data and, finally, for enhancing the results returned to the user.

The remainder of the paper is structured as follows. Section 2 describes the analysis performed on the semantic query logs and the three models used to exploit this analysis. Subsequent sections show how these models can be used to address the three problems described above. Section 3 demonstrates using the models to improve the abstraction problem between user terms and the underlying data vocabulary. Section 4 shows how the results can be augmented in two different ways by exploiting the models. Section 5 illustrates how the models can be used to assist in visualising large data sets for query formulation. It should be emphasized that the details presented in Sections 3, 4 and 5 are a proof of concept for the usage of the models presented in Section 2 (i.e. the results shown come from a “pen and paper” exercise as opposed to having been produced by an implementation of the approach). Finally Section 6 discusses the strengths and weaknesses of the approach and Section 7 draws a number of conclusions and describes directions for future work.

2 Semantic Query Logs Analysis

In previous work [11] we introduced a new approach for analysing and representing information need using semantic query logs. Information need was defined as “the set of concepts and properties users refer to while using SPARQL queries”. A SPARQL query can have one or more triple patterns, solution modifiers (such as LIMIT), pattern matching constructs (such as OPTIONAL) and mechanisms for restricting the solution space (such as FILTERs). A triple pattern consists of three components: a subject, a predicate and an object with each component being either bound (having a specific value) or unbound (as a variable).

Extending our previous analysis [11] we formulate the information inherent in semantic query logs into three models which capture:

- the concepts used together in a query: the query-concepts model
- the predicates used with a concept: the concept-predicates model
- the concepts used as types of one Linked Data entity: the instance-types model

Table 1. Statistics summarising the query logs analysed.

	USEWOD2012	USEWOD2011
Number of queries	8866028	4951803
Number of unique triple patterns	4095011	2641098
Number of unique bound triple patterns	3619216	2571662

We follow the same extraction steps but only extract triple patterns with bound subjects or objects to identify concepts (type of the subject/object) and predicates queried together which are used to build the proposed models.

2.1 Data Set

We use two sets of DBpedia query logs made available at the USEWOD¹ workshops (see Table 1). After extracting bound triple patterns [11], we identify the types associated with each distinct resource appearing as a subject or an object in the query by querying the Linked Data endpoint.

2.2 Models

In order to describe the proposed models, we use the following example query throughout the rest of this section:

```
SELECT DISTINCT ?genre, ?instrument WHERE
{
  <http://dbpedia.org/resource/Ringo_Starr> ?rel <http://dbpedia.org/resource/The_Beatles>.
  <http://dbpedia.org/resource/Ringo_Starr> dbpedia:genre ?genre.
  <http://dbpedia.org/resource/Ringo_Starr> dbpedia:instrument ?instrument.
}
```

Query-Concepts Model This model captures the Linked Data concepts used in a whole query. All bound triple patterns (bound subject or object) in a single query are first identified and their types are retrieved from the Linked Data endpoint. The frequency of co-occurrence of each concept pair is accumulated. For the example query, the types retrieved for ‘Ringo Starr’ include `dbpedia:MusicalArtist` and `umbel:MusicalPerformer` while the ‘The Beatles’ has among its types `dbpedia:Band` and `schema:MusicGroup`. The frequency of co-occurrence of each concept in the first list with each concept in the second list is therefore incremented (e.g. `MusicalArtist` and `Band`).

Concept-Predicates Model This model captures the Linked Data concepts and predicates in a query. Again, bound triple patterns are identified; however, only types of instances used as subjects are retrieved. The frequency of co-occurrence of each of the types with the predicate used in the triple pattern – if available – is accumulated. To illustrate, the second triple pattern in the example query increments the co-occurrence of `dbpedia:MusicalArtist` with `dbpedia:genre` and `umbel:MusicalPerformer` with `dbpedia:genre`.

¹ [http://data.semanticweb.org/usewod/2011\(2012\)/challenge.html](http://data.semanticweb.org/usewod/2011(2012)/challenge.html)

Instance-Types Model Ontologies consist of hierarchies of classes. In theory, these classes are linked together through subsumption or equivalence relationships. In practice, datasets in the Linked Data cloud are yet loosely coupled; lacking the required links [16,26]. A Linked Data entity can have several concepts as its types – from multiple datasets – which are not linked. The knowledge of one type is hence not sufficient for complete reasoning on the data. The *instance-types* model captures the concepts used as types for one instance. For the entity ‘Ringo Starr’, the frequency of co-occurrence of its types `dbpedia:MusicalArtist` and `umbel:MusicalPerformer` are accumulated in the model.

3 Matching User Terms to Linked Data Vocabularies

As explained above, non view-based semantic search approaches face the problem of matching user terms found in a query to the vocabulary of the search-space. [22] tried to tackle this problem on the traditional Web by mapping query terms to relevant concepts in DBpedia. However, scalability can be an issue for both since the matching process can be very expensive especially with the size of the Linked Data cloud. Additionally, although DBpedia is known as a central hub in the cloud, matching query terms to one specific dataset may lead to missing information associated with other semantically-equivalent concepts in different datasets due to the lack of links between them. This affects the ability to reason on the data and return complete results [16,21,26].

Based on their analysis of query logs issued to DBpedia, [11] and [20] drew two important conclusions. Firstly that a small number of classes appeared more frequently in the queries than others (e.g. Film, Place, MusicalArtist, Drug) and, secondly, that the dataset population is not an accurate indicator of the usage/interest in a specific concepts; some concepts had large number of instances and were only queried few times. Supported by the above observations, we believe that creating a model of usage of concepts and relations from query logs has a potential to improve the performance of a semantic search approach with respect to the matching task.

Initial stages followed by the proposed approach for processing the query involves standard NL parsing steps such as tokenization, stemming and lemmatization, and producing a parse tree. Entity extraction and classification is achieved by AlchemyAPI² and the NERD ontology³ is used to map AlchemyAPI classes to the DBpedia classes found in the usage models. Rather than having to query all the underlying data, the models attempt to provide a small abstraction of that data which can act as a source of *collaborative knowledge* [25] capturing the most frequently queried Linked Data concepts and predicates. This could be used to provide matches and, in turn, answers for many commonly issued queries on Linked Data. Since the query terms associated with the entity can be either properties of that entity or concepts in a relation with it, both *query-concepts* and *concept-predicates* model are then queried for matches.

² <http://www.alchemyapi.com>

³ <http://nerd.eurecom.fr/ontology/>

3.1 Illustrative Examples

In the rest of this section, we will use a set of examples and show the results returned by a selection of state-of-the-art systems in order to demonstrate the issues and limitations discussed above. The selected systems are of interest in the SW community, spanning different categories: SW gateways, QA systems and mashups. The examples are carefully selected so that they are not targeting a specific category, and will allow us to illustrate how the proposed models can be used in the matching task.

Example 1: What is the population of New York? The query can be given as a NL question to QA systems or as keywords – also entity query – to other systems; *population of New York*.

Sindice : The top 5 results returned by Sindice are as follows:

1. Armonk, New York : http://www.mpii.de/yago/resource/Armonk,_New_York
2. About: New York City : http://dbpedia.org/page/New_York_City
3. New York State Senate : http://dbpedia.org/resource/New_York_State_Senate
4. Nova Iorque, New York : http://linkeddata.uriburner.com/.../resource/New_York_City
5. Indian-American Population : <http://www.scribd/.../New-York-Citys-IndianAmerican-Population>

The second item in the results is showing the DBpedia page for New York city which contains the answer to the query. However, although being at the top of the list, 60% of the results are only syntactically related to the query (as opposed to *semantically* related); i.e., containing the terms ‘New York’, ‘population’ or both. This is due to using syntax-driven techniques in the matching task which in turn affects the precision of the results and the user experience.

PowerAqua : PowerAqua returns the following answer to the query:

Richard Lewontin : < PopulationGeneticists , birthPlace, New_York >

Although PowerAqua could provide correct answers for other queries, this one shows the effects of the matching problem on the tool’s performance. Attempting to find mappings for the query terms in the whole dataset – DBpedia in this case – resulted in 17 different ones for ‘population’ and three for ‘New York’. They included non-semantically equivalent ones like `yago:RussianPopulationGroups` and `res:General_Population`, which – although they could be excluded before returning the final answer to the user – affected the tool’s performance causing it to require approximately 13 seconds to return the found triples.

FalconS : The top 5 results returned by the object retrieval search option provided by FalconS are as follows:

1. New York City : http://dbpedia.org/resource/New_York_City
2. York : <http://dbpedia.org/resource/York>
3. New York State Assembly: http://dbpedia.org/resource/New_York_State_Assembly
4. York County : http://www.rdfabout.com/rdf/usgov/geo/us/pa/counties/york_county
5. New York State : <http://www.rdfabout.com/rdf/usgov/geo/us/ny>

Although it returns the resource ‘New York City’ in the first rank, the other results retrieved are again only syntactically related to the query terms.

Our Approach : For this example, ‘New York’ is extracted and classified as `alchemyapi:City` which is then mapped to `dbpedia:City`. While no matches were found for ‘population’ associated with the concept `dbpedia:City` in the *query-concepts* model, the *concept-predicates* model returned several matches including `populationTotal` and `dbprop:populationDensityKm`. Since the user query did not provide a specific intent for ‘population’, all the mappings are considered and the answers from equivalent ones (`dbprop:populationTotal` and `populationTotal`) are merged.

To illustrate the use of the *query-concepts* model, consider the query ‘Which islands belong to Portugal’. With syntactical matching, both ‘island’ and ‘Portugal’ could be matched with several instances, predicates or concepts (e.g., `island` and `res:Administrative_divisions_of_Portugal`). However, our approach attempts to find mappings only associated with the concept `dbpedia:Country` – classified type for Portugal – in the *concept-predicates* and the *query-concepts* models resulting in `dbpedia:Island` as the only match returned. The search is therefore done for a relation linking these two concepts. Querying the DBpedia endpoint, these concepts are found to be linked with the property `dbpedia:country` and a list of islands is returned, including `dbpedia-res:Porto_Santo_Island` and `dbpedia-res:Santa_Maria_Island` among others.

Example 2: Give me all the soccer clubs in Spain This query is from QALD workshop open challenge⁴ where both Freya and PowerAqua – the participating tools – did not manage to return back all the results. We’ll be using this example to explain the use of the *instance-types* model. We only compare it to PowerAqua since a demo for Freya is not available.

PowerAqua : PowerAqua matches Spain with six resources including `dbpedia-res:Spain` which is the main resource describing the country in the dataset. This however leads to missing results when the answer is given as cities or other places in Spain rather than the country itself. For instance, one such example of a missing result is `res:CD_Pozo_Estrecho` which is located in `res:Cartagena, Spain`, a resource associated with several types including `dbpedia:Place`, `gml:Feature`⁵ and `schema:Place`. Tools following this approach thus favor precision over recall.

Tools favouring Recall : The other alternative that can be taken by tools in favour of recall over performance (time and scalability) is not to limit the results to a specific type. For the example query, this approach would search for soccer clubs that have a relation with any resource with a label containing the term ‘Spain’ (usually `rdfs:label` is used as a human identifier for the resource). The problem with this approach is that it affects the ability of a tool to scale over large datasets and to return answers for queries in real time [9, 19].

Our Approach : The two approaches explained above can be seen as the two ends of a ‘precision and performance’ versus ‘recall’ spectrum. Our approach attempts to balance the three. The same steps explained in Example 1 are followed

⁴ <http://www.sc.cit-ec.uni-bielefeld.de/qald-1>

⁵ `gml:Feature` refers to <http://www.opengis.net/gml/Feature>

resulting in extracting ‘Spain’ as an entity, classified as a Country and mapped to `dbpedia:Country`. The concepts associated with this class in the *instance-types* model are then extracted. An attempt to find mappings for the noun phrase ‘soccer club’ in the *query-concepts* and *concept-predicates* models results in `dbpedia:SoccerClub` as a match. Next, the approach tries to find instances of `dbpedia:SoccerClub` having a relation with instances of any of the following concepts (retrieved from the instance-types model for `dbpedia:Country`):

```
dbpedia:Country,dbpedia:Place,dbpedia:PopulatedPlace,schema:Place,schema:Country,
gml/_Feature,umbel:Country,umbel:PopulatedPlace
```

This query returns results including ones which would not be retrieved (e.g. `res:CD_Pozo_Estrecho`) if a specific type was specified (e.g., Country). On the other hand, it does not harm the performance since it limits the search space to a set of concepts rather than the whole dataset. One can argue that such information can be extracted for instances from the domain and range of certain properties. However, this not only requires knowledge of the exact properties that would return the results – in this example one of the properties is `dbprop:ground` – but also, as observed by [9], DBpedia properties included in <http://dbpedia.org/property/> do not provide domain and range classes.

4 Results Selection

In an attempt to improve the user experience, Google, Yahoo! and Bing use structured data embedded in web pages to enhance their search results (for example, by providing supplementary information relevant to the query)⁶. Although Semantic Web search engines and question answering systems index much more structured data, a similar functionality (results enhancement) is not yet provided to their users. FalconS returns extra information together with each entity found as an answer to a query. It returns predicates associated with the entity in the underlying data (e.g. type, label, etc.); [32] showed that augmenting the answer with such extra information provides a richer user experience. This is, however, different from Linked Data mashups (e.g. Sig.ma) and browsers (e.g. Tabulator [3]) which attempt to create rich comprehensive views of entities and allow interactive exploration and navigation of Linked Data respectively. Furthermore, [23] and [22] suggested that returning information related to entities found in a query would be of interest to the user.

4.1 Illustrative Examples

In the rest of this section, we illustrate how the proposed models can be used to return more information with the results. We distinguish between providing more information about each result item and more information that is related to the query keywords including concepts and entities.

⁶ For example, Google Rich Snippets: <http://googlewebmastercentral.blogspot.com/2009/05/introducing-rich-snippets.html>

Return additional result-related information To our knowledge, only VisiNav and FalconS return extra information about each entity in the result list. For the query given in 3.1 and for the entity ‘New York City’, FalconS lists the following 10 properties with their values:

```
populationAsOf,dbprop:populationTotal,populationTotal,PopulatedPlace:populationTotal,
populationDensity,PopulatedPlace:populationDensity,dbprop:populationDensitySqMi,
dbprop:populationBlank,dbprop:populationMetro,PopulatedPlace:populationUrban'
```

However, the strength of the proposed idea lies in utilising query logs as a source of collaborative knowledge able to capture perceptions of Linked Data entities and properties and use it to select which information to show the user rather than depending on a manually (or, indeed, randomly) predefined set. Additionally, [22, 23] observed that a class of entities is usually queried with similar relations and concepts.

In order to return more information about each result item, the type of instance returned is first identified then the most frequently queried predicates associated with it are extracted from the *query-predicates* model. The top ranked ones are shown to the user, limited by the space available without cluttering the view and affecting the user experience. The user is given the ability to add more results which would retrieve the next set in the ranked list of predicates. Examples of concepts with their associated predicates list are given below:⁷

```
MusicalArtist-> rdfs:label,rdf:type,thumbnail,.....,genre,associatedBand,occupation,instrument,
birthDate,birthPlace,hometown,prop:yearsActive,foaf:surname,prop:associatedActs, ...
Film-> rdfs:label,rdf:type,foaf:page,.....,prop:starring,prop:director,prop:name,releaseDate,
prop:gross,prop:budget,writer,producer,runtime,prop:language,prop:cinematography, ...
Country-> rdfs:label,rdf:type,thumbnail,....,capital,foaf:name,anthem,language,leaderName,
currency,largestCity,prop:areaKm,motto,...,geo:long,geo:lat,leaderTitle,prop:governmentType, ...
```

Return additional query-related information Returning related information with the results of a query is an attempt to place the queried entities and concepts within context in the surrounding data which indeed assist users in discovering more information and useful findings that otherwise would not be noticed. Following our approach, the query concepts (include concepts and types of entities used in the query) are first identified. The most frequently occurring concepts used with them are extracted from the *query-concepts* model. Again, only a limited set (the actual size of which is determined on an application requirements basis) from the top ranked ones is returned. A set of examples are listed below with their co-occurring concepts.

```
MusicalArtist-> Film,Work,Band,Album,.....,schema:Movie,MusicalWork,Place,Actor,Athlete,
TelevisionShow,WrittenWork,Model,City,GridironFootballPlayer,Writer,schema:Event, ...
City-> Book,Town,WorldHeritageSite,.....,Person,foaf:Person,Country,Organisation,SportsTeam,
SoccerClub,Scientist,Artist,MusicGroup,Film,RadioStation,University,River,Hospital,Park, ...
Company->RecordLabel,foaf:Person,Work,.....,LawFirm,Place,Software,schema:Place,Website,
Broadcaster,TelevisionStation,University,Country,GovernmentAgency,Magazine,Convention, ...
```

⁷ prop is used as a prefix for <http://dbpedia.org/property/> while the default prefix () is for <http://dbpedia.org/ontology/>

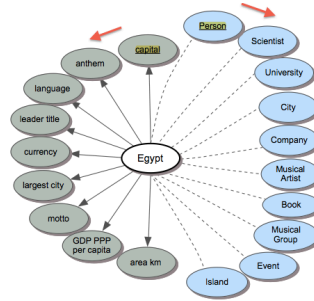


Fig. 1. Results returned by our approach for **Egypt**. Related concepts are on the right side and predicates on the left. For each side, elements are ranked with the top-most being most common and reducing in frequency in the direction of the arrows.

5 Data Visualisation

Query logs have been used in IR to provide query recommendations and suggest similar queries to users [2, 33]. [23] analyzed a set of Yahoo! query logs to learn prefixes and postfixes that can be suggested for a specific type of entities. Similarly, Google and Bing return *related searches* for a query. These approaches are however limited to suggesting parts or complete queries provided by other users in a related context rather than guiding users in formulating their queries.

On the Semantic Web, supporting query formulation is provided by *view-based/visual-query interfaces* (e.g., [4, 6]) which allow users to explore the underlying data. This can be very helpful for users, especially those unfamiliar with the search domain. A problem facing these tools is the technical limitations such as the number of items that can be included in a graph without cluttering the view and affecting user experience. This increases in heterogenous spaces like the open web since it is a challenge to decide what should be shown to users.

In an attempt to tackle this challenge and to identify a specific area of interest, Smeagol [6] introduces a “specific-to-general” interface where it starts from an entity or a term entered by the user and builds a related subgraph extracted from the underlying data. After the user disambiguates the query term from a list of candidates, the tool returns a list of triples containing that term for the user to select from and add to his specific subgraph of data. In a dataset such as DBpedia – currently used by the tool’s demo –, this list will often contain thousands of triples for the user to examine in order to select the required ones.

Our proposed approach uses the *concepts-predicate* and *query-concepts* models to move a step forward towards a more specific subgraph that allows users to explore the data around the entities they start with. It exploits the collaborative knowledge collected from different users and applications to derive the selection of concepts and predicates added to the subgraph of interest.

Using **Egypt** as a starting entity, Fig. 1 shows a set of concepts and predicates associated with this entity’s type in the models. Selecting a related concept retrieves a similar subgraph for the new one and shows the predicates connecting the two concepts.

Table 2. PowerAqua matches for the given queries. The second column gives the number of matches found in the dataset and the third one shows triples generated from the matches to return the answers. Timings in the fourth column are approximate.

Query #	# Found Matches	Relevant Facts	Time (sec)
1	official language: 7 philippines: 4	Language, officialLanguage, Philippines ?, prop:officialLanguages, Philippines	28
2	official websites: 6 Charmed: 4, actors: 5 television show: 3 websites: 5	Award, geminiAward, Actor Award, laurenceOlivierAward, Actor Charmed, IS_A, TelevisionShow Actor, starring, Charmed	22
3	1950: 11 organisations: 7 founded: 9	Organisation, title, 1950's Organisation, foundation, 1950's Organisation, established, 1950's Organisation, founded, 1950's ... Organisation, artist, Project_1950 Organisation, recordLabel, Project_1950	29

6 Discussion

The previous sections illustrate how our models could be adopted in the two main issues discussed: matching user terms to Linked Data vocabulary and returning more information with the results. In this section, we use the following queries to facilitate the discussion of the main strengths and weaknesses of the proposed approach. The queries are from the ‘Interacting with Linked Data’ workshop⁸.

1. What are the official languages of the philippines?
2. Give me the official websites of actors of the television show Charmed.
3. Which organisations were founded in 1950?

In order to show the effect of the matching problem on the tool performance, the mappings for the given queries and the time required – as given by PowerAqua – to find them are given in Table 2.

Query# 2 makes use of the *instance-types* model since the match `prop:website` is found among the predicates queried with `Person` rather than with `Actor`. Additionally, Query# 3 shows that the proposed approach is not limited to queries containing entities. In this as well as similar queries, the *query-concepts* and the *concept-predicates* models are used to find matches for the query terms (e.g. organisations). The matches are then ranked according to their syntactical similarity (exact or partial match) as well as the frequency of usage in the models, resulting in selecting the concept `dbpedia:Organisation` for this query.

As shown in Section 4.1, both *query-concepts* and *concept-predicates* models can contain semantically-equivalent concepts and predicates associated with one concept, from one or more datasets. Although they are used in the matching process in order not to miss candidate results (e.g., `prop:founded`, `foundingDate` in Query# 3), showing several concepts or properties to the user which have the same meaning but different names affects the readability of the results and the user experience. Therefore, the approach should include a schema-matching step before returning information to the user. This is not yet achieved by our approach; it is a challenging task and is part of our future work.

⁸ <http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php>

Table 3. Matches found by the proposed approach for the given queries. The second column shows the query entities and their types. The third one shows the matches found while the last 2 columns show the extra information extracted from the models

Query#	Entity:Type	Found Matches	More Info.	Related Info.
1	philippines Country	schema:Language, Language, language, prop:language prop:officialLanguages, officialLanguage, prop:languages	capital, anthem, language, leaderName, currency, largestCity, areaKm, motto	Person,Scientist, University,City, Company, Book MusicalArtist, Event, MusicalGroup
2	Charmed: Television- Show	Actor, prop:website	occupation, genre, birthPlace, birthDate, surName, hasPhotoCollection, nationality, gender	Film, Album, Organisation, MusicalArtist, SportsTeam, Broadcaster
3	N/A	Organisation, OrganisationMember Non_ProfitOrganisation, GeopoliticalOrganisation prop:founded, foundationPlace, prop:foundationDate, foundingDate	industry, city, country, website, divisions, subsid, president, established, staff, yearsActive, owner, founded foundationPlace	Company,Band Broadcaster, RadioStation, University, School, SoccerClub, MilitaryUnit, Airline

In order to reduce inconsistencies due to noise found in the query logs (incompatible concepts and predicates), the Linked Data endpoint is queried to check the validity of using a specific predicate with a given concept or the existence of a relation between two concepts before adding them to the models. However, there is no similar way to prevent errors in the *instance-types* model since they are caused by inconsistencies found in the dataset. For instance, the concept **Person** was found together with the concept **Country** as types for one Linked Data resource and thus was associated with it in the model. Fortunately, they have a very low frequency of co-occurrence and thus can be easily identified and removed. Another issue to consider is the existence of a few popular generic predicates (e.g. **label**) frequently occurring with most of the concepts and thus ranked at the top of the their associated predicates list. The ones we observed include **rdfs:label**, **rdf:type**, **thumbnail**, **foaf:page**, **rdfs:comment**, **foaf:depiction**, **abstract** and **foaf:homepage**. Although in deciding which predicates to show the user for a specific concept while building a query (Fig. 1), we chose to exclude these generic predicates similar to a stop list in IR, we believe this choice needs to be evaluated through a usability study which could reveal a different view.

7 Conclusions and Future Work

This paper has proposed an approach to support Semantic Search tools in challenges facing them such as matching user terms with Linked Data vocabulary, returning related information with the results and supporting users while building their queries. Following *wisdom of the crowds* and exploiting *collaborative knowledge* found in semantic query logs, the approach attempts to create models of usage of Linked Data concepts and properties. As a proof of concept, we analyzed around 13.5 million DBpedia queries. However, the proposed approach is independent from a specific dataset. Our preliminary results have shown the

potential of adopting the proposed models in an improved semantic search approach. We plan to further evaluate the approach with respect to its performance in matching user terms to Linked Data concepts as well as the quality and relevancy of the returned results as perceived by real users. We think it can additionally be used to create a new vocabulary relying on information needs of Linked Data users and applications and hence customised to best fit their queries. Although the current approach is promising, part of our future work is to investigate the potential benefits available of combining our current models with ones created from traditional query logs as opposed to semantic ones.

References

1. Arias, M., Fernández, J.D., Martínez-Prieto, M.A., de la Fuente, P.: An Empirical Study of Real-World SPARQL Queries. In: Proc. of USEWOD 2011
2. Baeza-Yates, R., Hurtado, C., Mendoza, M.: Query Recommendation Using Query Logs in Search Engines. LNCS 3268 (2004) 588–596
3. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and analyzing linked data on the semantic web. In: Proc. of SWUI 2006
4. Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C.: Querying Ontologies: A Controlled English Interface for End-users. In: Proc. of ISWC 2005
5. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid Search: Effectively Combining Keywords and Ontology-based Searches. In: Proc. of ESWC 2008
6. Clemmer, A., Davies, S.: Smeagol: A specific-to-general semantic web query interface paradigm for novices. In: Proc. of DEXA 2011. LNCS 6860, Springer (2011) 288–302
7. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F.: Searching the semantic web: Approximate query processing based on ontologies. IEEE Intelligent Systems **21**(1) (2006) 20–27
8. D'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., Motta, E.: Characterizing Knowledge on the Semantic Web with Watson. In: EON (2007) 1–10
9. Damjanovic, D., Agatonovic, M., Cunningham, H.: Natural Language Interface to Ontologies: combining syntactic analysis and ontology-based lookup through the user interaction. In: Proc. of ESWC 2010
10. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C., Sachs, J.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proc. of CIKM 2004, ACM Press (2004)
11. Elbedweihy, K., Mazumdar, S., Cano, A.E., Wrigley, S.N., Ciravegna, F.: Identifying Information Needs by Modelling Collective Query Patterns. In: Proc. of COLD 2011
12. Ell, B., Vrandečić, D., Simperl, E.: Deriving human-readable labels from SPARQL queries. In: Proc. of I-Semantics 2011
13. Halpin, H.: A Query-Driven Characterization of Linked Data. In: Proc. of LDOW 2009
14. Harth, A.: VisiNav: A system for visual search and navigation on web data. J. Web Sem. **8**(4) (2010) 348–354

15. Hölscher, C., Strube, G.: Web Search Behavior of Internet Experts and Newbies. *Computer Networks* **33**(1-6) (2000) 337–346
16. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology alignment for linked open data. In: *Proc. of ISWC 2010*
17. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: *Proc. of ISWC/ASWC 2007*
18. Kaufmann, E., Bernstein, A., Zumstein, R.: Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs. In: *Proc. of ISWC 2006*
19. Lopez, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: *The Semantic Web: Research and Applications*, Springer (2006) 393–410
20. Luczak-Rösch, M., Bischoff, M.: Statistical Analysis of Web of Data Usage In: *Proc. of ISWC 2011*
21. Mascardi, V., Locoro, A., Rosso, P.: Automatic ontology matching via upper ontologies: A systematic evaluation. *IEEE Trans. on Knowl. and Data Eng.* **22** (2010) 609–623
22. Meij, E., Bron, M., Hollink, L., Huurnink, B., de Rijke, M.: Mapping queries to the Linking Open Data cloud: A case study using DBpedia. *Web Semantics: Science, Services and Agents on the World Wide Web* **9**(4) (2011) 418 – 433
23. Meij, E., Mika, P., Zaragoza, H.: Investigating the Demand Side of Semantic Search through Query Log Analysis. In: *Proc. of SemSearch 2009*
24. Möller, K., Hausenblas, M., Cyganiak, R., Grimnes, G.A.: Learning from Linked Open Data Usage: Patterns and Metrics. In: *Proc. of WebSci 2010*
25. Murray, G.C., Teevan, J.: Query log analysis: social and technological challenges. *SIGIR Forum* **41** (2007) 112–120
26. Parundekar, R., Knoblock, C.A., Ambite, J.L.: Linking and building ontologies of linked data. In: *Proc. of ISWC 2010*
27. Silverstein, C., Marais, H., Henzinger, M., Moricz, M.: Analysis of a very large web search engine query log. *SIGIR Forum* **33**(1) (1999) 6–12
28. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: live views on the web of data. In: *Proc. of WWW 2010*
29. Tummarello, G., Oren, E., Delbru, R.: Sindice.com: Weaving the Open Linked Data. In: *Proc. of ISWC/ASWC 2007*
30. Uren, V., Lei, Y., Lopez, V., Liu, H., Motta, E., Giordanino, M.: The usability of semantic search tools: a review. *The Knowledge Engineering Review* **22**(4) (2007) 361–377
31. Wen, J.R., Nie, J.Y., Zhang, H.J.: Clustering user queries of a search engine. In: *Proc. of WWW 2001*, New York, NY, USA, ACM Press (2001) 162–168
32. Wrigley, S., Elbedweihy, K., Reinhard, D., Bernstein, A., Ciravegna, F.: Evaluating semantic search tools using the SEALS platform. In: *Proc. of IWEST 2010*
33. Zaïane, O.R., Strilets, A.: Finding Similar Queries to Satisfy Searches Based on Query Traces. In: *Proc. of OOIS 2002*

Linguistic Modeling of Linked Open Data for Question Answering

Matthias Wendt, Martin Gerlach, and Holger Düwiger

Neofonie GmbH, Robert-Koch-Platz 4, 10115 Berlin, Germany
{wendt,gerlach,duewiger}@neofonie.de,
WWW home page: <http://www.neofonie.de/Forschung>

Abstract. With the evolution of linked open data sources, question answering regains importance as a way to make data accessible and explorable to the public. The triple structure of RDF-data at the same time seems to predetermine question answering for being devised in its native subject-verb-object form. The devices of natural language, however, often exceed this triple-centered model. But RDF does not preclude this point of view. Rather, it depends on the modeling. As part of a government funded research project named Alexandria, we implemented an approach to question answering that enables the user to ask questions in ways that may involve more than binary relations.

Introduction

In recent years, the Semantic Web has evolved from a mere idea into a growing environment of Linked Open Data (LOD)¹ sources and applications. This is due in particular to two current trends: The first is automatic data harvesting from unstructured or semi-structured knowledge that is freely available on the internet, most notably the DBpedia project [1]. The second notable trend is the evolution of linked data sources with possibilities of collaborative editing such as Freebase². The growth of LOD gives rise to a growing demand for means of semantic data exploration. Question Answering (QA), being the natural device of querying things and acquiring knowledge, is a straightforward way for end users to access semantic data.

RDF³ and other languages for triple-centered models, which are often used to model and describe linked data, seem to predetermine a specific way of thinking - and of asking questions. Many RDF sources offer information in the form “X birth-place Y” and “X birth-date Z”, etc. Of course, in natural language, we are used to formulate complex queries. It is natural to make statements like “X was born in Y on Z”. While this does not matter as long as singular events like birth or death are involved, things become more complicated as soon as events are involved that can occur more than once. For example, the question “Who was married to Angelina Jolie in 2006?” can only be answered if the temporal (and potentially limited) nature of a relation like marriage is taken into account.

¹ See <http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

² <http://www.freebase.com/>

³ <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

In this paper we present the QA-driven ontology design behind Alexandria⁴, a platform for exploring data of public interest comprising a system for answering questions in German. The domain consists of persons, organizations, locations as well as works such as books, music albums, films and paintings. Moreover, the Alexandria ontology is designed for holding information on events relating the various resources, including temporal information and relations involving more than two participants – so called N-ary Relations. Also, we describe the mapping algorithm used in our question answering system and how it benefits from the ontology design.

The ontology is built from and continuously being updated with data primarily from Freebase, and few parts from DBpedia, news feeds, and user generated content.

Related Work

Open domain question answering is of current research interest. There are several approaches to the subject based on linguistic analysis of natural language questions for generating queries against linked data.

FREyA [5] and PowerAqua [9,10] are both question answering systems that are to a certain degree independent of the underlying ontology schema. Both systems work on existing Linked Open Data *as is* and can be configured to use multiple ontologies. They rely on rather shallow approaches to query mapping, in favor of portability and schema-independence. However, this also limits them to the data structures and languages used by the schemas (e.g., DBpedia does not support N-ary relations).

There are also systems based on deeper, compositional mapping approaches. For example, ORAKEL [3,4] translates syntax tree constructed by lexicalized tree adjoining grammars (LTAGs) to a representation in first order logic which can be converted to F-Logic [8] or SPARQL⁵ queries, depending on the target knowledge base. ORAKEL also principally supports N-ary relations. Though the system is in principle very similar to the one presented in this paper, it is not proven to scale up to a large data set.

In contrast to other projects that use Linked Open Data for question answering, our approach is an attempt to combine the advantages of availability of huge LOD sources and of tailoring the T-Box to the use case of QA. While the latter facilitates the fully automated mapping of natural language questions to SPARQL queries, we trade off the possibility to use existing labels for T-Box entities, which, combined with existing lexical resources such as WordNet⁶, GermaNet⁷, etc., boost lexical coverage.

Another difference to the above-mentioned projects is that the focus of Alexandria is on answering questions in German, not English.

Design of the Alexandria Ontology

The design of the Alexandria Ontology was basically driven by practical demands of the application as well as linguistic considerations. According to [7], our approach can be seen as a unification of the “Type 4” and the “Type 3” approaches to ontology creation. The knowledge base has to meet the following requirements:

⁴ <http://alexandria.neofonie.de/>

⁵ <http://www.w3.org/TR/rdf-sparql-query/>

⁶ <http://wordnet.princeton.edu/>

⁷ <http://www.sfs.uni-tuebingen.de/lsd/>

Linguistic Suitability The data model needs to be suitable for natural language question answering, i.e. mapping natural language parse tree structures onto our data must be possible.

LOD Compatibility Compatibility with existing LOD sources like Freebase and DBpedia needs to be maintained in order to facilitate mass data import for practical use.

Scalability Large amounts of data need to be stored, maintained and updated while keeping the time for answering a question at minimum.

One of the major aspects relating to *linguistic suitability* in the Alexandria use case is that its target domain goes beyond what we refer to in the following as *attributive data*, i.e. data about things that are commonly known as named entities like persons, organizations, places, etc. In addition, the domain was designed to contain what we call *eventive data*, i.e. (historic) events and relations to participants within them.

As mentioned above, there are certain relations, such as *birth*, where this distinction is not important, because n-ary relations consisting of unique binary parts (like place and date of birth) can be covered by joining on a participant (the person) as proposed in [4]. The distinction between eventive and attributive data becomes important when relations are involved, which (may) occur repetitively and/or include a time span. Questions like “Who was married to Angelina Jolie in 2001?” and “Which subject did Angela Merkel major in at the German Academy of Sciences?” can no longer be generally answered by joining binary facts.

It is possible to model such eventive n-ary facts as proposed in Pattern 1, use case 3 of the W3C Working Group Note on N-ary Relations on the Semantic Web⁸. This approach is also close to the semantic model advocated in Neo-Davidsonian theories [12], where participants in an event are connected to the event using roles.

As for the aspect of *LOD compatibility*, it is our aim to access existing large-scale sources to populate our knowledge base. DBpedia was the first LOD source to retrieve and constantly update its data repository by crawling Wikipedia⁹. Apart from its possibilities for end-users to add and update information, the majority of data contained in Freebase is obtained from Wikipedia as well. Therefore, using one (or both) of these sources is an obvious starting point for harvesting information on a broad range of popular entities, as it is required by Alexandria.

However, though DBpedia contains much valuable attributive data for entities of our interest, it does not offer eventive information as stated above. Also, DBpedia’s T-Box does not provide a model for adding such n-ary facts, either.

As opposed to DBpedia, which relies on the RDF standard, Freebase implements a proprietary format. Whereas in RDF, all information is abstractly represented by triples, Freebase abstractly represents information as links between topics. The Freebase data model incorporates n-ary relations by means of Compound Value Types¹⁰, also called “Mediators”. A mediator links multiple topics and literals to express a single fact.

So Freebase’s data model suits our requirements, but we need to use RDF to be able to use Virtuoso Open Source Edition¹¹ which has proven to *scale well* for both loading and querying the amounts of data we expected.

⁸ <http://www.w3.org/TR/swbp-n-aryRelations/>

⁹ <http://www.wikipedia.org/>

¹⁰ http://wiki.freebase.com/wiki/Compound_Value_Type

¹¹ <http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/>

Using the Freebase query API to pull a set of topics and links, an RDF based knowledge base can be built according to the Neo-Davidsonian model. The API also supports querying link updates for continuously updating the knowledge base.

There are straightforward mappings of Freebase topic and mediator types onto OWL¹² classes, and of Freebase link types onto OWL properties. For example, a marriage relation is imported from Freebase as follows:

```
nary:m_02t82g4 rdf:type      dom:Marriage ;
                dom:spouse    res:Angelina_Jolie ;
                dom:spouse    res:Brad_Pitt ;
                alx:hasStart   "2005"^^xsd:gYear .
```

The subject URI is generated from the Freebase mediator ID. The resource URIs are generated from Freebase topic names with some extra processing and stored permanently for each Freebase topic ID.

We differentiate the following three layers of our ontology (which correspond to three namespace prefixes alx:, dom:, and res: that appear in the examples):

The Upper Model (alx:) contains the abstract linguistic classes needed for a language-, domain- and task-independent organization of knowledge. The Alexandria upper model is inspired by [6].

The Domain Model (dom:) contains the concrete classes and properties for entities, events and relations of the modeled domain (e.g., Marriage, Study) as subclasses of the upper model classes and properties. Needed to make the domain-specific distinctions which are necessary for the task of question answering.

The A-Box (res:) consists of all “resources”, i.e. entity, event and relation instances, known to Alexandria.

The examples of Angela Merkel’s education and Angelina Jolie’s and Brad Pitt’s marriage, which we used above, would be represented as shown in Table 1.

Upper model concept	Domain concept	U.m. role props.	Domain props.	Participant
agentive process	study	agent	student	Angela Merkel
effective process	study	affected	subject	Quantum Chemistry
locative relation	study	location	institution	German Academy of Sciences
attributive rel.	marriage*	located	student	Angela Merkel
		carrier	spouse	Brad Pitt
		attribute	spouse	Angelina Jolie
temporal concept	marriage	start	wedding date	2005

* In this example, marriage is modeled as a symmetric relation, expressing a spouse as attribute of the other spouse, i.e. carrier and attribute may be swapped.

Table 1. Upper and domain model

¹² Web Ontology Language, builds on RDF, see <http://www.w3.org/TR/owl2-overview/>

Syntactically, we model our domain concepts as OWL subclasses of one or more upper model concepts and our domain properties as OWL subproperties of one or more upper model properties, where the latter correspond to the Neo-Davidsonian roles mentioned earlier. We can then obtain hints to the upper model concepts and roles of interest by mapping question verbs onto domain concepts and then try to match the roles defined for the upper model classes to the respective given parts of the question.

Putting the Model in Action: Question Answering

As mentioned above, one of the major design goals of our ontology schema was to stay reasonably close to the phenomena and structure of natural language. Achieving this would facilitate the mapping of a natural language question to a SPARQL graph pattern that conveys the information need expressed in the question. The basic idea of the translation algorithm is to understand the problem of mapping of natural language to SPARQL as a graph mapping problem.

From a linguistic viewpoint, the syntactic structure of a sentence may be represented in the form of a dependency tree, as obtained by the application of a dependency parser. A dependency graph is formalized like this:

Given a set L of dependency labels, a *dependency graph* for the sentence $x = w_1 \dots w_n$ is a directed graph $D = (V_D, E_D)$ with:

1. V_D is a set of vertices $\{w_0, w_1, \dots, w_n\}$ and
2. $E_D \subseteq V_D \times L \times V_D$ a set of labeled edges

The vertices are composed of the tokens in the sentence plus an artificial root node w_0 . A well-formed dependency graph is a tree rooted at w_0 .

Likewise, the structure of a SPARQL Select query basically consists of a graph pattern (in the Where clause) and a projection. Given a set of variable names N_V ($?x, ?y \dots$), the set of concept names N_C , a set of role names N_P , a set of resource names N_R and a set of literals N_L defined by the ontology, we define a SPARQL Select Graph $G = (V_G, E_G, P_G)$ as:

1. $V_G \subseteq N_V \cup N_R \cup N_C \cup N_L$
2. $E_G \subseteq N_V \cup N_R \times N_P \times V_G$
3. the projection $P_G \subseteq N_V$

Formally, we define the translation as a mapping $f(D)$ of a dependency graph D to a SPARQL Select Graph G .

Linguistic Processing

The dependency graph is the result of the application of a **linguistic analysis** to the input sentence. An example of a resulting dependency structure may be found on the left in Figure 1. The analysis consists in tokenization, POS-tagging¹³ and dependency parsing. Dependency parsing is conducted using the MaltParser [11], which was trained

¹³ For German tokenization and POS-Tagging we use OpenNLP with some pre-trained models. (<http://incubator.apache.org/opennlp/>)

on the German Tiger corpus¹⁴ [2]. The corpus has been slightly adapted by adding a small sub-corpus of German questions and a minor change to the set of role labels used.

To normalize surface form variation and identify morphosyntactic features, lemmatization and morphological analysis is applied to each of the tokens. This is roughly illustrated by the lemmata in square brackets at the verbal nodes (e.g. “verheiratet” has the lemma “verheiraten”).

Compositional Semantics

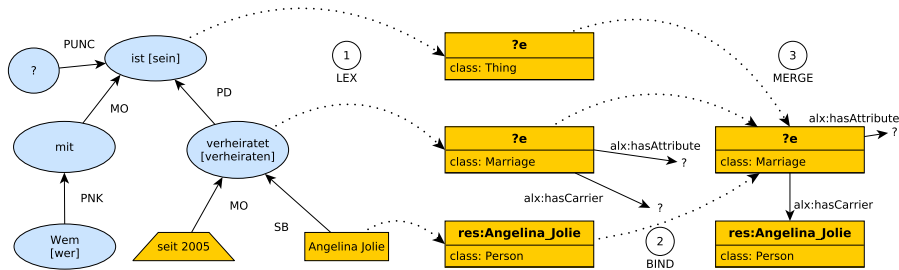


Fig. 1. Dependency parse of the sentence “Mit wem ist Angelina Jolie seit 2005 verheiratet?” and examples of the lexicalization (1) for a subset of its nodes, and the application of the actions BIND (2) and MERGE (3).

The mapping of the dependency graph to the SPARQL query is largely done in two steps: *lexicalization* and *composition*. By *lexicalization* we refer to the process of mapping tokens (lemmata) or multi-word units to corresponding ontological units.

We refer to the identification of resources (identified by resource URIs) of the A-Box as *lexical named entity identification*. For this, we make use of the title (the name of an entity) and the alternative names (consisting of synonyms and different surface forms) that are imported from Freebase into a Lucene¹⁵ index containing the resource URI in Alexandria (e.g. **res:Angelina_Jolie**), and the OWL classes it belongs to. While the user enters a question, matching entities are looked up in the index based on whole words already entered and a disambiguation choice is continuously updated. The user can select from the found entities at any time, whereupon the respective part of the question is updated.

The second noteworthy component in lexical named entity identification is the identification of dates (and time). For these, we have adapted the open source date parser provided by the Yago project¹⁶ to German.

All other linguistic tokens or configurations (linguistic units) corresponding to T-Box concepts are mapped using hand-crafted lexica.

The complete set of mappings for the question shown in Fig. 1 is shown in Table 2.

¹⁴ <http://www.ims.uni-stuttgart.de/projekte/TIGER/>

¹⁵ <http://lucene.apache.org/>

¹⁶ <http://www.mpi-inf.mpg.de/yago-naga/javatools/>

T-Box Class	T-Box Role	A-Box URI	Literal	T-Box Class
Custom Lexica		Lucene	Date, Literal Parser	Custom Lexica
"wer"	"mit"	"Angelina Jolie"	"2005"	"verheiraten" "sein"
dom:Person	alx:hasAttribute	res:Angelina_Jolie	"2005"^^xsd:date	dom:Marriage owl:Thing

Table 2. Types of Lexical Mappings

Our syntax-semantics mapping is largely done by the composition of the lexical semantic entries attached to each dependency node. This lexicalized approach devises the notion of a *semantic description*. A semantic description represents the semantic contribution of a dependency node or (partial) dependency tree and encodes obligatory semantic complements (slots). During the composition, the slots are being filled by semantic descriptions (properties) until the semantic description is satisfied.

By virtue of the lexical mapping each linguistic unit is mapped to a set of semantic descriptions, also called *readings*.

Given a set of variable names N_V , the set of concept names N_C , a set of role names N_P , a set of resource names N_R and a set of literals N_L defined by the ontology, a semantic description S of an ontological entity $n \in N_V \cup N_L \cup N_R$ is defined as a five-tuple $S = (n, c, Sl, Pr, Fl)$ with:

1. $c \in N_C$ the concept URI of the semantic description
2. $Sl = [r_1, r_2, \dots, r_n]$ a ordered set of slots ($r_i \in N_P$)
3. $Pr = \{(p_1, S_1), \dots (p_m, S_m)\}$ with S_j a semantic description and $p_m \in N_P$
4. $Fl \subseteq \{proj, asc, desc\}$ a set of flags (with *proj* indicating that n to be part of the projection of the output graph

For convenience, we define the following access functions for the semantic descriptions $S = (n, c, Sl, Pr, Fl)$:

1. $node(S) = n \Leftrightarrow S = (n, c, Sl, Pr, Fl)$
2. $pred(S) = \{p | (p, o) \in Pr\} \Leftrightarrow S = (n, c, Sl, Pr, Fl)$

A semantic description $S = (n, c, Sl, Pr, Fl)$ is well-formed if the set of bound properties and the slots are disjoint $pred(S) \cap Sl = \emptyset$ and all bound properties are uniquely bound $\forall (p, o) \in Pr \rightarrow \neg \exists (p, o_1) \in Pr \wedge o \neq o_1$.

By definition, there is a strong correlation between a semantic description and a SPARQL Select query. A SPARQL Select query can recursively be built from a semantic description $S = (n, c, Sl, Pr, Fl)$ and an initially empty input graph $G_0 = (V_{G_0} = \emptyset, E_{G_0} = \emptyset, P_{G_0} = \emptyset)$:

```

toSPARQL( $S, G_0$ ):  $G_m$ 
 $G_0 = (V_0 \cup \{n\}, E_0, P_0)$  : the output SPARQL graph pattern
 $E_0 = E_s \cup \{(n, a, c)\}$ 
 $P_0 \leftarrow P \cup \{n\} \Leftrightarrow proj \in Fl$  otherwise  $P_0 = P$ 

foreach  $(p_i, o_i)$  in  $(p_1, o_1), \dots, (p_m, o_m) = Pr$ 
begin
   $E_i \leftarrow E_{i-1} \cup (n, p, node(o))$ 
   $G_i \leftarrow \text{toSPARQL}(o_i, G_{i-1})$ 
end
return  $G_m$ 

```

To give an example in an informal notation, the linguistic units of the sentence “Mit wem ist Angelina Jolie seit 2005 verheiratet?” are displayed in Table 3. The first row shows the linguistic unit, the ontological unit described corresponds to the variable or resource URI in the second row. The prefix `?!x` in a variable designation (e.g. `?!x`) is equivalent to the flag *proj*, denoting that the variable will be part of the projection, i.e. $proj \in Proj$. Note that the verbal nodes “verheiratet” and “sein” are each mapped to a (distinct) variable `?e`, which corresponds to the Neo-Davidsonian event variable *e*.

Slots and bound properties are displayed in the third column. The slots are designated with the argument being just a `?`, whereas a variable is denoted by the prefix `?` and a lower case letter (e.g. `?v`). In the example above the semantic descriptions for “verheiratet” and “mit” contain the slots `alx:hasCarrier` (verb only) and `alx:hasAttribute`.

“Angelina Jolie”	<code>res:Angelina.Jolie</code>	<code>a dom:Person</code>
“seit 2005”	<code>?e</code>	<code>a alx:TemporalRelation ; alx:hasStart 2005 .</code>
“mit”	<code>?e</code>	<code>a alx:AttributiveRelation ; alx:hasAttribute ?</code>
“wem”	<code>?!x</code>	<code>a dom:Person</code>
“sein”	<code>?e</code>	<code>a alx:AttributiveRelation</code>
“verheiratet”	<code>?e</code>	<code>a dom:Marriage ; alx:hasCarrier ? ; alx:hasAttribute ?</code>

Table 3. Semantic descriptions of lexical units.

Putting it all Together

The composition algorithm devises a fixed set of two-place composition operators, called actions. An action defines the mapping of two semantic descriptions related to an edge in the dependency graph to a composed semantic description, corresponding to the semantics of the subgraph of the dependency tree.

The two most important actions are **BIND** and **MERGE**. These two basic operations on the semantic descriptions involved in the composition intuitively correspond to (1) the mapping of the syntactic roles to semantic roles (otherwise called semantic role labeling) and (2) the aggregation of two nodes to one in the output graph pattern. Given two semantic descriptions $S_1 = (n_1, c_1, Sl_1, Pr_1, Fl_1)$ and $S_2 = (n_2, c_2, Sl_2, Pr_2, Fl_2)$, the semantic operators are defined like this:

$$\begin{aligned}
 BIND(S_1, S_2) &= \begin{cases} S(v, c_1, Sl_1 - \{\max(Sl_1)\}, Pr_1 \cup \{(\max(Sl_1), S_2)\}) & \text{if } \text{range}(\max(Sl_1)) \sqcap c_2 \neq \perp \\ NULL & \text{otherwise} \end{cases} \\
 MERGE(S_1, S_2) &= \begin{cases} S(v, lcs(c_1, c_2), Sl_1 \cup Sl_2, Pr_1 \cup Pr_2) & \text{if } c_1 \sqcap c_2 \neq \perp \\ \wedge pred(S_1) \cap pred(S_2) = \emptyset & \\ NULL & \text{otherwise} \end{cases}
 \end{aligned}$$

The function $lcs(c_1, c_2)$ gives the least common subsumer of two concepts, i.e. a concept c such that $c_1 \sqsubseteq c$ and $c_2 \sqsubseteq c$ and for all e such that $c_1 \sqsubseteq e$ and $c_2 \sqsubseteq e$ then $c \sqsubseteq e$. The semantic role labeling implemented by BIND depends on a total order of semantic roles, which has to be configured in the system, e.g.:

`alx:hasAgent > alx:hasAffected > alx:hasRange`

This order determines the ordering of the slots Sl in a semantic description. It stipulates a hierarchy over the semantic roles of an n-ary node in the SPARQL graph pattern. It is reflected by an ordering over the syntactic role labels which is roughly equivalent to the linguistic notion of an *obliqueness hierarchy* [13], for example:

`SB > OC > OC2`

Formally, the obliqueness hierarchy defines a total order $>_L$ over the set of dependency labels L .

For the composition, each of the labels in the label alphabet is assigned one of the semantic operators. The algorithm chooses the operator that is defined to build the composition. The following table shows an excerpt of this mapping:

SB	OC	PNK	MO	PD	PUNC
BIND	BIND	BIND	MERGE	MERGE	IGNORE

The action IGNORE simply skips the interpretation of the subtree. The composition algorithm iterates over the nodes in the dependency graph in a top-down manner, for each edge applying the action defined for the edge label pairwise to each reading of the source and target node.

The algorithm works in a directed manner by sorting the outgoing edges of each node in the dependency graph according to a partial order \geq_D .

$$(v_1, l_1, w_1) \geq_D (v_2, l_2, w_2) \Leftrightarrow l_1 > l_2$$

This ordering stipulates a hierarchy over the syntactic arguments in the dependency graph that is reflected by a total ordering on the role labels of the SPARQL pattern graph W : $>_W$. The correspondance between these orders controls the order in which the graph is traversed and therefore, in particular, the correlation of syntactic and semantic roles (semantic role labeling).

The mapping is implemented by the transformation algorithm sketched below. It takes as input a dependency graph $D(V_D, E_D)$ with the root node w_0 as the initial node c in the graph traversal. The nodes are traversed in the order of the hierarchy to assure the correct binding. Note that the transformation may have multiple semantic descriptions as its output. An output semantic description $S = (n, c, Sl, Pr, Fl)$ is only accepted, if all of its slots Sl have been filled. We then apply the toSPARQL operation to arrive at the final SPARQL query.

transform $(D, c) : S$
 $c \leftarrow w_0$: the current node
 $S \leftarrow \emptyset$: the set of output readings

```

foreach  $(c, l, v)$  in sort(outgoing( $c$ ),  $\geq_D$ )
begin
   $R_c \leftarrow \text{readings}(c)$ 
   $R_v \leftarrow \text{transform}(v, D)$ 
  foreach  $(r_c, r_v)$  in  $R_c \times R_v$ 
  begin
     $s \leftarrow \text{apply}(\text{operator}(l), r_c, r_v)$ 
    if  $(s \neq \text{NULL})$ 
       $S \leftarrow S \cup \{s\}$ 
    end
  end
end

```

Results

The N-ary modeling requires more triples for simple (binary) facts than using RDF/OWL properties like DBpedia, because there is always an instance of a relation concept comprising `rdf:type` and participant role triples.

At the time of writing, the Alexandria ontology contained approx. 160 million triples representing more than 7 million entities and more than 13 million relations between them (including literal value facts like amounts, dates, dimensions, etc.). We imported the triples into Virtuoso Open Source Edition, which scales as well as expected with respect to our goals.

80% of all query types understood by the algorithm (i.e. mappable onto valid SPARQL queries) take less than 20ms in average for single threaded linguistic processing on a 64 bit Linux system running on Intel[®] Xeon[®] E5420 cores at 2.5GHz, and pure in-memory SPARQL processing by Virtuoso Open Source Edition on a 64 bit Linux system running on eight Intel[®] Xeon[®] L5520 cores at 2.3GHz and 32GB of RAM.

The question answering system works fast enough to be used in a multi-user Web frontend like <http://alexandria.neofonie.de>.

The performance of the algorithm is in part due to the high performance of malt parser with a liblinear model, which runs in less than 5ms per question. By using a liblinear model, however, we trade off parsing accuracy against performance in terms of processing time per question. This sometimes becomes noticeable in cases where subject-object order variation in German leads to an erroneous parse.

Answer set	avg. precision	avg. recall	avg. f-measure
All answers	0.25	0.27	0.25
Generated answers	0.49	0.52	0.48
Answers without data mismatch	0.59	0.57	0.57
Generated answers without data mismatch	0.92	0.89	0.89

Table 4. Quality of results of the Alexandria question answering on the QALD-2 training set translated to German.

The performance of the question answering system has been measured using the training set of the QALD-2 challenge¹⁷. As the question answering in Alexandria currently covers only German, all 100 questions were translated to German first. The results are shown in Table 4. For 49 of the questions no query could be generated. The second row shows the results for the questions for which a SPARQL query could be generated.

It has to be noted that the results provided in the gold standard rely on the DBpedia SPARQL endpoint. As Alexandria is built upon its own schema and the imported data comes from Freebase instead of DBpedia, the comparability of the results is limited. The comparison of both datasets results in various mismatches. For example, the comparison of questions having a set of resources as answer type, is done via the indirection of using the labels. This is possible just because the labels are extracted from Wikipedia by both Freebase and DBpedia. However, some of the labels have been changed during the mapping.

Overall, we have identified the following error types:

1. different labels for the same entities
2. different number of results for aggregate questions
3. query correct but different results
4. training data specifies “out of scope” where we can provide results
5. question out of scope for Alexandria

Type 1 applies particularly often to the labels of movies, most of which are of the form “Minority Report (film)” in DBpedia, and “Minority Report” in Alexandria. Another source for errors (2) results when aggregate questions (involving a count) retrieve a different number of resources. The question “How many films did Hal Roach produce?” for example yields 509 results in DBpedia and 503 results in Alexandria. The third type corresponds to a difference in the data set itself, that is when different information is stored. For example, in Alexandria the highest mountain is the “Mount Everest” whereas in DBpedia it is the “Dotsero”.

The last two error types involve questions that are out of scope (4 and 5). The data model used in Alexandria differs from the model in DBpedia as a result to the considerations explained above. On the other hand, Alexandria lacks information since we concentrate on a mapped subset of Freebase. According to the evaluation, the answer “out of scope” is correct if the question cannot be answered using DBpedia.

Out of the 82 questions containing (any) erroneous results 63 belong to one of the error classes mentioned above. The last two rows of Table 4 show the results for all questions that do not belong to any of these error classes.

Acknowledgements

Research partially funded by the German Federal Ministry for Economics and Technology as part of the THESEUS research program¹⁸.

¹⁷ <http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=challenge&q=2>

¹⁸ <http://theseus-programm.de/en/about.php>

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia : a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web* **7**(3) (2009)
2. Brants, S., Dipper, S., Hansen, S., Lezius, W., Smith, G.: The TIGER Treebank In: *Proc. of the Workshop on Treebanks and Linguistic Theories* (2002)
3. Cimiano, P.: ORAKEL: A Natural Language Interface to an F-Logic Knowledge Base. In: *Proc. of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB)* (2004) 401–406
4. Cimiano, P., Haase, P., Heizmann, J., Mantel, M.: Orakel: A portable natural language interface to knowledge bases. Technical Report, University of Karlsruhe (2007)
5. Damjanovic, D., Agatonovic, M., Cunningham, H.: FREyA: an Interactive Way of Querying Linked Data using Natural Language. In: *Proc. of QALD-1 at ESWC 2011*
6. Elhadad, M., Robin, J.: SURGE: a Comprehensive Plug-in Syntactic Realization Component for Text Generation. Technical Report (1998)
7. Hovy, E.: Methodologies for the Reliable Construction of Ontological Knowledge. In: *Proc. of ICCS 2005* 91–106
8. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* **42** (1995) 741–843
9. Lopez, V., Motta, E., Uren, V.: PowerAqua: Fishing the Semantic Web. In: *Proc. of ESWC 2006* 393–410
10. Lopez, V., Nikolov, A., Sabou, M., Uren, V., Motta, E., DAquin, M.: Scaling Up Question-Answering to Linked Data. In: Cimiano, P., Pinto, H. (eds.): *Knowledge Engineering and Management by the Masses. LNCS 6317* (2010) 193–210
11. Nivre, J., Hall, J.: Maltparser: A language-independent system for data-driven dependency parsing. In: *Proc. of the 4th Workshop on Treebanks and Linguistic Theories* (2005) 13–95
12. Parsons, T.: *Events in the Semantics of English: A study in subatomic semantics.* MIT Press (1990)
13. Pollard, C., Sag, I.: *Information-based Syntax and Semantics, Vol. 1. CSLI Lecture Notes 13* (1987)

QAKiS @ QALD-2

Elena Cabrio¹, Alessio Palmero Aprosio^{2,3}, Julien Cojan¹,
Bernardo Magnini², Fabien Gandon¹, and Alberto Lavelli²

¹ INRIA, 2004 Route des Lucioles BP93, 06902 Sophia Antipolis cedex, France.
{elena.cabrio, julien.cojan, fabien.gandon}@inria.fr

² FBK, Via Sommarive 18, 38100 Povo-Trento, Italy.
{lavelli, magnini}@fbk.eu

³ Università degli Studi di Milano, Via Festa del Perdono 7, 20122 Milano, Italy.
alessio.palmero@unimi.it

Abstract. We present QAKiS, a system for Question Answering over linked data (in particular, DBpedia). The problem of question interpretation is addressed as the automatic identification of the set of relevant relations between entities in the natural language input question, matched against a repository of automatically collected relational patterns (i.e. the WikiFramework repository). Such patterns represent possible lexicalizations of ontological relations, and are associated to a SPARQL query derived from the linked data relational patterns. Wikipedia is used as the source of free text for the automatic extraction of the relational patterns, and DBpedia as the linked data resource to provide relational patterns and to be queried using a natural language interface.

Keywords: Question Answering, Linked Data, relation extraction

1 Introduction

As the social web is spreading among people and the web of data continues to grow (e.g. Linked Data initiatives), there is an increasing need to allow easy interactions between non-expert users and data available on the Web. In this perspective we address the development of methods for a flexible mapping between natural language expressions, and concepts and relations in structured knowledge bases. Specifically, we present QAKiS, the Question Answering system that we have submitted at the QALD-2 evaluation exercise.

QAKiS, Question Answering wiKiframework-based System, allows end users to submit a query to an RDF triple store in English and get results in the same language, masking the complexity of SPARQL queries and RDFS/OWL inferences involved in the resolution, while profiting from the expressive power of these standards. In the actual implementation, QAKiS addresses the task of QA over structured knowledge bases (e.g. DBpedia), extracted from corpora in natural language. QAKiS builds on top of previous experiences on QA over structured data in closed domains, particularly the QALL-ME system [3], aiming at enhancing the scalability of the approach and its portability across domains. A crucial issue in Question Answering over linked data, and a major focus of the QAKiS system, is the interpretation of the question in order to convert it into a corresponding query in a formal language (e.g. SPARQL). Most of

current approaches (e.g. PowerAqua [4]) base this conversion on some form of flexible matching between words of the question and names of concepts and relations of a triple store. However, a word-based match may fail to detect the relevant context around a word, without which the match might be wrong. In our first participation at QALD-2 we try to exploit a relation-based match, where fragments of the question are matched to relational textual patterns automatically collected from Wikipedia (i.e. the WikiFramework repository). The underlying intuition is that a relation-based matching would provide more precision w.r.t. matching on single tokens, as done by current QA systems on linked data.

2 WikiFramework patterns

Our QA approach makes use of relational patterns automatically extracted from Wikipedia and collected in the WikiFramework repository [5]. Relational patterns capture different ways to express a certain relation in a given language. For instance, the relation `birthDate(Person, Date)` can be expressed in English by the following relational patterns: `[Person was born in Date]`, `[Person, (Date)]`, `[Person, whose date of birth is Date]`.

Goal of the WikiFramework is to establish a robust methodology to collect relational patterns in several languages, for the relations defined in DBpedia ontology (similarly to [2, 6]). Therefore, following the semantic web RDF model, each relation is represented by a set of triples $\langle S, P, O \rangle$ where S is the subject (domain) of the relation, O is the object (range), and P (predicate or property) is the name of the relation. For example, an instance of the `crosses` relation is:

```
<http://dbpedia.org/resource/Golden_Gate_Bridge>
<http://dbpedia.org/ontology/crosses>
<http://dbpedia.org/resource/Golden_Gate>
```

or using namespaces and prefixes:

```
dbr:Golden_Gate_Bridge dbo:crosses dbr:Golden_Gate
```

We assume that there is a high probability that the structured information present in the Infobox (i.e. a table put in the right-hand corner of most of Wikipedia articles providing a structured summary of information mentioned in the text) is also expressed using natural language sentences in the same Wikipedia page. Therefore as a first step, we collect all the triples for the 1296 DBpedia ontology relations: for each of them we extract the subject (e.g. `Golden Gate Bridge`), and we automatically match each DBpedia relation with the Wikipedia pages whose topic is the same as the subject, and in which such relation is reported in the Infobox. For instance, given the relation `crosses` reported above, the Wikipedia page about the Golden Gate Bridge is selected (Figure 1).

As a second step, we collect all the sentences in the selected Wikipedia pages where both the strings of subject and object of the relation match. For instance, in the page about the Golden Gate Bridge (Figure 1), the sentence “The Golden Gate Bridge is a suspension bridge spanning the Golden Gate” is detected, since both entities match (i.e. domain: Golden Gate Bridge, range: Golden Gate).

Such sentence is extracted and the subject and object are substituted with the corresponding DBpedia ontology classes (i.e. for the relation **crosses**, **Bridge** is the domain and **River** is the range). The pattern [The Bridge is a suspension bridge spanning the River] is obtained and stored in a pattern repository.

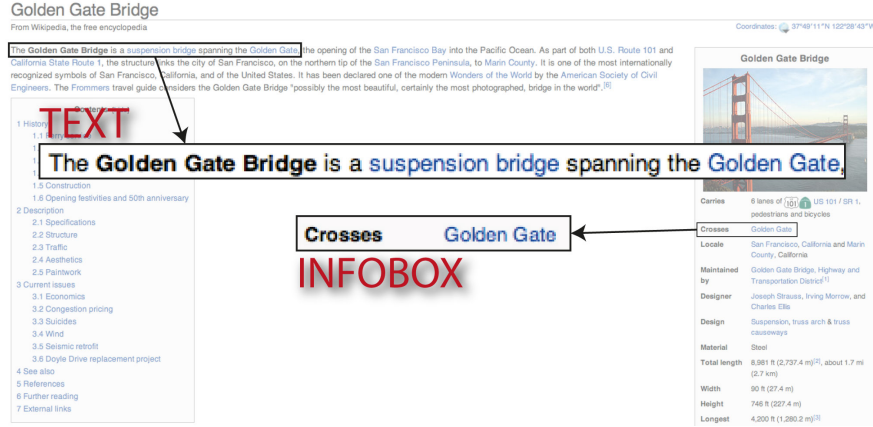


Fig. 1. An example of Wikipedia page with infobox.

To increase the recall of the pattern extraction algorithm outlined above, we apply different matching strategies. The first one involves the exact match of the entire string (the subject), as provided by Wikipedia. If the algorithm does not find such string, we clean it deleting the expressions between brackets, used to disambiguate pages with the same title, as in “Carrie (novel)” and “Carrie (1976 film)”. In the final step, the original string is tokenized and lemmatized using Stanford CoreNLP⁴ and, given the set of obtained tokens, a new string is built by combining them (preserving the original word order). For instance, starting from “John Fitzgerald Kennedy”, we obtain the new strings “John Fitzgerald”, “John Kennedy”, “Fitzgerald Kennedy”, “John”, “Fitzgerald” and “Kennedy”. In case of numeric or time ranges, we apply the entity extractor of Stanford CoreNLP, and match the value in the extracted string with the given normalized value.

Once the patterns for all relations are collected, we cluster them according to the lemmas that are present between the domain and the range. Then, we sort such patterns according to the frequency they appeared in Wikipedia pages, and we wipe out: *i*) the ones whose frequency is less than 2, and *ii*) the ones that contain only non discriminative words (e.g. punctuation marks, prepositions, articles, etc. as in the pattern [Person (Date)]). Table 1 reports an extract of the set of patterns collected for the relations **spouse** and **crosses**.

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

Table 1. Examples of patterns.

Relation	Patterns
spouse	Person wife Person
	Person married Person
	Person husband Person
crosses	Bridge spanning River
	Bridge bridge River
	Bridge crossing River

As the last step, three sets of keywords for each relation are created, respectively for most frequent tokens, lemmas and stems. Each set contains 20 words, sorted by the frequency of presence in the collected patterns. To further improve recall, we finally append to the sets of keywords the tokens, lemmas and stems extracted from the CamelCase name of the relation (e.g. the tokens “birth” and “date” are added to the keywords of the relation **birthDate**).

3 QAKiS system description

This section presents the architecture of QAKiS, and a step by step description of the system work-flow.

3.1 System architecture

QAKiS is composed of two main components (as shown in Figure 2):

- the *query generator* is the entry point to the system: it reads the questions from the input file, generates the typed questions to be matched with the patterns (as explained in Section 3.2), and then generates the SPARQL queries from the retrieved patterns (Section 3.3). If several SPARQL queries are generated, the query with the highest matching score is selected among those that have at least one result;
- the *pattern matcher* takes as input a typed question, and retrieves the patterns (among those stored in the pattern repository) that match such question with the highest similarity (Section 3.2).

3.2 Typed questions generation and pattern matching

The first version of QAKiS (with which we participated in the QALD-2 challenge) targets questions containing a Named Entity (NE) that is related to the answer through one property of the ontology, as *Which river does the Brooklyn Bridge cross?* (id=4, training set). According to our WikiFramework-based approach, such questions match a single pattern (i.e. one relation). Before running

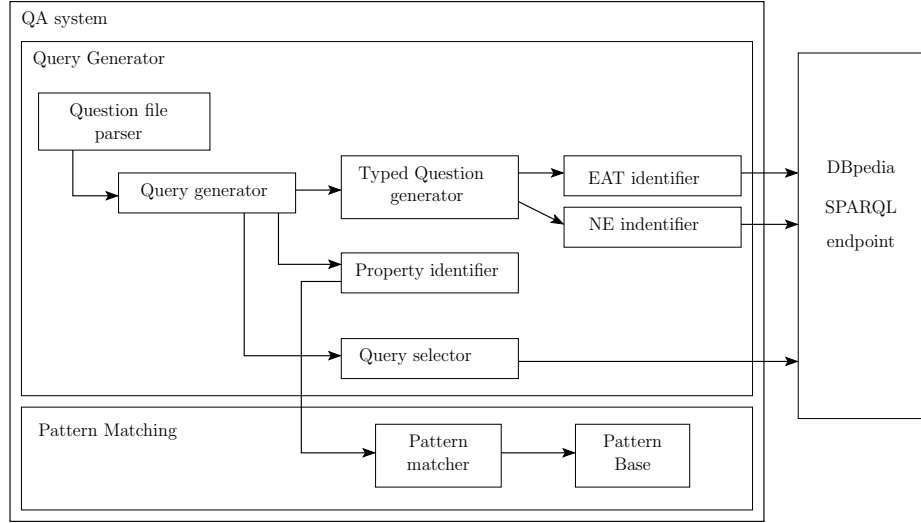


Fig. 2. QAKiS architecture.

the pattern matcher component, we replace *i)* the NE present in the question by its types, and *ii)* the question keywords by the Expected Answer Type (EAT), obtaining what we call a *typed question* (e.g. [River] does the [Bridge] cross?). The answer can then be retrieved with a SPARQL query over a single triple. In the following, each step of the system work-flow is explained in details.

EAT identification To implement a generic method for QA, we do not take advantage of the attribute *answertype* provided by the QALD-2 organizers for each question, to detect the EAT. Instead we implemented simple heuristics to infer it from the question keyword, i.e. if the question starts with:

- “When”, the EAT is [Date] or [Time];
- “Who”, the EAT is [Person] or [Organisation];
- “Where”, the EAT is [Place];
- “How many”, the EAT is [nonNegativeNumber];
- either “Which”, “Give me all”, “List all”, we expect the EAT to be the terms following the question keyword (we consider the longest term matching a class label in the DBpedia ontology).

NE identification Three strategies are followed, applied in a cascade order (i.e. if a strategy fails, we try the next one):

1. we run the Named Entity Recognizer in Stanford Core NLP to identify the NEs. If the question contains the label of an instance from the DBpedia ontology, and this label contains one of the NEs identified by Stanford, then

we keep the longest expression containing it. This is usually the best option, however we noticed that for many expressions containing “of” (e.g. *mayor of Berlin*), it was better to keep only what comes after “of” (i.e. *Berlin*), otherwise the page “Mayor of Berlin” (present in Wikipedia) is selected. In such cases, we extract both NEs;

2. if Stanford does not recognize any NE, we look if it recognizes at least one proper noun (whose Part of Speech is NNP). If this is the case, we take the longest label in the DBpedia ontology containing it;
3. if the two first strategies fail, we just look for the longest DBpedia instance label found in the question.

Typed questions generation Once the set of EATs and NEs are identified for a question, the typed questions are generated by replacing the question keywords by the supertypes of the EAT, and the NE by its types. For instance, for the question “Who is the husband of Amanda Palmer?” (id=42, test set), as EAT both [Person] or [Organisation] (both of them subclasses of [owl:Thing]) are considered. Moreover, the NE *Amanda Palmer* has type [MusicalArtist], [Artist] and [owl:Thing]. All 9 typed questions are generated.

WikiFramework pattern matching Before pattern matching, stopwords in the typed questions are deleted, and stems, lemmas and tokens are extracted. A Word Overlap algorithm is then applied to compare the stems, lemmas, and tokens of questions with the patterns and keywords for each relation. A similarity score is provided for each match: the highest represents the most likely relation.

3.3 Generation of SPARQL queries

A set of patterns is retrieved by the pattern matcher component for each typed question (we restrict the number of patterns retrieved to 5 per typed question). The patterns retrieved for all the typed questions are sorted by decreasing matching score. For each of them, one or two SPARQL queries are generated, either *i)* `select ?s where{?s <property> <NE>}`, *ii)* `select ?s where{<NE> <property> ?s}` or *iii)* both, according to the compatibility between their types and the property domain and range. Such queries are then sent to the SPARQL endpoint. As soon as a query gets results, we return it; otherwise we try with the next pattern, until a satisfactory query is found or no more patterns are retrieved.

4 Results on QALD-2 data

In this section we present and discuss QAKiS’ performances on extracting correct answers for natural language questions (provided by the QALD-2 challenge) from DBpedia. Given a certain question, QAKiS carries out a fully automatic process

(Section 3), and as output specifies both a SPARQL query and the answers, as obtained querying the SPARQL endpoint provided by the challenge organizers.⁵ Table 2 reports on the evaluation both on QALD-2 training and test set, with respect to precision, recall and f-measure. Furthermore, statistics are provided with respect to the number of questions for which QAKiS found an answer, and among them, the number of correct and partially correct answers.

Table 2. QAKiS performances on DBpedia data sets

	DBpedia	
	Training set	Test set
Precision	0.476	0.39
Recall	0.479	0.37
F-measure	0.477	0.38
<i># answered questions</i>	40/100	35/100
<i># right answers</i>	17/40	11/35
<i># partially right answers</i>	4/40	4/35

4.1 Error analysis and discussion

As introduced before, the current version of QAKiS addresses questions expressing only one relation between the subject (the entity in the question) and the object of the relation (the Expected Answer Type). More precisely, for now the subject must be a NE (that QAKiS recognizes as described in Section 3.2). Even if the strategies we defined for NER seem to correctly recognize most of the NEs present in the questions, a module for coreference resolution should be added to capture e.g. the DBpedia entry *Juliana_of_the_Netherlands* from the question: *In which city was the former Dutch queen Juliana buried?* (id=89, test set). Moreover, the strategy we added to deal with cases such as *the mayor of Berlin*, i.e. consider as NE both the one recognized by Stanford NER and the longest instance of the DBpedia ontology containing it, fails to correctly capture the book title in *Who wrote the book The pillars of the Earth?*, since patterns that match with a higher score are found for *Earth* (the NE recognized by Stanford).

Most of QAKiS' mistakes concern wrong pattern matching, i.e. the highest similarity between a typed question (generated as described in Section 3.2) and patterns in the pattern repository, is provided for patterns expressing the wrong relation. As described in Section 3, such similarity is calculated using a Word Overlap algorithm. We plan to substitute such algorithm with more sophisticated approaches, to consider also the syntactic structure of the question.

Another problem we faced concerns the fact that patterns (and questions) can be ambiguous, i.e. two questions using the same surface forms can in fact

⁵ <http://greententacle.techfak.uni-bielefeld.de:5171/sparql>

refer to different relations in the DBpedia ontology (e.g. *Who is the owner of Universal Studios?* relies on the relation `owner` for answer retrieval, while in *Who owns Aldi?* the correct answer is the subject of the relation `keyPerson`). A possible solution could be clustering similar relations (with several patterns in common), so that the system tries to find an answer for all the relations in the cluster, and selects the relation providing a non null answer.

Considering the ontology we rely on for relation extraction and answer retrieval, QAKiS addresses only the questions tagged by QALD-2 organizers as `onlydbo:true` (i.e. where the query relies solely on concepts from the DBpedia ontology). Surprisingly, in a few cases we were also able to provide correct answers for questions tagged `onlydbo:false`, e.g. in *What is the time zone of Salt Lake City?* (id=58 test set), matching the relation `timeZone`, or in *Give me all video games published by Mean Hamster Software* (id=71, training set), matching the relation `publisher` (since such company develops only video games).

Most of the partially correct answers we provide concern questions considering more than one relation, but for which QAKiS detects only one of them (due to the actual version of the algorithm). For instance, for *Give me all people that were born in Vienna and died in Berlin.* (id=19, test set), we retrieve and list all the people that died in Berlin. We plan to target such kind of questions in a short time, since the two relations are easily separable. On the contrary, more complex strategies should be thought to deal with questions with a more complex syntactical structure, as *Who is the daughter of Bill Clinton married to?* (id=3, training set), for which at the moment we answer with the name of Bill Clinton’s wife (we match only the relation `spouse`).

We plan short term solutions for boolean questions as *Is the wife of president Obama called Michelle?* (id=7, training set): we correctly match the relation `spouse` but we provide as answer *Michelle_Obama*, instead of the boolean *true*.

5 Conclusions

This paper describes our first participation to the QALD-2 open challenge using the QAKiS system, a QA system over linked data (in particular, DBpedia) based on the WikiFramework approach. In general, the results we obtained using the proposed approach are in line with other systems’ results, fostering future research in this direction. Due to the high variability and complexity of the task, much work is still to be done, and improvements should be planned on different fronts: *i*) extending the WikiFramework pattern extraction algorithm following [6], *ii*) improving the NE detection strategy, and addressing also questions whose subject is a class and not a specific NE (e.g. *What is the longest river?* (id=15, test set); *iii*) investigating the applicability of the Textual Entailment approach (a framework for applied semantics, where linguistic objects are mapped by means of semantic inferences at a textual level [1]) to improve the question-pattern matching algorithm; *iv*) addressing boolean and questions requiring more than one relation, to increase the system coverage. Finally, future work will also address the issue of natural language answer generation.

References

1. Dagan, I., Dolan, B., Magnini, B., Roth, D. (2009), Recognizing textual entailment: Rational, evaluation and approaches, in JNLE vol. 15 (4), pp. i-xvii.
2. Gerber, D., Ngonga Ngomo, A.C. (2011), Bootstrapping the Linked Data Web, in 1st Workshop on Web Scale Knowledge Extraction ISWC 2011, Bonn, Germany.
3. Ferrandez, O., Spurk, C., Kouylekov, M., Dornescu, I., Ferrandez, S., Negri, M., Izquierdo, R., Tomas, D., Orasan, C., Neumann, G., Magnini, B., Vicedo, J.L. (2011), The QALL-ME Framework: A specifiable-domain multilingual Question Answering architecture, in Web Semantics: Science, Services and Agents on the World Wide Web journal, vol. 9 (2), pp. 137-145.
4. Lopez V., Uren V., Sabou, M., Motta, E. (2011), Is Question Answering fit for the Semantic Web?: a Survey, in Semantic Web - Interoperability, Usability, Applicability journal, vol. 2(2), pp. 125-155.
5. Mahendra, R., Wanzare, L., Bernardi, R., Lavelli, A., Magnini, B. (2011), Acquiring Relational Patterns from Wikipedia: A Case Study, in Proceedings of the 5th Language and Technology Conference, Poznan, Poland.
6. Wu F., Weld, D.S. (2010), Open information extraction using Wikipedia, in Proceedings of ACL, Uppsala, Sweden.

A System Description of Natural Language Query over DBpedia

Nitish Aggarwal and Paul Buitelaar

Unit for Natural Language Processing, Digital Enterprise Research Institute,
National University of Ireland, Galway
{nitish.aggarwal,paul.buitelaar}@deri.org

Abstract. This paper describes our system, which is developed as a first step towards implementing a methodology for natural language querying over semantic structured information (semantic web). This work focuses on interpretation of natural language queries (NL-Query) to facilitate querying over Linked Data. This interpretation includes query annotation with Linked Data concepts (classes and instances), a deep linguistic analysis and semantic similarity/relatedness to generate potential SPARQL queries for a given NL-Query. We evaluate our approach on QALD-2 test dataset and achieve a F1 score of 0.46, an average precision of 0.44 and an average recall of 0.48.

Introduction

The rapid growth of Linked Data offers a wealth of semantic data for facilitating a interactive way to access the Web. However, Linked Data also brings several challenges in providing a flexible access over the Web for all users. Structured query languages like SPARQL provide the capability of accessing these this data, but these languages are restricted to the vocabulary defining the data. This data should be easily searchable and consumable for casual users to query in their native language, similar as with traditional web of documents through web search engines for document search.

In order to facilitate NL-queries over Linked Data, we implemented a basic pipeline that includes entity annotation, a deep linguistic analysis and semantic similarity/relatedness. This pipeline is very similar to the system implemented by Freitas A. et.al. [1], which is based on the a combination of entity search, a Wikipedia-based semantic relatedness (using Explicit Semantic Analysis) measure and spreading activation. However, our work focuses additionally on a deep linguistic analysis and categorization of a NL-Query. For example, a given NL-Query, such as "who designed the Brooklyn Bridge", is first categorized as a person- type query and, then the verb "designed" is modified to "designer". We also further investigate the approaches used for computing semantic similarity and relatedness.

Query Interpretation Approach

In our system, the interpretation of NL-Query is driven by semantic matching between Linked Data vocabulary and terms appearing in the NL-Query, to construct a SPARQL query. A well- interpreted SPARQL query from a given NL-Query can overcome the semantic gap between user- described queries and Linked Data vocabularies.

This includes three components:, namely query annotation, a deep linguistic analysis and semantic similarity/relatedness as shown in Fig.1. We describe below these components by taking an example NL-Query over the DBpedia dataset.

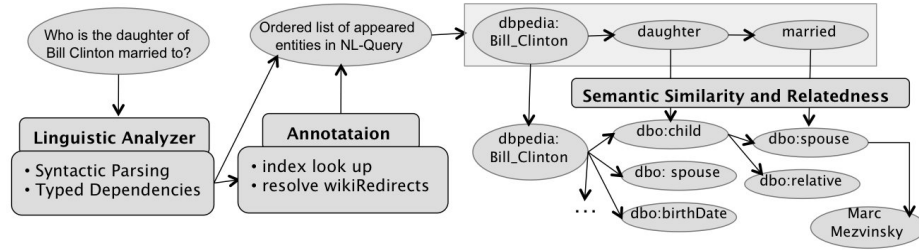


Fig. 1. Query interpretation pipeline for an example NL-Query "Who is the daughter of Bill Clinton married to?".

Query Annotation The interpretation process starts by identifying the potential entities, i.e. DBpedia instances and classes present in the NL-Query. For identifying these entities, we created two separate lucene indices, one for labels & URIs of all DBpedia instances and other one for all DBpedia classes. Annotating a NL-Query includes the extraction of keywords by removing stop words and identification of possible DBpedia classes followed by identification of DBpedia resources by performing keyword search over both lucene indices. After identifying potential resource labels, we perform disambiguation to recognize the most appropriate DBpedia resource URI, as there are multiple URIs for the same DBpedia resource label. The disambiguation is performed by retrieving wikiPageRedirects URIs, if the recognized URI redirects to any other DBpedia resource URI, e.g. in our system "Bill Clinton" is identified as URI "http://dbpedia.org/resource/BillClinton" which redirects to the right URI of label "Bill Clinton" i.e. "http://dbpedia.org/resource/Bill_Clinton".

Linguistic Analysis A deep linguistic analysis is performed by generating a parse tree and typed dependencies by using the Stanford parser. Generated parse trees provides a phrase extraction for identifying them as potential DBpedia resources or DBpedia classes. For instance, in our example query, the phrase "Bill Clinton" is identified as a noun phrase. It suggests us to perform a lucene search

over the whole phrase "Bill Clinton" rather than separate searches for "Bill" and "Clinton".

We convert the given NL-Query into an ordered list of potential terms by using typed dependencies generated by the Stanford parser. For creating this ordered list, first we select a central term among all the identified terms, where the central term is the most plausible term to start matching of a given NL-Query to the vocabulary appeared in the DBpedia graph. This selection is performed by prioritizing the DBpedia resources over DBpedia classes. Then, we retrieve the direct dependent terms of this central term following the generated typed dependencies and add them into the ordered list. Similarly, we perform the same for all the other terms in the list. For instance, in our example NL-Query, firstly, the system identifies "Bill Clinton" as a central term and then "daughter" as direct dependent of "Bill Clinton" followed by "married" as direct dependent of "daughter" shown in Fig.1.

Semantic Similarity and Relatedness A semantic similarity can be defined on the basis of taxonomic (is-a) relations of two concepts, while relatedness covers a broad range of relations, e.g. meronym and antonym. In our problem space, we want to get the best semantic match of terms appearing in the NL-Query to the vocabulary of the DBpedia dataset. We can not however rely solely on semantic similarity measures (as in our example NL-Query), as we can see relatedness can better map the term "married" on and the retrieved property "spouse" as they are semantically related terms but not semantically similar.

To find the best semantic match we are investigating two approaches for semantic relatedness, i.e. Wikipedia based Explicit Semantic Analysis (ESA) [2] and a semantic relatedness measure based on WordNet structure [3]. Due to the computational cost involved in getting the relatedness measure using ESA, currently we are experimenting with measures based on WordNet only.

Evaluation

To evaluate over approach, we calculate average precision, average recall and F1 score of the results obtained by our approach on QALD-2 test dataset, which includes 100 NL-Queries over DBpedia. The results are shown in Table 1.

Total	Answered	Right	Partially right	Avg. Precision	Avg. Recall	F1
100	80	32	7	0.44	0.48	0.46

Table 1. Evaluation on QALD-2 test dataset of 100 NL-Queries over DBpedia

Conclusion and Future Work

This paper presented a system for natural language querying over Linked Data, which includes query annotation, a deep linguistic analysis and semantic simi-

larity/relatedness. Currently, our approach does not fully explore all the types of queries appeared in dataset as it consists more challenging complex NL-Queries such as SPARQL aggregation and ask type queries. Future work will concentrate on improving the annotation step with better handling linguistic variations and a sophisticated semantic similarity/relatedness measures by that taking contextual information into account.

Acknowledgements

This work is supported in part by the European Union under Grant No. 248458 for the Monnet project and by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

1. Freitas, A., Oliveira, J. G., O’Riain, S., Curry, E., Da Silva, J. C. P.: Querying linked data using semantic relatedness: a vocabulary independent approach. In: Proc. of NLDB’11 (2011)
2. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: Proc. of IJCAI 2007
3. Pirró, G.: A semantic similarity metric combining features and intrinsic information content. *Data Knowl. Eng.* **68**(11) (2009) 1289–1308

TypeCraft

Collaborative databasing and Resource sharing for Linguists

Dorothee Beermann¹ and Pavel Mihaylov²

¹ Norwegian University of Science and Technology, Trondheim, Norway
`dorothee.beermann@hf.ntnu.no`

² Ontotext, Sofia, Bulgaria
`pavel@ontotext.com`

Abstract. We present a linguistic application that uses web technologies to promote the reuse of research data in the form of Interlinear Glossed Text (IGT), which is a well-established data format within philology and the structural and generative fields of linguistics. Here we present the modules and procedures of the online database TypeCraft.³ IGT is a sought after commodity in NLP and an integral part of scholarly linguistic work. It not rarely represents the only structured data available for less-resourced or endangered languages. While archiving of structured data from endangered languages is already well on its way [2], the free creation and exchange of linguistic data in the form of linked IGTs still needs to gain in popularity.

1 Introduction

With more linguists interested in the description of endangered and less-described languages, we see a rising interest in the electronic creation and management of linguistic data. Distinct from computational approaches to language annotation, where tagging of language data serves to facilitate automatic processing, annotation by linguists is meant for human consumption. TypeCraft (TC), the tool we are going to present here, is specialised on the creation, management and exchange of Interlinear Glossed Text (IGT), exemplified by an example from Runyankore Rukiga (ISO 639-3,nyn), a Lacustrine language of the Great Lakes area in Uganda:

Omu nju hakataahamu abagyenyi
òmù njù hākàtààhàmù àbàgyéngyì
Omu n ju ha ka taah a mu a ba gyenyi
in CL9 house CL16 PST enter IND LOC IV CL2 visitor
PREP N V N
'In the house entered visitors'

At present the main hindrance for IGTs to be a prime linguistic resource is the still prevalent lack of glossing standards. In-depth manual linguistic annotation is time consuming and often cyclic since it is an integral part of a scientific

³ www.typecraft.org

discovery process. A tool facilitating this process needs to allow for partial annotation. It further needs to take into consideration that linguistic annotation is often distributive work done in close collaboration between native speakers and linguists, so that a tool that can be used independently by all partners has to cater to the needs of users with different levels of linguistic expertise and computer literacy.

The discussion is organised as follows: In Section 2, we present the TypeCraft system from a user perspective. We discuss the creation and management of IGT and outline how collaborative online databasing can contribute to making linguistic data re-usable and mobile. In Section 3, we give a system description and in Section 4, we will summarise, and outline future work.

2 TypeCraft

TypeCraft (TC) [1] consists of a relational database combined with a tabular text editor for the computer-aided manual annotation of text. The tool's outer wrapper is a customised mediawiki which serves as a general entrance port and collaboration tool. The list below gives an overview over TC's main functionalities:

Annotation

Manual import of continuous text or sentence collections

Tabular interface for word level glossing with automatic sentence break-up and direct access to pre-defined gloss-sets

Easy access to gloss definitions and linking to an online ontology of grammatical concepts

Data management

Wiki facilitated management of primary data and metadata

Easy navigation between primary and annotated data

Easy linking of data to the TC-wiki for an integrated multi-media representation of data

Collaboration

Graded access and individual work spaces: the user decides which data is shared, TC texts and phrases have their own URL and thus can be acquired and exchanged freely online

User groups can share data

Co-editing of TC-wiki pages for early dissemination

Data export

Export of annotated sentences (individuals or sets) to Microsoft Word, Open Office and LaTeX for paper publications. Print-friendly versions of the TypeCraft web pages including exported database material

Export of XML for automatic data processing

2.1 General Information

While other manual annotation tools designed for normal linguists are desktop systems,⁴ TC is a multi-user online system. Users administer their own data in their private space at TC, but they can also make use of other users' shared data. TC is loaded by directing a browser⁵ to www.typecraft.org. We use standard wiki functionality to direct the user to the database via *New text*, *My texts*, and *Text- or Phrase search*. *My texts* displays the user's repository of annotated material,

⁴ The other tools are Toolbox and FLEEx, both developed by SIL.

⁵ TC runs in Firefox, Chrome, Safari and Opera.

called 'Texts'. *My texts*, the user's private space, is divided into two sections: *Own texts* and *Shared texts*. This reflects the graded access design of TC. Texts can be shared between groups. Data-sets can be easily exchanged between users and co-edited in user-groups. Interlinear Glosses can be loaded to the TC-wiki where they are displayed as part of a TC-wiki page. Under the printing of wiki pages TC data export automatically. One additional feature is that wiki-import from the TC database gets automatically updated when the database changes. As a collaborative tool, TC tries to utilise the effect of collaboration for the further standardisation of linguistic glosses. Glossing rules are conventional standards and one way to spread them is to make them easily accessible at the point where they are actively used. TC insists that annotators use a pre-defined set of glosses which are rooted in the Leipzig Glossing Rules [5], but have been extended to meet the needs of annotation in a more multi-lingual setting. We have grouped all glosses into annotation classes and mapped them to the GOLD (General Ontology for Linguistic Description) via URI-pointers. GOLD has been created as a tool to facilitate a more standardised use of basic grammatical features [3]. Through the linking to GOLD, TC users can relate annotations to grammatical concepts and to find relevant bibliographic resources as they annotate.

At present TC has 300 users, and the number is growing at a steady pace. We started 2012 with 13 000 annotated phrases from 97 mostly African languages. Since 2005 the TypeCraft system is under constant development and the planning of new development is based on the feed-back from system users and interested colleagues. Most of the active TypeCraft users are junior linguists and graduate students, mostly working on a less-resourced languages. TypeCraft is user driven. At present 54.2% of the data in the TC database remains private due to pending publication. However, also part of that data is partially represented on TC-wiki pages most of which were created by users of the system. A feature liked by many users is the possibility to join an annotation group. Through workshops and regular classes in linguistic text annotation new user groups can discover the tool. In addition, TypeCraft has proven to be a useful tool for e-learning for linguists. The TC wiki *Classroom* namespace contains several examples of classroom collaborations involving TC.

2.2 The TypeCraft Editor

After having imported a text into the TC Editor, which is easily accessed from the TC sites navigation bar (*New text*), the text is run through a simple, but efficient sentence splitter. The user can then select via mouse click one of the phrases to enter into the annotation mode. The system distinguishes between translational, functional and part-of-speech glosses. Properties can be assigned to linguistic phrases as well as to words or morphemes. Under Construction description the user can add notes and write down open questions which are stored together with the rest of the annotations in the XML-schema. The possibility to report and retrieve notes is an important feature of an annotation tool which is in particular needed at the beginning of an analysis, for collaborative annotation and when new layers of annotation are added to previous annotations.

The annotation interface is a table. Information is ordered horizontally (the phrase and the free translation) and vertically, so that words and morphs are aligned with their baseform and their part of speech, as well as their gloss. TC assumes that free class morphemes are annotated for meaning while closed class items receive a gloss. Phrases in TypeCraft are Url encoded data and thus can be exchanged with other web applications. For further processing TC data can be exported as XML.

The system prompts the user for the Lazy Annotation Mode.⁶ Manual annotation is further added by making TC-tags accessible from the TCwiki navigation bar as well as by drop-down menu under annotation. Other important resources, such as Ethnologue [4] can be directly reached from the TC Editor.

Since TypeCraft uses Unicode, every script that the user can produce on the PC can be entered into the browser. As of recent, TypeCraft offers the transliteration of Mandarin Chinese and Telugu, a Dravidian language of India, to make data that has been entered into the database using the original script more widely accessible. The export of data to the main text editors is one of the services that TC offers. TC tokens can be exported to Microsoft Word, OpenOffice.org Writer and LaTeX, which allows easy integration of databased material into the user's favourite text editing software. Although annotating in TypeCraft is still time consuming it is the re-usability of data that pays off. Export can be selected from the text editing window or from the SEARCH interface.

2.3 TypeCraft Search

A TypeCraft search operates on phrases, which means that the result of a query is a phrase level representation. Each line (or block) of the search result represents a linguistic phrase. In our experience, search results consisting of lists of sentences can be evaluated more easily by humans than lines of concordances. Search results come in two flavours, either as lines of sentences which allow a first quick scan of the data, or as blocks of IGTs which give the linguist access to the sentence internal annotations. Using general browser functionality, search results can be easily scanned. TypeCraft allows for complex searches on several tiers from the search interface where words or morphemes queries can relatively freely be combined with a search for specific glosses or combinations of glosses co-occurring either in a phrase, or on a word or a morpheme. Visitors of TC can at present search freely in the 45% of the private data on TypeCraft that has been made available for general search.

3 System Description

TypeCraft is based on a remote server and a web-based client running locally in the user's browser. The various components of the system and their interactions are shown in Figure 1.

⁶ Lazy Annotation Mode (LAM) is a function that automatically enriches annotation tables with word related information already known to the database.

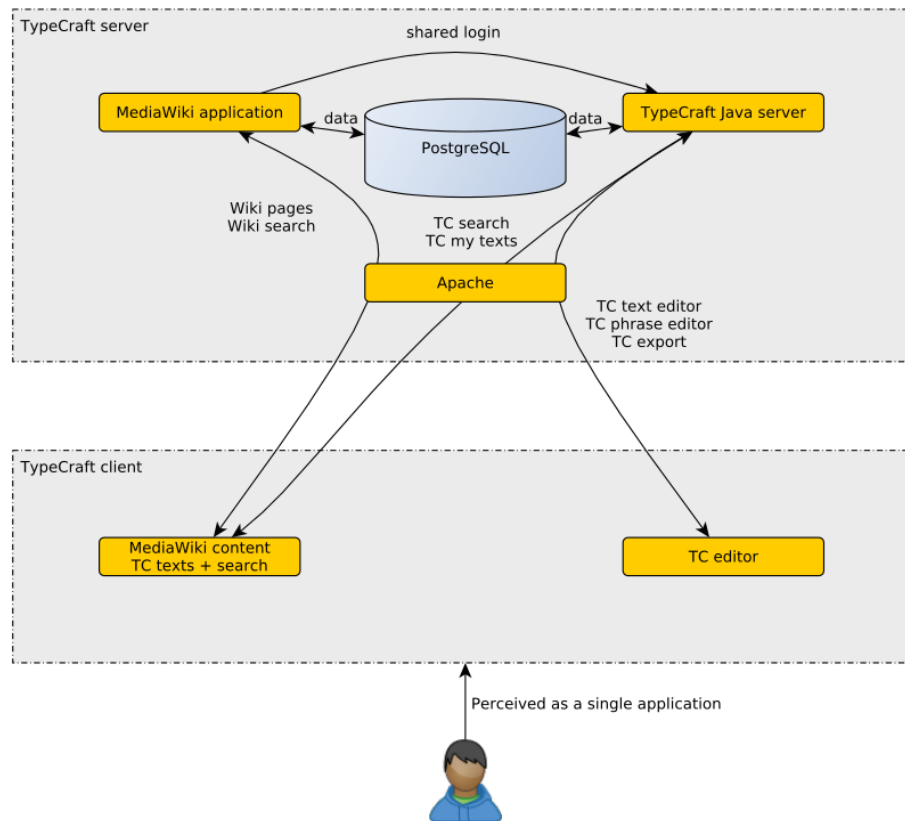


Fig. 1. TypeCraft architecture

3.1 Server

The TC server consists of a Java server application running in Tomcat, MediaWiki running in Apache, and a PostgreSQL database. The Apache server acts as a single entry point for both MediaWiki and the TC server application. MediaWiki has been extended with various extensions to enable specific TC functionality, such as *My texts*, *Phrase search* and *TC search*. The login is handled by MediaWiki but once authorised it will be shared with the TC server application. This guarantees TC will only divulge data for which the logged in user has permission.

Storage and data model TC uses PostgreSQL for data storage. The data mapping between Java objects and database tables is managed by Hibernate so the system is not bound to any specific SQL database. TC data can be divided into two specific groups:

- Common data: pos tags, gloss tags, global tags, ISO 639-3 languages. This is data shared between all annotated tokens and users.
- Individual data: texts, phrases, words and morphemes, together with their annotation. This is data specific to each user.

Individual data items reference common data items. E.g. everyone who uses the pos tag N will share the reference to a single common tag N.

3.2 Client

The user interacts with TC through a web-based interface. The interface consists of the customised MediaWiki content and a text-and-phrase editor (TC editor). The MediaWiki content provides wiki pages, wiki search, as well as the TC-specific functions mentioned previously. The editor is used to edit texts and phrases and assign annotation to phrases. It is written entirely in JavaScript and builds a GUI using HTML elements. The editor opens in separate browser window and is not directly connected to the MediaWiki content. The present GUI uses the YUI library and a large amount of TC-specific code. An important point is that for the end-user, TC is a single web application integrating the wiki and the editor.

4 Summary and Outlook

Setting it apart from the other main linguistic tools in its category, TC focuses on collaboration and the exchange of annotated linguistic data in the form of IGT. TC is low-tech on the outside. Non-computer-oriented linguists, language specialists and language communities can start using it almost instantly. While other linguistic software makes use of forums, blogs and other social software to service their user communities, TypeCraft *IS* social software. It is a powerful tool, however its real potential resides in its user community. Active users promote the creation and exchange of IGT, they link their data and contribute in this way to the availability, connectedness and quality of linguistic data.

One of the difficulties we faced over time was how to extend the data model when there are new requirements, or when a specific group needs a slightly different annotation model. Another issue is how to integrate TC annotations with third-party annotations on another level of the same data, e.g. audio and video. We believe the right way to solve these issues is to switch to an RDF-based data model that will allow us to be flexible and also allow us to benefit from existing Linked Open Data (e.g. reference annotated tokens to WordNet). This will also open possibilities to create open RDF-based standards for linking heterogeneous annotations.

References

1. Beermann, D., Mihaylov, P.: e-Research for Linguists. In: Proc. of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (2011) 24–32
2. Broeder, D., Sloetjes, H., Trilsbeek, P., Van Uytvanck, D., Windhouwer, M., Wittenburg, P.: Evolving challenges in archiving and data infrastructures. In: Nau, H. N., Schnell, S., Wegener, C. (eds.): Documenting endangered languages: Achievements and perspectives (2011) 33–54
3. Farrar, S., Langendoen, T.: A linguistic ontology for the Semantic Web. *GLOT International* **7**(3) (2003) 97–100
4. Ethnologue: Languages of the World. <http://www.ethnologue.com/web.asp>
5. Leipzig Glossing Rules. Max Planck Gesellschaft (2010)
<http://www.eva.mpg.de/lingua/resources/glossing-rules.php>

Author Index

Aggarwal, Nitish, 96	Gracia, Roberto, 21
Aprosio, Alessio Palmero, 87	
Beerman, Dorothee, 100	Harth, Andreas, 35
Brunetti, Josep Maria, 21	Kämpgen, Benedikt, 35
Buitelaar, Paul, 96	
Cabrio, Elena, 87	Lavelli, Alberto, 87
Ciravegna, Fabio, 60	Magnini, Bernardo, 87
Cojan, Julien, 87	Mihaylov, Pavel, 100
d'Aquin, Mathieu, 9	
Düwiger, Holger, 74	O'Riain, Sean, 35
Elbedweihy, Khadija, 60	Simperl, Elena, 49
Ell, Basil, 49	
Gandon, Fabien, 87	Vrandečić, Denny, 49
Gerlach, Martin, 74	Wendt, Matthias, 74
Gil, Rosa, 21	Wrigley, Stuart N., 60