Photo of the Oriental Pearl Tower by Tom Thai (flickr user eviltomthai)

Proceedings of the ISWC 2010 Posters & Demonstrations Track:

ISWC 2010 Posters & Demos

Collected Abstracts

November 7-11, 2010

Editors

Axel Polleres
Huajun Chen

Preface

On behalf of the program committee for ISWC2010 Posters & Demonstrations Track, it is our great pleasure to present these proceedings, which form a collection of abstracts describing the presented posters and demos presented at the conference. The posters and demonstrations track of ISWC 2010 continues the established tradition of providing an interaction and connection opportunity for researchers and practitioners to present and demonstrate their new and innovative work-in-progress. The track gives conference attendees a way to learn about novel on-going research projects that might not yet be complete, but whose preliminary results are already interesting. The track also provides presenters with an excellent opportunity to obtain feedback from their peers in an informal setting from knowledgeable sources. New in this year, we also encouraged authors of accepted full research or in-use papers to present a practical demonstration or poster with additional results.

In total, there were initially 90 submissions, of which we accept 28 regular demos, 6 demos accompanying full papers, as well as 19 posters.

Apart from the authors of all these contributions, we would like to thank all the members of the program committee as well as the additional reviewers who have spent lots of their valuable time within a very tight schedule in reviewing the submissions and organizing this session. We thank all of these dedicated people for their valuable discussions and feedback, and we wholeheartedly appreciate their voluntary and enthusiastic cooperation. We are convinced to have arrived at an inspiring mix of posters and demos which tackle the Semantic Web idea from various angles and are looking foward to an exciting session at the conference.

Lastly, we want to thank our fellow organizers of ISWC, foremost the general chair, Ian Horrocks, as well as particularly Peter Patel-Schneider and Birte Glimm for their help with easychair and in compiling these proceedings.

November 2010 Axel Polleres & Huajun Chen

Posters & Demonstrations Track – Organization

Program Chairs

Axel Polleres	DERI, National University of Ireland, Galway
Huajun Chen	CCNT, Advanced Computing and System Laboratory, China

Program Committee

Faisal Alkhateeb
Renzo Angles
Sören Auer
Diego Berrueta
Eva Blomqvist
Fernando Bobillo
Paolo Bouquet
Irene Celino
Kei Cheung
Oscar Corcho
Gianluca Correndo
Philippe Cudre-Mauroux
Mathieu d'Aquin
Danica Damljanović
Emanuele Della Valle
Klaas Dellschaft
Gianluca Demartini
Birte Glimm
Karthik Gomadam
Andreas Harth
Olaf Hartig
Michael Hausenblas
Aidan Hogan
Katja Hose
Giovambattista Ianni
Luigi Iannone
Michael Kohlhase
Thomas Krennwallner
Reto Krummenacher
Holger Lewen
Areti Manataki
Elena Montiel-Ponsoda
Knud Möller
Yuan Ni
Martin O'Connor
Ignazio Palmisano
Jeff Z. Pan
Alexandre Passant
Carlos Pedrinaci
Valentina Presutti
Jorge Pérez
Yves Raimond
Jinhai Rao
Marta Sabou
Simon Schenk
Wolf Siberski
Patrick Sinclair
Mari Carmen Suárez-Figueroa
Maria Esther Vidal
Yimin Wang
Gregory Williams
Guo Tong Xie

Additional Reviewers

Sofia Angeletou
Marcelo Arenas
Paul Buitelaar
Lorenz Bühmann
Richard Cyganiak
Laura Dragan
Timofey Ermilov
Benjamin Heitmann
Sebastian Hellmann
Marcel Karnstedt
Jacek Kopecky
Danh Le Phuoc
Dong Liu
Nuno Lopes
Uta Lösch
Marco Marano
Alessandra Martello
Philipp Obermeier
Antje Schultz
Jürgen Umbrich
Andreas Wagner

Table of Contents

Part I. Posters

Parallelization Techniques for Semantic Web Reasoning Applications

Alexey Cheptsov, Matthias Assel

[1] HLRS - High Performance Computing Center Stuttgart, University of Stuttgart,
Nobelstrasse 19, 70569 Stuttgart, Germany
{cheptsov, assel}@hlrs.de

Abstract. Performance is the most critical aspect towards achieving high scalability of Semantic Web reasoning applications, and considerably limits the application areas of them. There is still a deep mismatch between the requirements for reasoning on a Web scale and performance of the existing reasoning engines. The performance limitation can be considerably reduced by utilizing such large-scale e-Infrastructures as LarKC - the Large Knowledge Collider - an experimental platform for massive distributed incomplete reasoning, which offers several innovative approaches removing the scalability barriers, in particularly, by enabling transparent access to HPC systems. Efficient utilization of such resources is facilitated by means of parallelization being the major element for accomplishing performance and scalability of semantic applications. Here we discuss application of some emerging parallelization strategies and show the benefits obtained by using such systems as LarKC.

Keywords: Semantic Web Reasoning, LarKC, parallelization, multi-threading, message-passing

1 Introduction

Current Semantic Web reasoning systems do not scale to the requirements of the rapidly increasing amount of data, such as those coming from millions of sensors and mobile devices or the terabytes of scientific data produced by automated experimentation.

The latest attempts to overcome the above-mentioned limitations resulted in infrastructures for large-scale semantic reasoning, such as one set up by LarKC (the Large Knowledge Collider [1]) which focuses on reasoning over billions of structured data in heterogeneous data sets. Along with a number of original solutions for obtaining Web scale by semantic applications, LarKC offers services for transparently accessing diverse computing architectures, including multi-core (many-core) multi-processor, and cluster-based computer architectures as well as dedicated high-performance computers.

Parallelization enables simultaneous execution of independent computational operations and thus resolves the conflicts occurring between the concurrent operations

while performing computation. Given the large problem sizes that are addressed by LarKC, and considering the benefits of parallelization, it seems natural to explore use of the main parallelization strategies for semantic applications, too. Here we discuss some major parallelization techniques for providing parallelism on task-, instruction-, and data-level, applied for LarKC's pilot applications. However, the investigated approaches and techniques are quite generic and can be potentially applied for any other Semantic Web engine.

2 Parallelization Patterns

There are several parallelization techniques, which have proven their usability for a wide range of optimization tasks and might be beneficial for semantic applications. They can be roughly classified according to the level at which the parallelism takes place (Fig. 1):

1) between loosely-coupled components (workflow level) – implementation of parallelism by running multiple instances of the same plug-in simultaneously
   - task-level parallelism
     - *workflow branching*
2) within a separate component ("plug-in" level) – implementation of parallelism in the concurrent regions of the component's algorithms
   - instruction-level parallelism
     - shared-memory systems: *multi-threading*
     - distributed-memory systems: *message-passing*
   - data and instruction-level parallelism
     - *MapReduce data processing*

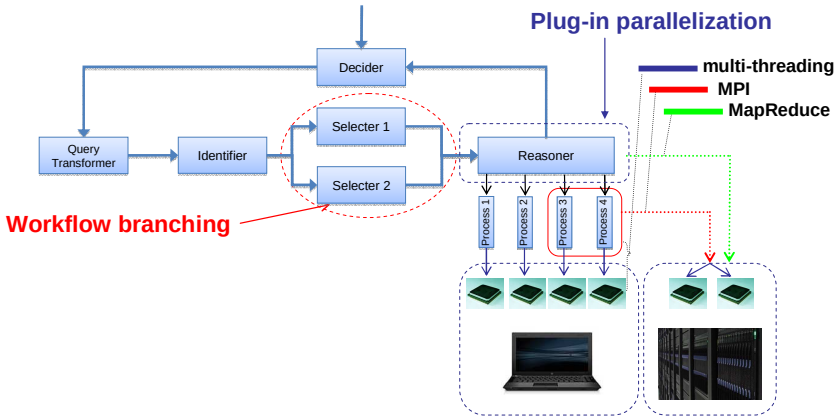


Fig. 1. For the semantic applications, which are described through complex workflows, parallelization can be applied on different levels: workflows (task- and data-level parallelism), or single components/plug-ins (data- and instruction-level parallelism).

3 Main Instruction-Level Parallelization Techniques

The techniques presented in Section 2 differ by complexity of their implementation and obtained performance impact. In this section we discuss only the approaches, which allow obtaining considerable performance impact with a minimum of implementation efforts for sequential code. In particular, we consider multi-threading and message-passing. The achieved performance impact is discussed as well.

3.1 Multi-threading

Most of today's CPUs are equipped with multiple cores. Unfortunately, many applications are still using only one of them for their processing (i.e., applications are still sequentially programmed) instead of distributing particular tasks to different processor cores concurrently. In order to make use of the capabilities provided by modern CPU architectures, applications must align their tasks according to the number of available cores.

Implementation of multi-threading for a sequential code is to large extent trivial and does not require much development efforts. For evaluation purposes, we implemented multi-threading support for the Urban Computing application of LarKC [2]. Realization of multi-threading for the most time consuming component of the investigated workflow allowed us to obtain a considerable performance speed-up (Table 1).

Table 1. Performance characteristics after applying multi-threading

Tested realization	Intel @ 1.8 GHz, 2 cores		Xeon @ 2.8 GHz, 8 cores	
	Time, ms	% of total execution	Time, ms	% of total execution
Single thread	4400	80	3400	74
Multiple threads	1100	37	900	36
Speed-up, times	**3.8**		**3.7**	

3.2 Message-Passing

The Message-Passing Interface (MPI) is the most widely used parallel programming paradigm for highly-scalable parallel applications. MPI enables sharing the application workload over various nodes of a parallel system (both shared and distributed memory architectures are supported). The synchronization between the nodes is achieved by means of the messages passed among the involved processes through the network interconnect. Implementations of MPI in Java (such as MPIJava or MPJ-Express) have enabled use of MPI also for Java applications. MPI is highly beneficial for computing-intensive applications, whereby scalability within a shared-memory space is not sufficient for obtaining the necessary performance.

For evaluation purposes, we implemented message-passing for the "Airhead" library from the S-Space package[1]. The parallelization technique was evaluated for the Linked Life Data subset used by University of Sheffield within the LarKC project. The obtained performance characteristics, collected in Table 2, prove great benefit of distributed-memory parallelisation not only for the investigated application, but also for similar ones coming from other areas of the Semantic Web.

Table 2. Performance characteristics after applying message-passing

Number of computing nodes	Intel @ 1.8 GHz, 2 cores		Xeon @ 2.8 GHz, 8 cores	
	Time, s.	Speed-up (to 1 CPU case)	Time, s.	Speed-up (to 1 CPU case)
1	750	1	57	1
2	-	-	20	**2.85**
4	-	-	10	**5.7**
8	-	-	5	**11.4**

4 Conclusions

In our tests we investigated the impact of the main instruction-level parallelization strategies, namely multi-threading and message-passing, on performance of two typical Semantic Web use cases. The first application was taken from the urban computing use case, where parallelization facilitates meeting real-time requirements. Whereas message-passing was not very useful for this application due to real-time performance requirements, applying multi-threading allowed the application to greatly benefit from the multi-core CPU architecture. The second application - random indexing - was much more complex as the first one, and made great benefit of message-passing that leveraged a cluster of shared-memory nodes for the application. Our future investigations will concentrate on further approaches presented here (such as MapReduce [3]) as well as hybrid algorithms combining them (e.g. multi-threading inside a shared-memory node combined with message-passing among nodes).

References

1. Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Della Valle, E., et al. Towards LarKC: A Platform for Web-Scale Reasoning, In: Proceedings of the 2008 IEEE international Conference on Semantic Computing ICSC, pp. 524--529, IEEE Computer Society (2008)
2. Della Valle, E., Celino, I., Dell'Aglio, D. The Experience of Realizing a Semantic Web Urban Computing Application, Transactions in GIS 14,2 (2010)
3. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F. Scalable Distributed Reasoning Using MapReduce. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009, LNCS, vol. 5823, pp. 634--649, Springer (2009)

[1] http://code.google.com/p/airhead-research/

Visual Reasoning about Ontologies

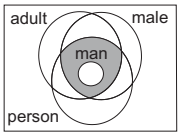
John Howse[1], Gem Stapleton[1], and Ian Oliver[2]

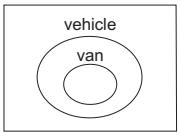
[1] University of Brighton, UK {john.howse,g.e.stapleton}@brighton.ac.uk
[2] Nokia, Helsinki, Finland ian.oliver@nokia.com

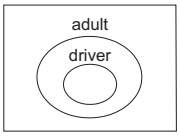
Abstract. We explore a diagrammatic logic suitable for specifying ontologies using a case study. Diagrammatic reasoning is used to establish consequences of the ontology.

Introduction. The primary (formal) notations for ontology modelling are symbolic, such as description logics or OWL [2]. The provision of symbolic notations, along with highly efficient reasoning support, facilitates ontology specification, but need not be accessible to the broad range of users. Using diagrammatic notations for reasoning, in addition to specification, can bring benefits. Standard ontology editors often support a visualization; Protégé includes a plug-in visualization package, OWLVis, that shows derived hierarchical relationships between the concepts in the ontology and, thus, is very limited. Currently, some diagrammatic notations have been used for specifying ontologies, but they are either not formalized [3] or do not offer many of the benefits that good diagrammatic notations afford [4]. In [6], we proposed ontology diagrams, which we now rename *concept diagrams*, for ontology modelling. We extend [6] by demonstrating how one can reason using concept diagrams.

Ontology Specification. We use a variation of the University of Manchester's People Ontology [1] as a case study. It relates people, their pets and their vehicles. We now formally define the ontology. The diagrams below assert: (a) a man is an adult male person, (b) every van is a vehicle, and (c) every driver is an adult:

(a): 

(b): 

(c): 

In (a), the shading asserts that the set man is equal to the intersection of the sets adult, male and person. Also, (d) every animal is a pet of some set of people:

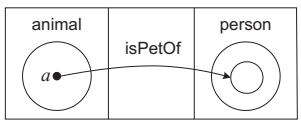
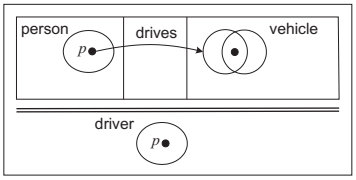
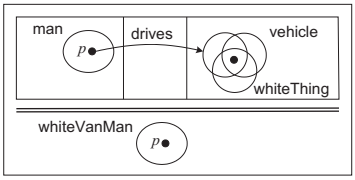
(d): 

Diagram (d) asserts that the relation *isPetOf* relates animals to people, and only people: each *animal* a is related by the relation *isPetOf* to a (possibly empty) subset of *people*. So, when a is instantiated as a particular element, e, the unlabelled curve represents the image of *isPetOf* with its domain restricted to $\{e\}$. As animal and person are not disjoint concepts – a person is an animal – the curves representing

these concepts are placed in separate sub-diagrams, so that no inference can be made about the relationship between them.

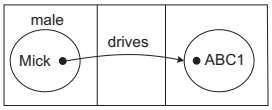
We define the concepts of being a *driver* and a *white van man*: (e) p is a person who drives some vehicle if and only if p is a driver, and (f) m is a man who drives a white van if and only if m is a white van man:

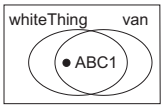
(e): 

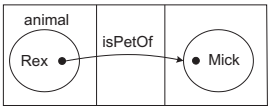
(f): 

The two parallel, horizontal lines mean *if and only if*; a single line means *implies*.

We now introduce an individual called Mick: (g) Mick is male and drives ABC1, (h) ABC1 is a white van, and (i) Rex an animal and is a pet of Mick:

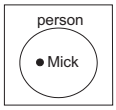
(g): 

(h): 

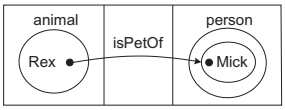
(i): 

Diagrammatic Reasoning. We have enough information to prove diagrammatically some lemmas, culminating in proving that Mick is a white van man.

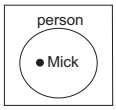
Lemma 1 Mick is a person:



Proof From diagram (i) and diagram (d) we deduce all of the individuals of which Rex is a pet are people:

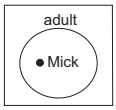


Therefore, Mick is a person, as required:

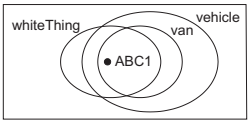


In the above proof, the deduction that the set of individuals of which Rex is a pet relied on pattern matching diagrams (i) and (d). We believe it is clear from the visualizations that one can make the given deduction. The last step in the proof simply deletes syntax from the diagram in the preceding step, thus weakening information, to give the desired conclusion. Much of the reasoning we shall demonstrate requires pattern matching and syntax deletion.

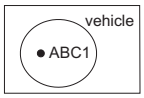
Lemma 2 Mick is an adult:



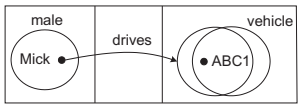
Proof From diagram (b) we know that all vans are vehicles so we deduce, from diagram (h):



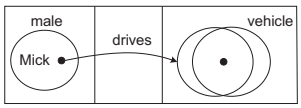
Therefore, ABC1 is a vehicle:



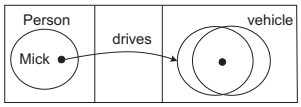
From diagram (g), we therefore deduce:



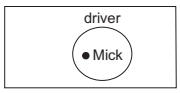
Now, ABC1 is a particular vehicle. Therefore, Mick drives some vehicle:



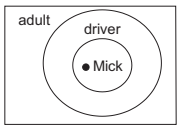
By lemma 1, Mick is a person, thus:



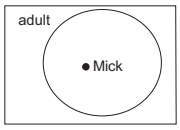
Hence, by diagram (e), Mick is a driver:



By diagram (c) drivers are adults:

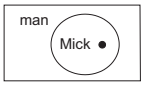


Hence, Mick is an adult, as required:

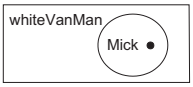


Lemma 3 follows from lemmas 1 and 2, together with diagrams (a) and (g) (the interested reader may like to attempt the proof):

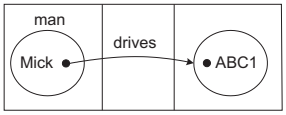
Lemma 3 Mick is a man:



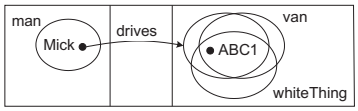
Theorem 1 Mick is a white van man:



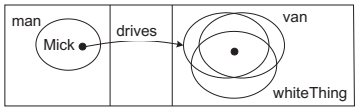
Proof By lemma 3, Mick is a man so we deduce, using diagram (g):



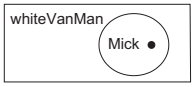
From diagram (h) we have:



Therefore Mick drives some white thing which is a van:



By diagram (f), we conclude that Mick is a white van man:



The visual reasoning we have demonstrated in the proofs of the lemmas and the theorem is of an intuitive style and each deduction step can be proved sound. We argue that intuitiveness follows from the syntactic properties of the diagrams reflecting the semantics. For instance, because containment at the syntactic level reflects containment at the semantic level, one can use intuition about the semantics when manipulating the syntax in an inference step. This is, perhaps, a primary advantage of reasoning with a well-designed diagrammatic logic.

Conclusion. We have demonstrated how to reason with concept diagrams. The ability to support visual reasoning should increase the accessibility of inference steps, leading to better or more appropriate ontology specifications: exploring the consequences of an ontology can reveal unintended properties or behaviour. These revelations permit the ontology to be improved so that it better models the domain of interest. Our next step is to formalize the inference rules that we have demonstrated and prove their soundness. Ideally, these rules will be intuitive to human users, meaning that people can better understand why entailments hold. This complements current work on computing justifications [5] which aims to produce minimal sets of axioms from which an entailment holds; finding minimal sets allows users to focus on the information that is relevant to the deduction in question which is important when dealing with ontologies containing very many axioms. Using a visual syntax with which to communicate *why* the entailment holds (i.e. providing a diagrammatic proof) may allow significant insight beyond knowing the axioms from which a statement can be deduced.

Acknowledgement. Supported by EPSRC grants EP/H012311, EP/H048480. Thanks to Manchester's Information Management Group for helpful discussions.

References

1. http://owl.cs.manchester.ac.uk/2009/iswc-exptut, 2009.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nadi, and P. Patel-Schneider (eds). *The Description Logic Handbook*. CUP, 2003.
3. S. Brockmams, R. Volz, A. Eberhart, and P. Löffler. Visual modeling of OWL DL ontologies using UML. *Int. Semantic Web Conference*, 198–213. Springer, 2004.
4. F. Dau and P. Ekland. A diagrammatic reasoning system for the description logic $\mathcal{ALC}$. *Journal of Visual Languages and Computing*, 19(5):539–573, 2008.
5. M. Horridge, B. Parsia, and U. Sattler. Computing explanations for entailments in description logic based ontologies. In *16th Automated Reasoning Workshop*, 2009.
6. I. Oliver, J. Howse, E. Nuutila, and S. Törmä. Visualizing and specifying ontologies using diagrammatic logics. In *Australasian Ontologies Workshop*, 2009.

Semantic-based Complex Event Processing in the AAL Domain

Yongchun Xu, Peter Wolf, Nenad Stojanovic and Hans-Jörg Happel

FZI, Research Center for Information Technology, Haid-und-Neu-Str. 10-14,
76131 Karlsruhe, Germany
{xu, wolf, nstojano, happel}@fzi.de

Abstract. Ambient assisted living (AAL) is a new research area focusing on services that support people in their daily life with a particular focus on elderly people. In the AAL domain sensor technologies are used to identify situations that pose a risk to the assisted person (AP) or that indicate the need of proactive assistance. These situations of interest are detected by analyzing sensor data coming from a whole variety of sensors. Considering the need for immediate assistance especially in the case of safety- and health-critical situations, the detection of situations must be achieved in real-time. In this paper we propose to use Complex Event Processing (CEP) based on semantic technologies to detect typical AAL-like situations. In particular, we present how the ETALIS CEP engine can be used to detect situations in real-time and how this can lead to immediate and proper assistance even in critical situations in conjunction with the semantic AAL service platform openAAL.

Keywords: Complex Event Processing**,** Ambient assisted living, Real-time, Semantic Technologies, Context-aware, ETALIS, openAAL

1 Introduction

Ambient assisted living (AAL) is a newly research area focusing on services that support people in their daily life and particular focus on elderly people. The services include reminding and alerting the assisted person (AP), giving feedback, advice, and impulses for physical or social activities, among others. [1] In AAL, sensor technologies monitor devices, environmental conditions, health parameters and location information. This information is used to build a knowledge base capturing the current situation of the AP. This situational knowledge is used by context-aware services to provide personalized assistance, i.e. services can adapt to the current situation of the user.

Semantic technologies that enable modeling of complex situations, easy integration of sensor data and automatic service discovery are considered to be a perfect fit for enabling context-aware services in the AAL domain. However, considering the need for immediate assistance especially in the case of safety- and health-critical situations, the detection of situations must be achieved in real-time.

Complex event processing (CEP), a software technology for the dynamic processing of high volume events, can be considered a perfect match for detecting critical situations. With CEP, it is possible to express causal, temporal, spatial and other relations between events. These relationships specify patterns in which the event stream is searched in real-time. [2]

In this paper, we combine the AAL service platform openAAL, that acts as central gateway to sensor events and AAL services, with the ETALIS [3] engine for complex event processing based on semantic technologies to detect critical situations based on sensor events in real-time.

2 Semantic-based Event Processing in AAL

Figure 1 shows the architecture of semantic-based event processing in the AAL domain. The system is based on the OSGi service middleware and consists of two main sub systems the service platform openAAL and the ETALIS CEP system, which are combined by the linkage component.

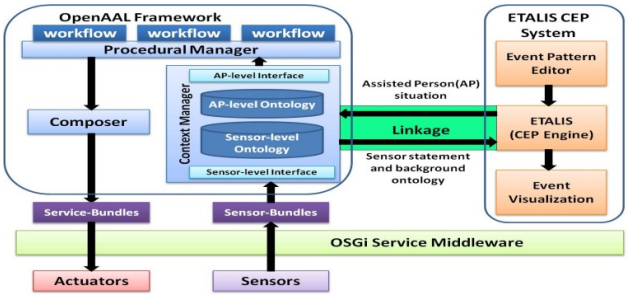


Fig. 1. The Architecture of Semantic-based Event Processing in the AAL domain

2.1 OpenAAL

OpenAAL[1] [1] represents a flexible and powerful middleware for AAL scenarios. The openAAL platform enables easy implementation, configuration and situation-dependent provision of flexible, context-aware and personalized IT services. It consists of three components: procedural manager, composer and context manager.

The context manager provides a sensor-level interface to connect with sensor-bundles. Sensor bundles provide sensors events which are saved in a context store. Both context store and sensor-level interface are derived from a sensor ontology. Situations of the assisted person (AP) are modeled by complex event patterns and are detected through CEP based on sensor events and background knowledge. The procedural manager manages and executes workflows, which are triggered as reaction

[1] OpenAAL is available as open source implementation at http://openaal.org

to certain situations. The composer selects and combines services from the set of currently available services to achieve service goals defined in the workflow.

2.2 ETALIS CEP System

We use CEP to detect critical situation based on sensor events in real-time. The ETALIS CEP system consists of three components: Event Pattern Editor, ETALIS CEP engine and Event visualization. The core component is the ETALIS engine[2], which is based on a declarative semantics, grounded in Logic Programming. Complex events are derived from simpler events by means of deductive rules. Due to its root in logic, ETALIS engine also supports reasoning about events, context, and real-time complex situations (i.e., Knowledge-based Event Processing), which is a very important feature for the event processing in the AAL domain. In addition to reasoning about sensor-events, ETALIS can also incorporate useful background knowledge into its reasoning. Moreover, ETALIS supports work with out-of-order events[3], which might occur often in a sensor network.

ETALIS detects safety- and health-critical situations of APs from sensor events according to the complex event patterns that can be defined by users through the event pattern editor. The event pattern editor provides a user friendly user interface enabling easy definition of patterns. The event visualization displays the event activities in the CEP system. Visualization enables a contextual analysis of emerging complex events and supports the decision making process (e.g. how to react on a particular situation).

2.3 Linkage

We combine both sub systems (openAAL and ETALIS) by creating a linkage component. The linkage component transforms sensor events and useful background knowledge (both represented in openAAL in a RDFS-like ontology) into the corresponding ETALIS formats and transmits the information. The linkage component also implements an adapter for ETALIS CEP engine to enable running ETALIS in a JAVA environment while ETALIS is implemented in Prolog.

3 Use Case and Validation

The benefits of CEP lie in the efficient real-time detection of complex events. In the AAL domain these aspects are needed when it comes to situations where assistance needs to be provided immediately.

Our use case is based on the problem area of forgetfulness which has been identified as one of the main problems among the elderly. [4] Due to forgetfulness a critical situation can emerge when devices like iron, stove and oven are forgotten

[2] ETALIS engine is open source and can be downloaded from http://code.google.com/p/etalis/.
[3] An out-of-order event is the event which has been registered with a delay

when leaving the home. Our idea is to identify such a safety-critical situation by means of CEP and then immediately remind the AP when he/she is still at home; thus allowing the person to intervene. In more severe cases of forgetfulness critical devices can be automatically turned off.

This use case can be accomplished by the interaction between openAAL and ETALIS. openAAL with its attached sensors is able to provide sensor information that can be used to detect such a critical situation. Sensors used in this scenario include electricity consumption sensors, door sensors, identification sensors (e.g. RFID), location sensors to distinguish leaving from entering the home and a sensor detecting the ringing of a door bell (the latter to enhance detection reliability). All those sensor events are sent to the ETALIS event engine which is identifying the critical situation by means of complex event patterns. Note that for a reliable detection of a person leaving the right order of events is especially important; also background knowledge can be integrated for further increasing detection reliability. In addition to the real-time properties, the ETALIS system is also supporting design and maintenance of complex event patterns which can help to reduce the creation of wrong or bad patterns. Once this situation has been detected, services running locally on the openAAL platform can be executed to immediately remind the AP of the potentially critical situation using a touch screen display right next to the front door.

4 Conclusion

In this paper we introduced intelligent semantic-based complex event processing in AAL using the ETALIS event engine to detect critical situations in real-time. In this approach sensor data coming from a variety of sensors is processed by ETALIS to detect situations modeled as complex event patterns. Based on this real-time situation detection, services, running on a local service platform, can provide proper and immediate assistance. As a next step we will evaluate the implemented system in a real world setting and improve the system according to the results of the evaluation. Furthermore, we would like to apply the system in other domains that require real-time processing especially in high-load environments.

Reference

[1] Wolf, P., Schmidt, A., Klein, M. **Applying Semantic Technologies for Context-Aware AAL Services: What we can learn from SOPRANO** In: Workshop on Applications of Semantic Technologies 09, Informatik 2009, Lecture Notes in Informatics vol. , GI, 2009

[2] Luckham, D. The **Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems**. Addison-Wesley, 2002

[3] Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., Studer, R. **A Rule-Based Language for Complex Event Processing and Reasoning,** RR 2010

[4] Larrabee, G. J., Crook III, T. H. **Estimated Prevalence of Age-Associated Memory Impairment Derived From Standardized Tests of Memory Function.** International Psychogeriatrics (1994), **6**:**1**:95-104 Cambridge University Press

Efficient processing of large RDF streams using memory management algorithms

Vaibhav Khadilkar, Murat Kantarcioglu, Latifur Khan, and Bhavani Thuraisingham

The University of Texas at Dallas

Abstract. As more RDF streaming applications are being developed, there is a growing need for an efficient mechanism for storing and performing inference over these streams. In this poster, we present a tool that stores these streams in a unified model by combining memory and disk based mechanisms. We explore various memory management algorithms and disk-persistence strategies to optimize query performance. Our unified model produces an optimized query execution and inference performance for RDF streams that benefit from the advantages of using both, memory and disk.

1 Introduction

An application that processes RDF streams does not know *apriori* the size of the stream. This makes it difficult to store these streams in memory as only a limited amount of data can be stored, and on the disk which requires a longer query processing time for small streams. Research has thus far focused on storing RDF data in relational databases [1] or in non-database approaches [2]. We have a developed a tool[1] that presents a unified model based on an optimal combination of memory and disk based solutions for storing RDF streams. This tool allows users to pose any SPARQL query (non-inference or inference) to an application. Our tool is implemented as a Jena graph which is the basic building block of the Jena framework. Our graph moves from Jena's in-memory graph to a Lucene[2] graph when we begin to run out memory. The Lucene graph mirrors the in-memory graph and queries to the unified model are rewritten to query the Lucene indices. We use different memory management algorithms to select nodes from the RDF stream to be left in memory based on the frequency of access patterns and the centrality of nodes in the stream. We have also tested two Lucene index creation strategies to optimize query performance. Finally, we switch to a Jena RDB graph when a threshold limit is reached, beyond which the RDB graph is better than Lucene for non-inference queries. Reference [3] presents an excellent survey on using memory management algorithms in relational databases. We have selected Lucene only as a temporary storage mechanism because we will switch to the RDB graph if more data is streamed. Our proposed model works very well for query execution and inference tasks with RDF streams.

2 Proposed Architecture

Figure 1 shows a flow of control to store RDF streams using the unified approach. We begin by storing the RDF stream in Jena's in-memory triple store.

[1] http://jena.sourceforge.net/contrib/contributions.html,http://cs.utdallas.edu/semanticweb
[2] http://lucene.apache.org/java/docs/index.html

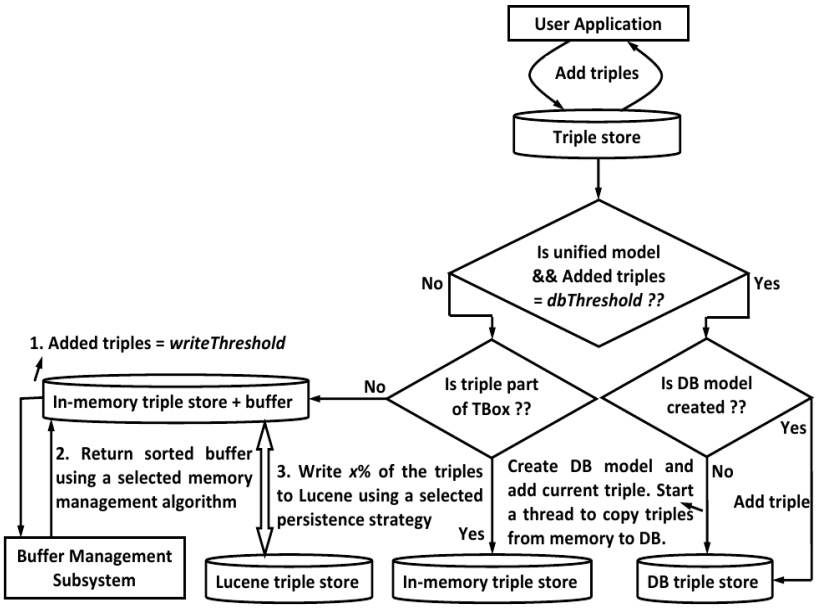


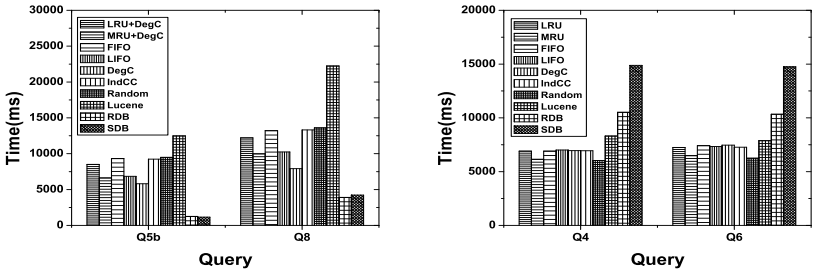
Fig. 1. Unified Approach Architecture - Creating a model

If the triple is an ABox triple we also store or update for every *subject* of each incoming triple, it's *degree*, a *timestamp* of when it was last accessed and a *pointer* to the triples belonging to it, in a memory buffer. The TBox triples are first read into memory without maintaining any statistics for them in the buffer. This helps to distinguish TBox triples from ABox triples. Since no statistics are maintained these TBox triples are never written to disk. When the *writeThreshold* is reached the buffer management subsystem returns a sorted buffer based on the selected memory management algorithm such as FIFO, LIFO, LRU, MRU and RANDOM. We also adapted social network centrality measures such as degree centrality (DC) and clustering coefficient (CC) [4] into memory management algorithms. The *writeThreshold* is defined as, $writeThreshold = initThreshold \times totalMem$. The equation takes *initThreshold* as a number of triples and the memory size (specified in gigabytes) given to the current run from the user. Triples from the in-memory graph are moved to the Lucene graph using the *pointer* of every *subject* node (from the sorted buffer) and the selected persistence strategy. This process of moving triples continues as long as $x\%$ of the *writeThreshold* is not reached (x is also user configurable). Finally, when the *dbThreshold* is reached we move all triples to Jena's RDB graph. From this point onwards all incoming triples are directly stored in the RDB graph. We use a combination of in-memory, Lucene and RDB graphs for non-inference models and a combination of in-memory and Lucene graphs for inference models. For query execution, the input query is submitted to the graph that is currently

being used. A non-inference query is run on either the in-memory and Lucene graphs or the RDB graph and a complete result is returned to the user. For an inference query, Pellet infers additional triples by reasoning over the result from the in-memory and Lucene graphs, using the TBox triples that are always in memory. The resulting triples are then returned to the user.

3 Experimental Results

We performed benchmark experiments to compare the performance of the unified model to both, the Jena database backends and a purely Lucene triple store. We used the Sp^{2}Bench [5] benchmark to check non-inference query execution and the LUBM [6] benchmark to test inference. Although we have tested all queries of both benchmarks on our system, in this section we show results only for Q5b and Q8 from Sp^{2}Bench and for Q4 and Q6 from LUBM as they are representative of the overall trend. The graphs below show only query time and they do not include loading times. We have also performed scalability tests with varying graph sizes, but in this poster we only show graph sizes of 50168 triples for Sp^{2}Bench and 1 university ($\approx$ 103000 triples) for LUBM. We use these small sizes since we only want to determine the best algorithm and Lucene persistence strategy in this paper. Further, we set $writeThreshold = (3/4) \times$ no. of triples in the graph, $totalMem = 1$ and $x =$ no. of triples in mem/90. We chose these values for the parameters so that we always have a good balance of triples between memory and Lucene giving us a good indication of the overall performance of various queries of both benchmarks.

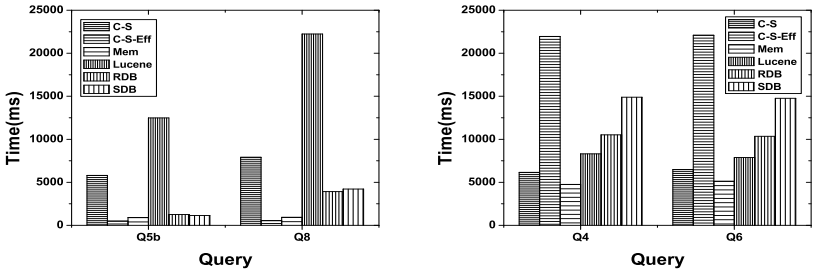


(a) Algorithms - Sp^{2}Bench (b) Algorithms - LUBM

Fig. 2. Comparison of all algorithms and persistence strategies

We have used the *degree* and *timestamp* values to implement the memory management algorithms. For example, if we use LRU, we sort the buffer in the increasing order of *timestamp* values while for degree centrality we sort the buffer in increasing order of DC values. The reader should note that the DC and CC values are recomputed for every node each time the buffer is sorted. We then move triples to the Lucene graph for every node starting from the top of the buffer until x% of the triples are moved. We have also combined LRU and MRU with both DC and CC by first using the *timestamp* to sort the buffer and if there is a tie we use DC or CC to break the tie. Figure 2 shows a comparison of all memory management algorithms that we have tested for Sp^{2}Bench and LUBM.

For Sp^{2}Bench, we see that *DC* gives us the best result because it keeps nodes that are relevant to the Sp^{2}Bench queries in memory. In comparison, for LUBM, we see that MRU performs the best. This is due to the fact that MRU leaves the least recently used nodes in memory that are used by the Pellet reasoner for inference and query execution.



(a) Persistence strategies - Sp^{2}Bench (b) Persistence strategies - LUBM

Fig. 3. Comparison of persistence strategies

We have also tested two Lucene persistence strategies, the first creates all indices at the same time (C-S) while the second creates each index as needed starting with the predicate, then the object and finally the subject (C-S-Eff). In the unified model we do not create the subject Lucene index, instead we set *dbThreshold* to be the number of triples at this point. The C-S strategy works well with LUBM queries but does not work for the Sp^{2}Bench queries as shown in figure 3. With C-S-Eff we get a query time comparable to in-memory storage for Sp^{2}Bench, but a much higher query time for LUBM. For LUBM, the Pellet reasoner needs to query the larger predicate Lucene index multiple times, making it slower than the C-S approach where the reasoner needs to query the smaller subject and object Lucene indices. The Sp^{2}Bench queries use the in-memory subject and object structures, and hence perform as well as the in-memory model.

4 Conclusion

In this paper we show that creating a unified model by combining the in-memory, Lucene and relational database models gives us excellent query execution and inference time with an enhanced scalability for RDF streams.

References

1. Dave Beckett. Scalability and Storage: Survey of Free Software/Open Source RDF storage systems. `http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/`, July 2002.
2. AllegroGraph RDFStore. `http://www.franz.com/agraph/allegrograph/`, 2005.
3. Hong-Tai Chou and David J. DeWitt. An Evaluation of Buffer Management Strategies for Relational Database Systems. In *VLDB*, pages 127–141, 1985.
4. M. O. Jackson. *Social and Economic Networks*. Princeton University Press, 2008.
5. Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. SP^{2}Bench: A SPARQL Performance Benchmark. In *ICDE*, pages 222–233, 2009.
6. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

Text Based Similarity Metrics and Deltas for Semantic Web Graphs

Krishnamurthy Koduvayur Viswanathan and Tim Finin
University of Maryland, Baltimore County
Baltimore, MD 21250 USA
{krishna3, finin}@umbc.edu

Abstract. Recognizing that two Semantic Web documents or graphs are similar and characterizing their differences is useful in many tasks, including retrieval, updating, version control and knowledge base editing. We describe several text-based similarity metrics that characterize the relation between Semantic Web graphs and evaluate these metrics for three specific cases of similarity: similarity in classes and properties, similarity disregarding differences in base-URIs, and versioning relationship. We apply these techniques for a specific use case – generating a delta between versions of a Semantic Web graph. We have evaluated our system on several tasks using a collection of graphs from the archive of the Swoogle Semantic Web search engine.

Keywords: Semantic Web graphs, similarity metrics, delta

1 Introduction and Motivation

Semantic Web search engines can benefit from recognizing nearly duplicate documents [2] for many of the same reasons that text search engines do. Comparing Semantic Web documents or graphs, however, is more complicated. In natural language text local word order is important to meaning while the order of triples in a Semantic Web document (SWD) is not important. As a result, equivalent Semantic Web documents may have completely different statement ordering. It is also possible to have two different SWDs, which become identical after performing inference. The presence of "blank nodes" in SWDs further complicates their comparison.

We explore three different ways in which a pair of Semantic Web graphs can be similar to each other: similarity in classes and properties used while differing only in literal content, difference only in base-URI, and implicit versioning relationship [4, 3]. We define text-based similarity metrics to characterize the relation between them. As a part of this, we identify whether they may be different versions of the same graph. Furthermore, if we determine a versioning relationship between a candidate pair, then we generate a *delta*, i.e., a detailed description of their differences at the triple level delta between them.

These methods enable a Semantic Web search engine to organize its query results into groups of documents that are similar with respect to the different metrics and also generate deltas between documents judged to have a versioning relationship.

2 Objective and Approach

Our objective is identify pairs of documents that are similar to each other in a collection of SWDs. We characterize SWD similarity along three dimensions: a similarity in classes and properties used while differing only in literal content, difference only in base-URI, and versioning relationship. For pairs of SWDs exhibiting a versioning relationship we compute a triple-level delta for them.

Our input corpus is in the form of a set of RDF documents. Our approach involves the following steps:

Convert to canonical representation: We convert all the documents into a uniform n-triple serialization. Since equivalent semantic web graphs may have different n-triple serializations, we apply the algorithm described in [4] which assigns consistent IDs to blank nodes and lexicographically order the triples.

Generate reduced forms: In order to compute the similarity measures, the canonical representations are decomposed into the following four reduced forms. Each is a document with:

1. only the literals from the canonicalized n-triples file
2. literals replaced by the empty string[1]
3. the base-URI of every node replaced by the empty string
4. literals and the base-URI of every node are replaced by the empty string

Thus, each Semantic Web graph has a canonical representation, and four reduced forms i.e. five forms in all.

Compute similarity measures: Given the text-based reduced forms, we can use the the following similarity measures from the information retrieval field.

1. **Jaccard similarity and containment**: We construct sets of character 4-grams from the input documents which are used to compute the measures. A high value for both Jaccard and containment metrics indicates a strong possibility of a versioning or equivalence relation between two.
2. **Cosine Similarity between semantic features**: Each SWD is represented as a vector of terms. The non-blank, non-literal nodes from each SWD are extracted and their term-frequency in the SWD is used as the feature weight.
3. **Charikar's Simhash**: We compute the Hamming distance between the simhashes of the documents being compared. Simhash [1] is a locality sensitive hashing technique where the fingerprints of similar documents differ in a small number of bit positions.

Pairwise computation of metrics: Given an input of Semantic Web documents, we need to find all pairs that are similar to each other. The total number of metrics computed for each pair of SWDs is 17: two kinds of cosine similarity, and three other metrics for each reduced form pair. The process is as follows:

1. Compute the two cosine similarity values between the canonical representations (already generated) of both the SWDs.

[1] Another viable approach is to replace each literal string by its XSD data type

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.973	0.000	1.000	0.973	0.986	1.000	yes
	1.000	0.027	0.973	1.000	0.987	0.996	no
Weighted Avg.	0.986	0.014	0.987	0.986	0.986	0.998	

Table 1: Accuracy by class (classes and properties used), using Naive Bayes

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1.000	0.040	0.962	1.000	0.980	0.979	yes
	0.960	0.000	1.000	0.960	0.980	0.990	no
Weighted Avg.	0.980	0.020	0.981	0.980	0.980	0.985	

Table 2: Accuracy by class (pairs different only in base URI), using Naive Bayes

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.864	0.045	0.950	0.864	0.905	0.909	yes
	0.955	0.136	0.875	0.955	0.913	0.909	no
Weighted Avg.	0.909	0.091	0.913	0.909	0.909	0.909	

Table 3: Accuracy by class (versioning relationship), using SVM with linear kernel

.

2. If the cosine similarity values are below a pre-determined threshold, then eliminate this pair from further consideration, else add this pair to a list of candidate pairs. The threshold for this step was determined empirically by performing pilot experiments, and was set at 0.7.
3. For all candidate pairs, compute the remaining three pairwise similarity metrics for each reduced form.

Thus the cosine similarity metric is used as an initial filter to reduce the remaining computations. It is to be noted that this pairwise comparison approach entails a quadratic number of comparisons.

3 Classification and Delta

We trained three classifiers (one for each kind of similarity defined) with a dataset collected from the Swoogle, annotated with human judgements of the three kinds of similarity. Pairwise similarity measures are computed for each candidate pair in the labeled dataset and three different feature vectors (one for each classifier) are constructed for each candidate pair, using appropriate attributes. The attributes used are the similarity measures that have been computed. These classifiers are then used to detect the three forms of similarity that we have defined. For a list of attributes used for each classifier, see [4].

Once it is determined that two SWDs have a versioning relationship between them, we compute the set of statements that describe the change between successive versions of the SWD. For details on the specific deltas that we compute between two versions, see [4].

4 Evaluation and Conclusion

Our system is based on several informal types of similarity that we have observed among documents in Swoogle's repository. In addition, there exists no standard labeled dataset of similar Semantic Web documents that we could use for the purpose of evaluating our system. Hence we constructed a collection of Semantic Web documents from Swoogles Semantic Web archive and cache services. Swoogle periodically crawls the Semantic Web and maintains several snapshots for each indexed SWD. We added such versions to our data-set and labeled them as having a versioning relationship. We constructed a labeled dataset of 402 SWDs (over 161,000 candidate pairs) from Swoogle's Semantic Web archive and cache services[2]. The results of the classification process are as shown in Tables 1, 2, and 3.

The results shown in Tables 1, and 2 were obtained by performing a ten-fold stratified cross validation using the labeled dataset. Table 3 shows results of evaluation on a test set that was constructed in the same way as the training data-set. The high true positive rate for determining version relationships between SW graph pairs allows us to develop applications like generation of deltas. For details on attributes used for each classifier, refer to [4]

We developed techniques to recognize when two Semantic Web graphs are similar and characterize their difference in three ways. When the system detects a versioning relationship between a pair, we also generates a delta in terms of triples to be added or deleted. One of the future directions is to increase scalability. We intend to implement a principled filtering mechanism to reduce the number of pairwise comparisons. We might use the Billion Triples Challenge dataset for experiments[3]. We also plan to develop a global similarity and ranking scheme where, given a sample, we would be able to find and rank the most similar matches. This would require representation of the similarity measures by more complex schemes, e.g., lattices.

References

1. M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proc. of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, New York, NY, USA, 2002. ACM.
2. G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors, *WWW*, pages 141–150. ACM, 2007.
3. K. Viswanathan. Text Based Similarity Metrics and Delta for Semantic Web Graphs. Master's thesis, 1000 Hilltop Circle, June 2010.
4. K. Viswanathan and T. Finin. Text Based Similarity and Delta for Semantic Web Graphs. Technical report, UMBC, 1000 Hilltop Circle, August 2010.

[2] http://swoogle.umbc.edu/index.php?option=com_swoogle_service&service=archive
[3] http://km.aifb.kit.edu/projects/btc-2010/

Towards Stable Semantic Ontology Measurement

Yinglong Ma[1,2]

[1] School of Control and Computer Engineering
North China Electric Power University, Beijing 102206, P.R.China
[2] State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.R.China
yinglongma@gmail.com

Abstract. Stable semantic ontology measurement is crucial to obtain significant and comparable measurement results. In this paper, we present a summary of the definition of ontology measurement stability and the preprocessing for stable semantic ontology measurement from [5]. Meanwhile, we describe two existing ontology metrics. For each of them, we compare their stability from the perspectives of structural and semantic ontology measurements, respectively. The experiments show that some structural ontology measurements may be unusable in cases when we want to compare the measurements of different models, unless the pre-processing of the models is performed.

1 Introduction

In recent years, ontology engineers have proposed many ontology metrics for assessing ontology quality such as the literatures [1–4]. However, some proposed ontology metrics are to measure ontology structure instead of ontology semantics which are the nature of ontology. They only simply calculate the number of classes and class inheritances by some labels in ontologies such as `owl:Class` and `rdfs:subClassOf`, and do not consider possibly implicit semantic subsumption between (complex) classes. Most ontology metrics do not take into account the open world assumption (OWA) and the possible addition of implicit axioms, which will cause incomparable measurement results [5]. To use the same metric to measure the ontologies with the same semantic knowledge will bring about variable values. Such ontology metrics may be unstable.

2 Stable ontology measurement and preprocessing

An ontology can be regarded as a set of triples of the form `(s,p,o)`. The structural description of an ontology $\mathcal{O}$ is the set of explicitly represented triples in $\mathcal{O}$. The semantic description of $\mathcal{O}$ is the set that contains not only the structurally described triples, but also all implicit triples obtained by reasoning $\mathcal{O}$. Note that an ontology with the same semantic description possibly has multiple structural descriptions (including $\mathcal{O}$).

Definition 1. *Let $Sem(\mathcal{O})$ be the semantic description of an ontology $\mathcal{O}$. $Sem(\mathcal{O})$ has the multiple structural descriptions, denoted $Stru(\mathcal{O})$=$\{\mathcal{O}, \mathcal{O}_1, \cdots, \mathcal{O}_n\}$. A stable ontology measurement $\mathcal{M}$ is mapping, $\mathcal{M} : Stru(\mathcal{O}) \rightarrow \mathbf{R}$ such that $\mathcal{M}(\mathcal{O}) = \mathcal{M}(\mathcal{O}_1) = \cdots = \mathcal{M}(\mathcal{O}_n)$, where $\mathbf{R}$ is a nonempty set of real numbers.*

We summarize the preprocessing for stable ontology measurement from [5].

1) Naming all anonymous classes and all anonymous individuals. We can automatically detect the related labels and name anonymous classes. Anonymous individuals can be detected and named by class membership. The set of named concepts of Ontology $\mathcal{O}$ is denoted $\mathcal{C}_{\mathcal{O}} = \{C_1, \cdots, C_n\}$, where each C_i is unique, and is either an atomic concept or a named anonymous concept.

2) Eliminating cycles of concept subsumption such as $A \sqsubseteq A_1, \cdots, A_n \sqsubseteq A$, where A, $A_i (1 \leq i \leq n)$ are concepts. Once we detect such a cycle of concept subsumption in an ontology, we replace all cyclic concept subsumption axioms with $B \sqsubseteq A_i$ $(1 \leq i \leq n)$, where B is a new concept name for each cycle.

3) Instances explicitly asserted by class membership, object property and datatype property should be enriched as deeply as possible by reasoning the ontology $\mathcal{O}$.

4) Getting rid of possible transitivity relationships. We attempt to adopt a definition of axiom fanouts per concept to get rid of the possible transitivity relationships. The reason to do this is that a well-founded measurement theory should avoid the *double counting* problem, e.g., a measurement unit is counted more than once. Once some axioms are counted, then the axioms derived from these counted axioms should not be counted. In the following, we specifically discuss axiom fanouts per concept.

Definition 2. *$\forall C, D \in \mathcal{C}_{\mathcal{O}}$, C is directly subsumed by D, i.e., directly-subsumed-by(C, D), iff $\forall C, D \in \mathcal{C}_{\mathcal{O}}(C \sqsubseteq D \wedge \neg\exists C' \in \mathcal{C}_{\mathcal{O}}(C' \sqsubseteq D \wedge C \sqsubseteq C'))$.*

Definition 3. *$\forall C \in \mathcal{C}_{\mathcal{O}}$, the axiom fanouts of C are denoted $AF_C = \{D_1, \cdots, D_m\}$, where for each $D_i (1 \leq i \leq m \leq |\mathcal{C}_{\mathcal{O}}|)$, directly-subsumed-by$(D_i, C)$ holds, and $|\mathcal{C}_{\mathcal{O}}|$ represents the cardinality of $\mathcal{C}_{\mathcal{O}}$.*

In the following, we simply analyze the correction of the preprocessing. On one hand, as mentioned above, for an ontology $\mathcal{O}$, its semantic description $Sem(\mathcal{O})$ contains not only the structural description of $\mathcal{O}$, but also the implicitly expressed knowledge derived from $\mathcal{O}$. This means that, for any axiom or assertion α in $\mathcal{O}$, $\mathcal{O}$ implies α iff $Sem(\mathcal{O})$ implies α. On the other hand, the preprocessing for stable ontology measurement is terminable because Step 1), Step 2) and Step 3) will be terminated if there is no complex concept, cycle of concept subsumption, and unenriched concept in $\mathcal{O}$. At last, this can guarantee that $Sem(\mathcal{O})$ should be finite and unique no matter how the ontology $\mathcal{O}$ is represented. In the case, the measurement result for $\mathcal{O}$ will be invariable and stable if we measure $\mathcal{O}$ by $Sem(\mathcal{O})$. We can also obtain the following corollary.

Corollary 1. *An ontology measurement of ontology's semantic description is stable by using the preprocessing for comparing the measurements of different models.*

3 Proposal of two ontology metrics

A structural ontology measurement is just to measure the explicitly expressed ontology without applying the preprocessing. In contrast to a structural ontology measurement, a semantic ontology measurement is just to measure the quality of semantic description of original ontology by using the preprocessing. We describe two ontology metrics related to axiom fanouts for validating the stability of ontology measurement.

Metric 1: Average Axiom Fanouts per Concept (AAFC)

AAFC of Ontology $\mathcal{O}$ can be defined as follows: AAFC($\mathcal{O}$)=$\frac{\sum_{C \in \mathcal{C}_\mathcal{O}} AF_C}{|\mathcal{C}_\mathcal{O}|}$.

Metric 2: Average Depth of Concept Subsumption of Leaf Concepts (ADCS-LC)

A concept $RC \in \mathcal{C}_\mathcal{O}$ is a root one iff $\neg\exists C \in (\mathcal{C}_\mathcal{O} \setminus RC)$ such that $RC \sqsubseteq C$. A concept $LC \in \mathcal{C}_\mathcal{O}$ is a leaf one iff $\neg\exists C \in (\mathcal{C}_\mathcal{O} \setminus LC)$ such that $C \sqsubseteq LC$. The depth of path p, denoted $|p|$, is the total number of concepts in p. ADCS-LC of $\mathcal{O}$ can be defined as ADCS-LC($\mathcal{O}$)=$\frac{\sum_{p \in PS} |p|}{|LS|}$, where PS and LS are the set of all paths and the set of leaf concepts in ontology $\mathcal{O}$, respectively.

AFC and ADCS-LC are ontology metrics related to ontology fanouts which can be often used as the indicators of some ontology quality properties such as complexity and cohesion [1–4]. They can be used for structural or semantic ontology measurements.

4 Experiments and measurement stability analysis

The goal of the experiments was designed to compare the stability of the two ontology metrics from the perspectives of structural and semantic ontology measurements, respectively. The experimental settings were as follows. 1) We randomly searched the 10 testing ontologies by the search engine, Swoogle. They were evaluated for validating the stability of ontology measurement; and 2) For each of AAFC and ADCS-LC, we collected the values of their structural and semantic ontology measurements, respectively. The measurement values of AAFC and ADCS-LC were shown in Figure 1.

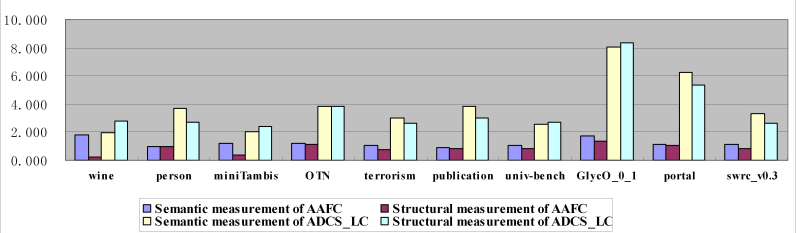


Fig. 1. Semantic and structural measurement values of AAFC and ADCS-LC

Pearson's correlation coefficient is used for analyzing the stability of AAFC and ADCS-LC, which is with the following hypotheses: $H_0 : \rho = 0$ (There is no correlation between the pair of values); $H_1 : \rho \neq 0$ (There is correlation between the pair of values). For each of AAFC and ADCS-LC, we calculated the correlation coefficient and p-value of pair of measurement values. The larger absolute value of the correlation coefficient means stronger correlation between the pair of variables. If the pair of variables are independent, the correlation coefficient is 0. P-values are used in hypothesis tests to either reject or fail to reject a null hypothesis. A small p-value indicates that a null hypothesis is false. A p-value ($<$0.001) means that we must reject the null hypothesis.

By the statistical software, SPSS , we can obtain the correlation coefficient and p-value of pairs of AAFC and ADCS-LC, respectively. We find that the correlation coefficient and p-value between the pair of semantic and structural measurement values of AAFC are -0.195 and 0.588, respectively. This means that there is no obvious correlation between the pair of AAFC. However, there is a very strong correlation between the pair of semantic and structural measurement values of ADCS-LC because their correlation coefficient and p-value are 0.946 and 0.000, respectively. Especially for the p-value, it is less than 0.001 such that we can obviously reject the null hypothesis.

From Corollary 1, if we we want to compare the measurements of different models, a semantic ontology measurement is stable by using the preprocessing. In the case, if the semantic and structural measurements of an ontology metric are strongly correlated, then this means that the structural measurement of the ontology metric may be usable to compare the measurement values of ontologies, and can be a useful indicator of some ontology quality properties such as complexity and cohesion. Otherwise, the structural measurement of the ontology metric is likely to be unusable. According to these analysis, we find that AAFC may be not usable to compare the measurement values of ontologies. In contrast to AAFC, ADCS-LC is usable for both semantic and structural ontology measurement. We believe that more experiments should be made to comprehensively validate the stability of ontology measurement.

5 Conclusions

We summarized the definition about stability of ontology metrics and the preprocessing for stable ontology measurement. By two ontology metrics to compare the measurement stability of different models, we found that some structural ontology measurements may be unusable, unless the pre-processing of the models is performed.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No.61001197), the Fundamental Research Funds for the Central Universities, and State Key Laboratory of Computer Science, IOS, Chinese Academy of Sciences (No.SYSKF1010).

References

1. Gangemi A., Catenacci C., Ciaramita C. and Lehmann J. A theoretical framework for ontology evaluation and validation. *Proceedings of SWAP2005*, 2005.
2. Tartir S., Arpinar I.B., Moore M., Sheth A.P. and Aleman-Meza B. OntoQA: Metric-based ontology quality analysis. *Proceedings of IEEE ICDE 2005 Workshop*, 2005.
3. Orme A., Yao H. and Etzkorn L. Coupling Metrics for Ontology-Based Systems. *IEEE Software*, **23(2)**, (2006) 102–108.
4. Orme A., Yao H. and Etzkorn L. Indicating ontology data quality, stability, and completeness throughout ontology evolution. *Journal of Software Maintenance and Evolution: Research and Experience*, **19(1)**, (2007) 49–75.
5. Vrandecic D. and Sure Y. How to Design Better Ontology Metrics. *Proceedings of ESWC2007*, (2007) 311–325.

T2LD: Interpreting and Representing Tables as Linked Data*

Varish Mulwad, Tim Finin, Zareen Syed, and Anupam Joshi
University of Maryland, Baltimore County, Baltimore MD USA 21250
{varish1,finin,joshi}@cs.umbc.edu,zareensyed@gmail.com

Abstract. We describe a framework and prototype system for interpreting tables and extracting entities and relations from them, and producing a linked data representation of the table's contents. This can be used to annotate the table or to add new facts to the linked data collection.
Keywords: linked data, human language technology, entity linking

1 Introduction

Vast amounts of information is available in structured forms like spreadsheets, database relations, and tables in documents found on the Web. We describe a framework for interpreting such tables and extracting entities and relations from them. The results can be used to annotate the table or to contribute its contents to linked data (LOD) collections. The process assigns column headers to classs from an appropriate ontology, links table cells (as appropriate) to entities in a LOD collection, and identifies relations between columns and links them to ontology properties. The resulting table interpretation can be used to confirm existing facts in an LOD collection and to propose new facts to be added.

Using the table headers and the data in its rows and columns, we query existing knowledge bases (KBs) including Wikitology [1] and DBpedia to select the best class labels for each column, which is then used to identify potential entity links for table cells. A classifier selects the best entity from the list and a second classifier decides whether the evidence is strong enough to link the cell value to it. Relations in the table are discovered using the generated column classes, linked entities, and KB information bases. Our implemented prototype was evaluated using a collection of tables from Google Squared, Wikipedia and tables found on the Web.

Caferella et al. [2] estimated that the Web contains around 14.1 billion HTML tables, over 154 million containing high quality relational data. This represents a huge source of knowledge currently unavailable on the Semantic Web. There is a need for systems that can automatically generate LOD from existing sources, be it unstructured (e.g., free text), semi-structured (e.g., text embedded in forms or Wikis) or structured (e.g., data in spreadsheets and databases). Interpreting tables is a problem of interest in many areas such as databases, web systems and the Semantic Web. See [3, 4] for a comprehensive study of related work on interpreting tables and converting databases and spreadsheets into RDF.

* Research supported in part by a gift from Microsoft Research, a Fulbright fellowship, NSF award IIS-0326460 and the Human Language Technology Center of Excellence.

Existing systems for extracting knowledge from tables [5] require human intervention and do not focus on a complete interpretation of the table, nor integrating the table with linked open data cloud. This poster paper focuses on an automatic framework for generating an linked RDF which can be integrated into the LOD cloud. The eventual goal of this work is to enrich the LOD cloud by learning new facts and knowledge from tables and publishing it on the Semantic Web.

To develop an overall interpretation of a table, we assign every column header a class label from an appropriate ontology, e.g., the column with header City is assigned a class label *dbpedia-owl:City* from the DBpedia ontology. For the table in Figure 1, we link "Baltimore" to *dbpedia:Baltimore*. Numbers can be mapped as values of properties which can be associated with entities in the table. We also identify the relations implicit between columns, e.g., that *dbpedia-owl:largestCity* seems to hold between the entities denoted by cell values in the first two columns (i.e., city and state). Finally this information is represented in a N3 serialization of RDF.

city	*state*	*mayor*	*population*
Baltimore	MD	S.Dixon	640000
Philadelphia	PA	M.Nutter	1500000
Washington	DC	A.Fenty	595000
New York	NY	M.Bloomberg	8400000
Boston	MA	T.Menino	610000

Fig. 1: In simple tables column headers suggests the type of data stored in columns and cell values denote instances of that type.

2 *T2LD* Framework

Given an table as input, the *T2LD* framework [6] begins with the process of assigning a class label to every column in the table. For all the cell values in every column of the table, the algorithm for assigning class labels (see Algorithm 1 in [3]) submits a complex query to the Wikitology knowledge base to determine the type of each cell value in the column. Each class label from the set of possible class labels obtained from query results is scored. The class label with the highest score is chosen as the class label to be associated with the column. We predict class labels from four vocabularies - DBpedia Ontology, Freebase, WordNet, and Yago.

Using the class labels as additional evidence, for every table cell, the algorithm for linking table cell to entities (see Algorithm 2 in [3] for detailed algorithm), re-queries the KB. For every table cell, the KB returns the top N possible entities. For each of the top N entities, the algorithm generates a feature vector consisting of the entity's KB score, entity's Wikipedia page length, entity's page rank, the Levenshtein distance between the entity and the string in the query and the Dice score between the entity and the string. The set feature vectors for each table cell are ranked using a SVM-Rank classifier. To the highest rank feature vector from SVM rank, two more features are added - the SVM rank score of the feature vector and the difference

MAP	columns
$m = 1$	11.53%
$0 < m < 1$	69.23%
$m = 0$	19.24%

Recall	columns
$r = 1$	46.15%
$0 < r < 1$	34.61%
$r = 0$	19.24%

Fig. 2: The percentage of columns with various MAP and recall scores.

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
@prefix dbpprop: <http://dbpedia.org/property/> .

"City"@en is rdfs:label of dbpedia-owl:City .
"State"@en is rdfs:label of dbpedia-owl:AdminstrativeRegion .

"Baltimore"@en is rdfs:label of dbpedia:Baltimore .
dbpedia:Baltimore a dbpedia-owl:City .
"MD"@en is rdfs:label of dbpedia:Maryland .
dbpedia:Maryland a dbpedia-owl:AdministrativeRegion .

dbpprop:LargestCity rdfs:domain dbpedia-owl:AdminstrativeRegion .
dbpprop:LargestCity rdfs:range dbpedia-owl:City .
```

Fig. 3: A example of N3 representation of a table as linked data

in SVM-Rank scores between the top two feature vectors. Based on this new feature vector, a second SVM classifier decides whether to link the table cell to this top ranked entity or not. If the evidence is not strong enough, it is likely that the table cell is a new entity not present in the KB; this step is useful in discovery of new entities in a given table. If the evidence is strong enough, the table cell is linked to the top ranked entity returned by SVM-Rank.

We also present a preliminary approach for identifying relations between table columns (see Algorithm 3 in [3]). The algorithm generates a set of candidate relations from the relations that exist between the strings in each row of the two columns. Each candidate relation is scored and the relation with the highest score is selected to represent relation between the two columns. We have also developed a preliminary template in N3 (see Figure 3), which is a compact and human readable serialization of RDF for representing tables as LOD.

3 Evaluation and Conclusion

Our implemented prototype was evaluated against 15 tables obtained from Google Squared, Wikipedia and from a collection of tables extracted from the Web. Excluding the columns with numbers, the 15 tables have 52 columns and 611 entities for evaluation of our algorithms. We used a subset of 23 columns for evaluation of relation identifcation between columns.

In the first evaluation of the algorithm for assigning class labels to columns, we compared the ranked list of possible class labels generated by the system against the list of possible class labels ranked by the evaluators. As shown in Figure 2 for *80.76*% of the columns the Mean Average Precision (MAP) between the system and evaluators list is greater than 0 which indicates that there was at least one relevant label in the top three of the system ranked list. Also seen in Figure 2, for *75*% of the columns, the recall of the algorithm was greater than or equal to 0.6. We also assessed whether our predicted class labels were reasonable based on the judgment of human subjects (see [3]). *76.92* % of the class labels predicted were considered correct by the evaluators. The accuracy in

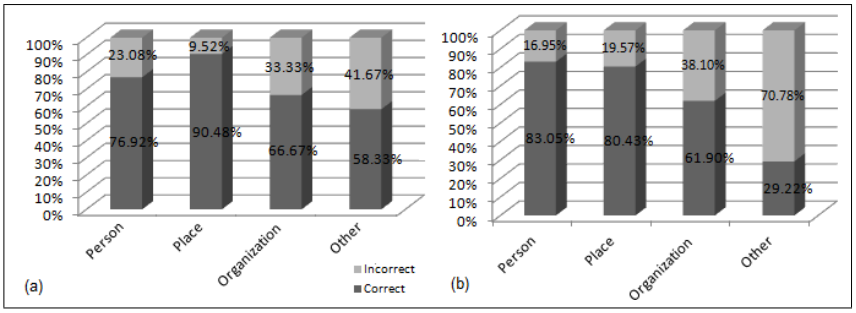


Fig. 4: Category wise accuracy for "column correctness" is shown in (a) and for entity linking in (b)

each of the four categories is shown in Figure 4. *66.12* % of the table cell strings were correctly linked by our algorithm for linking table cells. The breakdown of accuracy based on the categories is shown in Figure 4. Our dataset had 24 new entities and our algorithm was able to correctly predict for all the 24 entities as new entities not present in the KB. We did not get encouraging results for relationship identification with an accuracy of 25 % (see [3] for details).

Our existing system performs reasonably well in selecting appropriate types for columns and linking cell values to LOD entities. We have preliminary results for identifying and encoding the relationships implicit in the columns as well. Our current work is focused on improving relationship discovery and generating new facts and knowledge from tables that contain entities not present in the LOD knowledge bases.

References

1. Finin, T., Syed, Z.: Creating and Exploiting a Web of Semantic Data. In: Proc. 2nd Int. Conf. on Agents and Artificial Intelligence, Springer (2010)
2. Cafarella, M.J., Halevy, A.Y., Wang, Z.D., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. PVLDB **1** (2008) 538–549
3. Mulwad, V.: T2LD - An automatic framework for extracting, interpreting and representing tables as Linked Data. Master's thesis, U. of Maryalnd, Baltimore County (2010)
4. Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., Ezzat, A.: A survey of current approaches for mapping of relational databases to rdf. Technical report, W3C (2009)
5. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: from Spreadsheets to RDF. In: Seventh International Semantic Web Conference, Springer (2008)
6. Syed, Z., Finin, T., Mulwad, V., Joshi, A.: Exploiting a Web of Semantic Data for Interpreting Tables. In: Proceedings of the Second Web Science Conference. (2010)
7. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: Using linked data to interpret tables. In: Proc. First Int. Workshop on Consuming Linked Data. (2010)

LDSpider: An open-source crawling framework for the Web of Linked Data

Robert Isele[3], Jürgen Umbrich[2], Christian Bizer[3], and Andreas Harth[1]

[1] AIFB, Karlsruhe Institute of Technology
harth@kit.edu

[2] Digital Enterprise Research Institute, National University of Ireland, Galway
juergen.umbrich@deri.org

[3] Freie Universität Berlin, Web-based Systems Group
robertisele@googlemail.com, chris@bizer.de

Abstract. The Web of Linked Data is growing and currently consists of several hundred interconnected data sources altogether serving over 25 billion RDF triples to the Web. What has hampered the exploitation of this global dataspace up till now is the lack of an open-source Linked Data crawler which can be employed by Linked Data applications to localize (parts of) the dataspace for further processing. With LDSpider, we are closing this gap in the landscape of publicly available Linked Data tools. LDSpider traverses the Web of Linked Data by following RDF links between data items, it supports different crawling strategies and allows crawled data to be stored either in files or in an RDF store.

Keywords: Linked Data, Crawler, Spider, Linked Data tools

1 Introduction

As of September 2010, the Web of Linked Data contains more than 200 interconnected data sources totaling in over 25 billion RDF triples[4]. Applications that need to localize data from the Web of Linked Data [1] for further processing currently either need to implement their own crawling code or rely on pre-crawled data provided by Linked Data search engines or in the form of data dumps, for example the Billion Triples Challenge dataset[5]. With LDSpider, we are closing this gap in the Linked Data tool landscape. LDSpider is an extensible Linked Data crawling framework, enabling client applications to traverse and to consume the Web of Linked Data.

The main features of LDSpider are:

- LDSpider can process a variety of Web data formats including RDF/XML, Turtle, Notation 3, RDFa and many microformats by providing a plugin architecture to support Any23[6].

[4] http://lod-cloud.net
[5] http://challenge.semanticweb.org/
[6] http://any23.org/

- Crawled data can be stored together with provenance meta-information either in a file or via SPARQL/Update in an RDF store.
- LDSpider offers different crawling strategies, such as breadth-first traversal and load-balancing, for following RDF links between data items.
- Besides of being usable as a command line application, LDSpider also offers a simple API which allows applications to configure and control the details of the crawling process.
- The framework is delivered as a small and compact jar with a minimum of external dependencies.
- The crawler is high-performing by employing a multi-threaded architecture.

LDSpider can be downloaded from Google Code[7] under the terms of the GNU General Public License v3. In the following, we will give an overview of the LDSpider crawling framework and report about several use cases in which we employed the framework.

2 Using LDSpider

LDSpider has been developed to provide a flexible Linked Data crawling framework, which can be customized and extended by client applications. The framework is implemented in Java and can be used through a command line application as well as a flexible API.

2.1 Using the command line application

The crawling process starts with a set of seed URIs. The order how LDSpider traverses the graph starting from these seed URIs is specified by the crawling strategy. LDSpider provides two different round-based crawling strategies:

The breadth-first strategy takes three parameters: `<depth> <uri-limit> <pld-limit>`. In each round, LDSpider fetches all URIs extracted from the content of the URIs of the previous round, before advancing to the next round. The depth of the breadth-first traversal, the maximum number of URIs crawled per round and per pay-level[8] domain as well as the maximum number of crawled pay-level domains can be specified. This strategy can be used in situations where only a limited graph around the seed URIs should be retrieved.

The load-balancing strategy takes a single parameter: `<max-uris>`. This strategy tries to fetch the specified number of URIs as quickly as possible while adhering to a minimum and maximum delay between two successive requests to the same pay-level domain. The load-balancing strategy is useful in situations where the fetched documents should be distributed between domains without overloading a specific server.

[7] `http://code.google.com/p/ldspider/`

[8] "A pay-level domain (PLD) is any domain that requires payment at a TLD or cc-TLD registrar."[3]

LDSpider will fetch URIs in parallel employing multiple threads. The strategy can be requested to stay on the domains of the seed URIs.

Crawled data can be written to different sinks: File output writes the crawled statements to files using the N-Quads format. Triple store output writes the crawled statements to endpoints that support SPARQL/Update.

2.2 Using the API

LDSpider offers a flexible API to be used in client applications. Each component in the fetching pipeline can be configured by either using one of the implementations already included in LDSpider or by providing a custom implementation. The fetching pipeline consists of the following components:

The Fetch Filter determines whether a particular page should be fetched by the crawler. Typically, this is used to restrict the MIME types of the pages which are crawled (e.g. to RDF/XML).

The Content Handler receives the document and tries to extract RDF data from it. LDSpider includes a content handler for documents formatted in RDF/XML and a general content handler, which forwards the documents to an Any23 server to handle other types of documents including Turtle, Notation 3, RDFa and many microformats.

The Sink receives the extracted statements from the content handler and processes them usually by writing them to some output. LDSpider includes sinks for writing various formats including N-Quads and RDF/XML as well as to write directly to a triple store using SPARQL/Update. Both sinks can be configured to write metadata containing the provenance of the extracted statements. When writing to a triple store, the sink can be configured to include the provenance using a Named Graph layout.

The Link Filter receives the parsed statements from the content handler and extracts all links which should be fetched in the next round. A common use of a link filter is to restrict crawling to a specific domain. Each Link Filter can be configured to follow only ABox and/or TBox links. This can be used for example to configure the crawler to get the schema together with the primary data.

2.3 Implementation

LDSpider is implemented in Java and uses 3 external libraries: The parsing of RDF/XML, N-Triples and N-Quads is provided by the NxParser library[9]. The HTTP functionality is provided by the Apache HttpClient Library[10], while the Robot Exclusion Standard is repected through the use of the Norbert[11] library.

[9] `http://sw.deri.org/2006/08/nxparser/`
[10] `http://hc.apache.org/`
[11] `http://www.osjava.org/norbert/`

3 Usage examples

We have employed LDSpider for the following crawling tasks:

- We employed LDSpider to crawl interlinked FOAF profiles and write them to a triple store. For that purpose, we crawled the graph around a single seed profile (`http://www.wiwiss.fu-berlin.de/suhl/bizer/foaf.rdf`) and compared the number of traversed FOAF profiles for different number of rounds:

rounds	1	2	3	4	5
profiles	1	10	101	507	6730

- We employed LDSpider to crawl Twitter profiles, which expose structured data using RDFa. We started with a single seed profile (`http://twitter.com/aharth`) and wrote all traversed profiles to a triple store and compared the number of profiles for different number of rounds:

rounds	1	2	3
profiles	1	38	1160

  As the number of profiles grows faster than in the previsous use case, we can conclude that the interlinked Twitter profiles build a much denser graph than the FOAF web.
- LDSpider is used in an online service which executes live SPARQL queries over the LOD Web[12]
- We used LDSpider to gather datasets for various research projects; e.g. the study of link dynamics [4] or the evaluation of SPARQL queries with data summaries over Web data [2]

In summary, LDSpider can be used to collect small to medium-sized Linked Data corpora up to hundreds of millions of triples.

References

1. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
2. Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, Kai-Uwe Sattler, and Jürgen Umbrich. Data summaries for on-demand queries over linked data. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 411–420, New York, NY, USA, 2010. ACM.
3. Hsin-Tsang Lee, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Irlbot: scaling to 6 billion pages and beyond. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 427–436, New York, NY, USA, 2008. ACM.
4. Michael; Hogan Aidan; Polleres Axel; Decker Stefan Umbrich, Jürgen; Hausenblas. Towards dataset dynamics: Change frequency of linked open data sources. *3rd International Workshop on Linked Data on the Web (LDOW2010), in conjunction with 19th International World Wide Web Conference*, 2010.

[12] `http://swse.deri.org/lodq`

Semantic Web Technologies for a Smart Energy Grid: Requirements and Challenges*

Andreas Wagner, Sebastian Speiser, and Andreas Harth

Institute AIFB, Karlsruhe Institute of Technology, Germany
{a.wagner,speiser,harth}@kit.edu

Abstract. The Smart Grid aims at making the current energy grid more efficient and eco-friendly. The Smart Grid features an IT-layer, which allows communication between a multitude of stakeholders and will have to be integrated with other "smart" systems (e.g., smart factories or smart cities) to operate effectively. Thus, many participants will be involved and will exchange large volumes of data, leading to a heterogeneous system with ad-hoc data exchange in which centralised coordination and control will be very difficult to achieve. In this paper, we show parallels between requirements for the (Semantic) Web and the Smart Grid. We argue that the communication architecture for the Smart Grid can be built upon existing (Semantic) Web technologies. We point out differences between the existing Web and the Smart Grid, thereby identifying remaining challenges.

1 Introduction

The Smart Grid – a radical redesign of the traditional energy grid – aims at profoundly changing the way how energy is created, distributed and consumed, thereby saving a considerable amount of energy [1, 2]. The envisioned Smart Grid should be [1]: (V1) flexible, i.e., fulfil current requirements, but also allow future extensions, (V2) accessible, i.e., allow access to/from all participants, (V3) reliable, i.e., assure quality of supply and (V4) economic, i.e., provide best value and allow for innovation and competition. Keeping the Smart Grid vision in mind, we wish to design a communication architecture, which achieves the above goals.

In the following, we describe a preliminary communication architecture developed in context of the Smart Grid project MeRegioMobil[1]. Our contribution is two-fold: 1) We outline requirements for a communication architecture for the Smart Grid and describe how (Semantic) Web technologies meet them. 2) We outline the remaining differences between the (Semantic) Web and the Smart Grid, thereby identifying future research problems.

The remainder of the paper is structured as follows: We present architecture requirements and an initial architecture in Section 2. In Section 3, we describe the differences between the Web and the Smart Grid and outline novel problems. We conclude with Section 4.

* This work was in part supported by the German Federal Ministry of Economics and Technology (MeRegioMobil, Grant 01ME09005).

[1] http://meregiomobil.forschung.kit.edu/

2 A Semantic Web Architecture for the Smart Grid

In this section, we present requirements for a communication architecture, which we derived from the Smart Grid vision and the literature, e.g., [1, 2]. Further, we introduce a (Semantic) Smart Grid communication architecture meeting the requirements.

- *R1 - General Requirement* A suitable architecture should incorporate a layered (data access, data representation and application layers) communication stack providing different functionalities and levels of abstraction. Employing a layered architecture leads to a more flexible and versatile Smart Grid communication, as varying technologies may be integrated and functionalities can be modified or replaced (V1).
- *R2 - General Requirement* We wish an appropriate architecture to be decentralised and thus omit a single point of failure, in order to provide the desired reliability (V3).
- *R3 - Data Access Layer* In order to allow full access to/from all participants we need a naming mechanism to uniquely identify each participant (V2).
- *R4 - Data Access Layer* The Smart Grid needs flexible, open and scalable data access procedures (V1/V2/V4). Flexibility means that a communication architecture should be able to facilitate heterogeneous participants employing hardware of lower or higher specification. Further, procedures only available under restrictive licenses to a selected number of participants might hinder innovation. Thus, standards should be open and royalty-free. As huge amounts of data are handled within the Smart Grid, data access procedures should be light-weight, i.e., scale well w.r.t. the data volume.
- *R5 - Data Representation Layer* We need structured and machine interpretable data models for representation of data semantics and context, in order to allow flexible application and business logic at higher layers (V1).
- *R6 - Data Representation Layer* Data semantics may be used for data integration, thereby fostering the access of heterogeneous participants (e.g., employing different data schemas) (V1/V2).
- *R7 - Application Layer* We have to support participants in making (automated) decisions, i.e., provide the means to express application and business logic (V4).
- *R8 - Application Layer* For allowing decision making based on logic, we have to fulfil (complex) information needs, thus we need to provide (semantic) querying features (V2).
- *R9 - Application Layer* Last, via logic we have to ensure data security and privacy, i.e., safeguard the grid against attacks and enable data protection mechanisms (V3).

There have been various proposals for a communication architecture for the Smart Grid, e.g., [1, 2]. In these works, the authors aim at a top-down architecture design approach employing a wide spectrum of both open and proprietary protocols. However, we aim at a concrete communication architecture, based on open and royalty-free standards, which are already applied in similar networks such as the Web. Further, while stating in [2] that a semantic layer (providing data semantics and context) is needed, no suitable standards are identified. As a solution, we advocate the use of semantic technologies to provide machine-interpretable data, thereby enabling advanced Smart Grid applications and processes.

Considering the requirements and in particular the layered architecture, one might notice strong parallels to the (Semantic) Web Stack – an adaptation of which would result in a layered and decentralised architecture (R1/R2). More precisely, we recommend an architecture as follows:

- *Data Access Layers* We advocate URIs for identification of participants (R3). We employ a TCP/IP stack with HTTP as transfer protocol for establishing a connection and accessing data (R4). However, standard Internet protocols are usually not adequate for low-power devices, due to their overhead from the various protocol headers. Thus, special protocols developed for low-power devices (e.g., sensors) may be adapted: e.g., a light-weight layered architecture such as IEEE 802.15.4 (physical and MAC layer), 6LoWPAN (internet layer, IPv6 version for IEEE 802.15.4 networks) or a single layer coupled with a middle-ware (for communication with TCP/IP networks), e.g., [3] (R4).
- *Data Representation Layers* To support a semantic understanding we advocate RDF(S) (if necessary extended with OWL features) to provide light-weight means for machine-interpretable data encoding (R5). Via Linked Data principles, data from different sources can be linked and thus integrated (R6).
- *Application Layers* Application and business logic can be represented via RIF (R7). We may use SPARQL as means to query RDF data and thereby allow the articulation of information needs (R8). Last, employing proof and trust mechanisms (together with rules), we can model constraints for the necessary data privacy and security (R9).

3 Open Challenges in the Smart Grid

In this section, we identify future research questions in a (Semantic) Smart Grid.

Challenge 1: Support Heterogeneous Participants (Data Access Layer, R4) Devices in the Smart Grid have a higher level of (technical) heterogeneity than within the Web. That is, we have to enable a flexible and light-weight (ad-hoc) integration of low-power devices (e.g., sensors and actuators) using low-level protocols with traditional information systems working on higher levels of abstraction. In fact, problems well-known within sensor networks (e.g., data uncertainty, vastness or integration) are aggravated in the Smart Grid, as we have various distributed, heterogeneous, low-power device networks (e.g., households) and various high-level applications (e.g., billing or energy consumption prediction) which depend on reliable data in real-time.

Challenge 2: Flexible Data Schema (Data Representation Layer, R5/R6) In the Semantic Web schema learning, schema design or schema alignment are well-known problems. However, in the Smart Grid there is a large number of different stakeholders (e.g., energy producers, grid operators or appliance manufacturers) having divisive backgrounds and goals. Thus, creating and enforcing a common data schema may be challenging. Additionally, in the Smart Grid we have little a-priori information about the participants and the data exchanged. For example, customers may add new or remove old appliances within their households or new service providers may participate in the markets. Added participants may contribute new kinds of data, while existing ones still except a certain data input. Thus, we need a very flexible data schema incorporating some fixed parts

(modelling static aspects of the grid), while being easily expandable and (to some extend) adjustable.

Challenge 3: Large-scale Complex Event Processing (Application Layer, R7) The vast amount of data that comes from data sources within the grid has to be processed efficiently to enable smart behaviour. Billing and usage analysis can be done using conventional batch processing methods. However, the dynamic adaptation of the grid to the current situation (e.g., current energy consumption) requires real-time complex event processing on a very large scale. In particular, due to the data vastness and uncertainty (e.g., data from sensors), efficient and reliable event processing becomes very challenging. Note, in contrast to traditional Web scenarios, actions triggered in the Smart Grid have (possibly drastic) real-world effects (e.g., energy outages). Thus, we have a very low fault tolerance when making decisions.

Challenge 4: Privacy and Security (Application Layer, R9) Last, there is a strong need for privacy and security within the Smart Grid. Privacy concerns the data about individuals, e.g., information about premises, vehicles and appliances or energy consumption. Traditional access control mechanisms are helpful to block unwanted data access. However, there are many situations where initial data access is granted, but the subsequent data usage has to be restricted (e.g., restricted to few purposes or participants). Also, there may be regulations enforcing the publishing of specific data. Means for expressing usage restrictions and a technical enforcement such restrictions (e.g., at certain participants such as a metering provider) must be supported. Thus, e.g., work on WWW policies should be adapted to allow a privacy-aware grid. Further, the Smart Grid includes participants with very high security requirements (e.g., a clearinghouse or an energy provider). That is, a malicious access at such participants can have disastrous real-world effects. Thus, a communication architecture must provide strong means for securing high risk participants, while still allowing access to/from the remaining (open) grid.

4 Conclusion

In this paper, we have argued that open, royalty-free (Semantic) Web standards can provide the foundation for a Smart Grid communication architecture. Further, we listed the remaining challenges that stem from differences between the Smart Grid and the Web, i.e., support of (very) heterogeneous participants, a flexible schema, large-scale complex event processing and a strong need for privacy and security. In the future, we plan to extend our work by implementing the outlined architecture in our laboratory and conduct first field tests, thereby (on a step-by-step basis) addressing the outlined problems.

References

1. European Technology Platform - SmartGrids Vision and Strategy for Europes Electricity Networks of the Future. European Comission, 2006.
2. NIST Framework and Roadmap for Smart Grid Interoperability Standards. National Institute of Standards and Technology, 2010.
3. K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 1199–1202. VLDB Endowment, 2006.

A Graph-based Approach to Indexing Semantic Web Data

Xin He[1] and Mark Baker[1]

[1] School of Systems Engineering, University of Reading, Whiteknights, Reading, Berkshire, RG6 6AY, UK
{x.he, mark.baker}@reading.ac.uk

Abstract. To the best of our knowledge, existing Semantic Web (SW) search systems fail to index RDF graph structures as graphs. They either do not index graph structures and retrieve them by run-time formal queries, or index all row triples from the back-end repositories. This increases the overhead of indexing for very large RDF documents. Moreover, the graph explorations from row triples can be complicated when blank nodes, RDF collections and containers are involved. This paper provides a means to index SW data in graph structures, which potentially benefit the graph exploration and ranking in SW querying.

Keywords: Semantic Web, search, query, RDF, resource, ontology.

1 Introduction

The task of querying resources on the Semantic Web (SW) is different to information retrieval from the conventional Web. This is mainly due to the forms in which information is stored differing between the Web and the SW, and the distinct levels of semantic support. Instead of web pages and conventional databases, SW data is stored in Resource Description Framework (RDF) documents. RDF data consists of many triples, each of which contains a subject, a predicate and an object, represented using either Uniform Resource Identifier (URI) or Literal (human-readable text). Many existing SW search systems do not index graph structure, but only make mappings from literals to the corresponding resources [5, 7] or documents [2, 3, 4]. Other systems partially or wholly index the RDF graph structure in the backend semantic data repositories. This structure information is represented as individual triples, and is stored either as inverted indices in conventional IR engines [6, 8] or as database records [1]. However, these systems suffer from the following problems:

- Indexing an excessive number of triples. This is very costly when the search engine is geared towards the future SW.
- Limited support for access patterns in systems. Access patterns (S:?:?), (?:P:O), (S:?:O), and (?:?:O) are not efficiently supported in the systems that index triples using IR engines.
- Not supporting complex graph structures. Most of the SW search systems we have studied are not suitable for RDF graphs that owe to the use of blank nodes, RDF collections, and containers. Without the necessary graph structure information

indexed, exploring graphs that include blank nodes, RDF containers and collections relying on row triples can be very complicated.

By analysing the limitations in existing efforts and considering the specific way that SW data is stored, this paper presents a graph-based approach to indexing SW data.

2 Unit-Graph – Handling SW Graph Structures

The SW is not a simple hierarchical tree representing the subsumption relationships between concepts and their instances, but instead a complicated net-based Directed Labelled Graph (DLG) with mutual relations between nodes possibly existing. Established indexing techniques, such as B-trees and hash-tables, are designed for data with hierarchical structure. It is simple to index textual descriptions on the SW using such techniques, but is impractical to index SW graph structures.

Although a whole RDF graph is normally not hierarchical, we found that by dividing it into fractions, it is always possible to represent each fraction using a hierarchical structure. Therefore, an RDF graph can be indexed as a collcetion of the tree-based fractions. In this paper, such a fraction is called a *Unit-Graph*. Figure 1 illustrates three unit-graphs for resources *uorcs:M_Baker*, *uorcs:X_He*, and *ex:06Paper*, enclosed using blue, green, and red dashed lines respectively. In each unit-graph, the resource being described is called the *Root node*, while each resource describing the root node is called a *Subnode* (of the root node). From the root node to each subnode, only forward links are included. For example, the unit-graph for *uorcs:X_He* includes three literals (in conjunction with the three edges) and one subnode (in conjunction with edge *foaf:publications*). Edge *dc:creator* is not included.

Fig. 1. The unit-graphs for three resources on the Semantic Web

These resources may be described in one SW document, or separately described in multiple documents, and interlinked by semantic links. It should be noted that the only system properties indexed in unit-graphs are *rdfs:label* and *rdfs:comment*. System properties refer to those described by RDF-S and OWL. System properties include properties that are not related to the result resources for each query, such as the restrictions, and ontology versioning. There are also attributes indirectly related to the description of the result resources, e.g. *rdf:type*, *owl:sameAs*, and *rdfs:seeAlso*. These attributes are separately indexed.

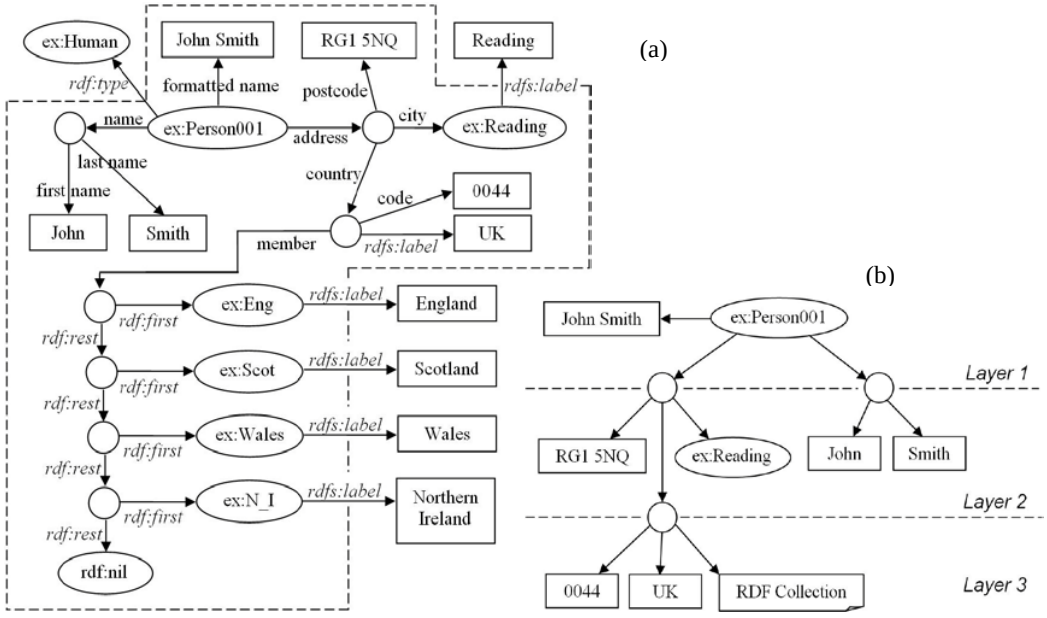


Fig. 2. The unit-graph for a resource that contains blank nodes and RDF collection

Although unit-graphs are strictly hierarchical, they are unsuitable for indexing using existing indexing techniques due to the possible inclusion of blank nodes, RDF containers and collections, e.g. the unit-graph in Figure 2 (a) (enclosed using dashed lines). User defined properties are represented using their label values for simplicity.

Using our approach, each unit-graph is modelled into layers, separated by blank nodes. These blank nodes are not those used in RDF containers and collections. The unit-graph for resource *ex:Person001* (shown in Figure 2 (a)) is modelled into three layers, as illustrated in Figure 2 (b). Property label values are omitted for simplicity.

We can intuitively see that each blank node has a maximum of three types of forward neighbours, that is, subnode, literal, and blank node. Thus, by modelling the root node or each of the blank node as an object, which contains three primitive value lists, storing subnode URIs, literal values, and blank nodes respectively, and letting each of these blank nodes have the type of the same object (as above), the data about the blank nodes in lower layers can be added to the model recursively. Thus, a unit-graph can be indexed layer by layer (from top to bottom) into an object.

In addition, multiple-values for each property (1: n relationship for the subject and object(s) of each property) are supported. Thus, an RDF container or collection can be pre-processed at indexing time and stored in a primitive value list as an item of one primitive value list for its "super" blank node object. The graph structure of RDF containers or collections is not recorded.

Furthermore, in our approach, graph structure and data values are separately indexed. Each literal is assigned an internal identifier, namely *Literal ID*, represented

using a positive integer. Two literals (from different graphs) that contain the same content will be assigned the same identifier. Each resource in a unit-graph (typically identified using URI, including the root node, its subnodes and properties) is also assigned an internal identifier, namely *Resource ID*, represented using a positive integer identifier. Thus, all primitive values are represented using integers. This severely minimises the storage size of unit-graphs, and facilitates the reuse of literals and resource URIs in unit-graphs. Moreover, using unit-graphs, graph explorations can be readily performed by matching between subnodes and rootnodes (in different unit-graphs). Due to the use of Resource IDs, the graph exploration process is actually the process of comparing numbers (see whether they are equal) rather than matching long strings (the resource URIs), which is believed to be much more time-efficient.

3 Conclusion

In this paper, we propose an approach to indexing SW data which can address the drawbacks in existing efforts in the same domain. We have presented a detailed tree-based data model to effectively hold RDF graph structures. We clarify how it deals with complex graph structures, especially when blank nodes, and RDF containers and collections are involved. We have presented how internal identifiers are employed to represent literals and resource URIs, and thereby minimises the disk capacity for indexing, and improves system performance. In addition, we explain the advantages our graph-based approach to dealing with RDF graphs has over the triple-based indexing schemes. Our graph-based approach provides ready accesses to the SW graph structures and flexible graph explorations without the need to index an excessive number of triples, and is capable of dealing with complex graph structures.

References

1. Hogan, A., Harth, A., Umbrich, J., Decker, S.: Towards a Scalable Search and Query Engine for the Web. In: Proceeding of the 16th WWW, Poster Session, pp. 1301--1302 (2007)
2. Ding, L., Finin, T., Joshi, A., Peng, Y., Cost, R., Sachs, J., Pan, R., Reddivari, P., Doshi, V.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proc. 13thCIKM (2004).
3. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: A Document-Oriented Lookup Index for Open Linked Data. In: Journal of Metadata, Semantics and Ontologies, vol. 3, no. 1, pp. 37--52. (2008)
4. d'Aquin, M., Sabou, M., Dzbor, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Motta, E.: WATSON: A Gateway for the Semantic Web. In: Proc. ESWC, Poster Session. (2007)
5. Lei, Y., Uren, V., Motta, E.: SemSearch: A Search Engine for the Semantic Web. In: Proc. EKAW, pp. 238--245, (2006)
6. Zhang, L., Liu, Q., Zhang, J., Wang, H., Pan, Y., Yu, Y.: Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data. In: Proc. ISWC2007 + ASWC2007, pp. 652--665. LNCS, vol. 4825, pp. 652--665, (2008)
7. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: Proc. WWW2008, Poster Session, pp. 1101--1102, (2008)
8. Wang, H., Zhang, K., Liu, Q., Tran, T., Yu, Y.: Q2Semantic: A Lightweight Keyword Interface to Semantic Search. In: 5th ESWC. LNCS, vol. 5021, pp. 584--598, (2008)

xhRank: Ranking Entities on the Semantic Web

Xin He[1] and Mark Baker[1]

[1] School of Systems Engineering, University of Reading, Whiteknights,
Reading, Berkshire, RG6 6AY, UK
{x.he, mark.baker}@reading.ac.uk

Abstract. In general, ranking entities (resources) on the Semantic Web (SW) is subject to importance, relevance, and query length. Few existing SW search systems cover all of these aspects. Moreover, many existing efforts simply reuse the technologies from conventional Information Retrieval (IR), which are not designed for SW data. This paper proposes a ranking mechanism, which includes all three categories of rankings and are tailored to SW data.

Keywords: Semantic Web, ranking, RDF, entity, resource, ontology.

1 Introduction

Semantic Web (SW) querying in general involves matchmaking, graph exploration, and ranking, which form a process pipeline. Existing approaches to ranking SW entities (resources) can be categorised into three types, based on importance, relevance, and query length respectively. Importance-based rankings [2, 4, 5, 6] rank the importance of SW resources, such as classes, instance resources and properties. Relevance-based rankings [2, 4, 5, 6] match keywords to SW resources. These approaches are purely based on word occurrence, and do not taken into account word order and dispersion in literal phrases. Query length-based rankings [6] rank resource by following the idea that shorter queries tend to capture stronger connections between key phrases. However, we rarely see ranking schemes used in existing SW search engines that cover all of these aspects. In addition, although Information Retrieval (IR) and web algorithms, such as *PageRank* and *TF-IDF* have been adapted for application in some SW search engines, we argue that they can still be further improved to be better suited for SW data.

Therefore, by analysing the limitations presented in existing research efforts and considering the specific way that SW data is stored, this paper proposes an approach, namely *xhRank*, to ranking SW resources. This includes relevance, importance, and query-length based rankings, all of which are particularly designed for SW data.

2 The xhRank Approach

In SW resource searching, there are in general three situations, in which a user input may match an instance resource that the user intends to find (*Target Resource*):

– (1) Only the target resource is matched. The user-input keywords uniquely match

with the literals that directly describe the target resource. In this case, the user intends to find a resource by providing its most direct annotations.

- (2) The target resource and its forward neighbouring resources are matched. The user-input keywords match not only the literals that directly describe the target resource, but the literals that describe its forward neighbours. These neighbours represent the attributes of the target resource. In this case, the user intends to find a resource by providing its most direct annotations as well as information about some attributes of the resource that is known to the user.
- (3) Only forward neighbouring resources of the target resource (but not the target resource itself) are matched. The user-input keywords match the literals describing the forward neighbours of the target resource, but not the literals describing the target resource itself. In this case, the user intends to find a resource by providing information about some attributes of the resource that is known to the user.

In xhRank, all these situations are covered in the overall ranking, which is a summation of the relevance, importance, and query-length rankings, as presented below.

2.1 Relevance-based Ranking

Phrase-level Ranking. xhRank employs an alternative phrase ranking approach to the word occurrence-based approach used by most existing SW search systems. In addition to syntactical similarity, our approach takes into account term order and dispersion. The degree of similarity of a phrase (*Key Phrase*) to another phrase (*Target Phrase*) is determined by a phrase, called *Related Key Phrase*, extracted from the key phrase, in which each word corresponds to a word in the target phrase and in which the term order is compliant with the target phrase. For example, given the key phrase “Audrey Hepburn Hollywood Actress” and the target phrase “Audrey Hepburn was a Belgian-born, Dutch-raised actress of British and Dutch ancestry”, the related key phrase is “Audrey Hepburn Actress”. It should be noted that there may be more than one such related key phrase exists for a key phrase - target phase pair.

In the context of SW query, a key phrase refers to a phrase extracted from the user input, whilst a target phrase refers to the value of a literal. Instead of returning an overall score as the result, the resulting related key phrases (*Phrase Similarity Result*) are returned, with each word in the related key phrases represented by its position in the key phrase, in conjunction with a rating value for that word. Each word in the related key phrase is rated according to the (1) Syntactical similarity *S*: the similarity score between the keyword and the corresponding word in the target phrase; (2) Importance of the keywords *I*: specified by the user; (3) Normalisation ratio *N*: used to normalise the related key phrase by the length of the literal. The higher the ratio of words in the key phrase to words in the target phrase, the more valuable these words are; and (4) Discontinuous weighting *D*: The more times the words in the related key phrase are divided by the non-related words, the less valuable these related words are.

Graph-level Ranking. The graph mentioned here is the resulting graph from a graph exploration process. The node where the graph exploration initiated is called *Central Node*, which is by design related to the user input, and the graph itself is called *Context Graph*. Graph-level ranking is to compute the relevance of the central node to

the user input, which is subject to all resources within the context graph whose literals are related to the user input. Each of such resources is called a *Related Node*.

The relevance of a graph to a user input is calculated based on how well the user-input key phrases are covered by the literals related to the user input. By assembling the phrase similarity results (each of which is obtained by the phrase-level ranking against a key phrase - related literal pair), all possible coverage against a key phrase is obtained. The relevance score is thus computed subjects to the best coverage result.

2.2 Importance-based Ranking

Resource (Node) Ranking. xhRank employs a variation on *ReConRank* [2] to rank the importance of resources. ReConRank (employed in *SWSE* [3]) is altered from the well-known *PageRank*/*HITS* algorithms. xhRank further improves on it by executing the ranking based on a complete graph (at global scale) and prior to query time.

Property (Edge) Ranking. In xhRank, the importance of SW property resources in RDF graphs (as edges) is dependent on the cost of that property. This is a prerequisite of the query length-based ranking, and is uniquely applied to the properties describing instance resources. The cost of a property *P* in the unit-graph [1] of a resource *A* is determined by the popularity of *P* among all instance resources of class *C*, where *A* is an instance of *C*. Thus, each property is ranked against a class.

2.3 Query Length-based Ranking

In xhRank, the query length-based ranking is used to evaluate a node (target node) within a graph (context graph) against a user input. The target node is evaluated based on the semantic distance between the target node and each of the nodes (related node) within the context graph that is related to the user input.

2.4 Overall Ranking

Overall ranking extends the graph-level (relevance) ranking by complementing it with importance and query-length based rankings. The input to the ranking process is a list of explored graphs generated by the graph exploration process (a process prior to ranking). Each explored graph has a related node as its root. Thus, overall ranking is performed against each of these explored graphs (as the context graph) and against a node within the graph (as the target node). In the three situations discussed above, in situation (1) and (2), the target node is just the root node of the explored graph, which is also a related node. However, in situation (3), the target node is not a related node, but the "super-node" (backward neighbour) of all related nodes within the context graph. Thus, for each explored graph, in addition to the root node, the *Top Node* is also selected as a target node. A top node of an explored graph is the node, from which all related nodes can be navigated to by means of only following forward links.

In addition, there are a few more points to note:

– Although explored graphs are strictly hierarchical, there can still be more than one top node in an explored graph. In this case, only the top node with the closest overall distance to the related nodes is selected.
– Top node strategy is applied only when there is more than one related node in the explore graph, which would otherwise fall into situation (1).
– Non-root related nodes in an explored graph are not selected as target nodes.

Therefore, in order to incorporate query-length based ranking into the graph-level (relevance-based) ranking, when performing graph-level ranking, prior to the related key phrases being assembled, the rating value for each keyword position is multiplied by the reciprocal for the cost of the path from the target node to the related node that is described by that literal. In order to introduce the importance-based ranking to the graph-level (relevance-based) ranking, the importance of each resource node and the cost of each property is applied to the graph-level ranking. Hence, the overall ranking of a target node against a user input is obtained. Consequently, the overall ranking value of all target nodes are ordered, and the best *K* results are returned to the user.

It should be noted that graph explorations are performed based on the SW data, which includes all semantic relations that have been deduced from the corresponding ontologies prior to query time. Therefore, by interpreting the three situations (by means of following the semantic links) all semantics of the SW data are discovered.

3 Conclusions

In this paper, a ranking approach, namely xhRank, is proposed, which is tailored to the nature of SW data, in particular, the three possible situations in SW resource searching. The phrase-level (relevance-based) ranking provides a means to compute the similarity between two phrases by considering term relevance, position, and dispersion, which is believed more accurate than pure word occurrence-based approaches. The introduction of the importance and query length-based rankings to the graph-level (relevance-based) ranking further improves the ranking accuracy.

References

1. He, X. et al: A Graph-based Approach to Indexing Semantic Web Data. In: Proc. 9th ISWC, Poster and Demo Session (2010)
2. Hogan, A. et al: ReConRank: A Scalable Ranking Method for Semantic Web Data with Scalable Ranking Method for Semantic Web Data with Context. In: Proc. 2nd SSWS (2006)
3. Hogan, A. et al: Towards a Scalable Search and Query Engine for the Web. In: Proc. 16th WWW, Poster Session, pp. 1301--1302 (2007)
4. Zhang, L. et al: Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data. In: Proc. 6th ISWC+ASWC2007, pp. 652--665. LNCS, vol. 4825, pp. 652--665, (2008)
5. Cheng, G. et al: Searching and Browsing Entities on the Semantic Web. In: Proc. 17th WWW, Poster Session, pp. 1101--1102, (2008)
6. Wang, H. et al: Q2Semantic: A Lightweight Keyword Interface to Semantic Search. In: Proc. 5th ESWC. LNCS, vol. 5021, pp. 584--598, (2008)

Extending SMW+ with a Linked Data Integration Framework

Christian Becker[1], Christian Bizer[2], Michael Erdmann[3], and Mark Greaves[4]

[1] MediaEvent Services GmbH & Co. KG, Berlin, Germany — chris@beckr.org
[2] Web-based Systems Group, Freie Universität Berlin, Germany — chris@bizer.de
[3] ontoprise GmbH, Karlsruhe, Germany — erdmann@ontoprise.de
[4] Vulcan Inc., Seattle WA, US — markg@vulcan.com

Abstract. In this paper, we present a project which extends a SMW+ semantic wiki with a Linked Data Integration Framework that performs Web data access, vocabulary mapping, identity resolution, and quality evaluation of Linked Data. As a result, a large collection of neurogenomics-relevant data from the Web can be flexibly transformed into a unified ontology, allowing unified querying, navigation, and visualization; as well as support for wiki-style collaboration, crowdsourcing, and commentary on chosen data sets.

Keywords: Linked Data, Data Integration, Semantic MediaWiki

1 Introduction

The Allen Brain Atlas[5] (ABA) comprises a growing collection of interactive image databases that integrate gene expression and neuroanatomic information for a variety of different organisms. Recently, the Allen Institute has started to explore new ways to accelerate scientific progress in its area, in particular by mappings between the ABA data and other standard data resources in neuroscience. As part of this project, the Semantic MediaWiki Linked Data Extension (SMW-LDE) is being built with two broad goals: to support unified querying, navigation, and visualization through a large collection of neurogenomics-relevant data sources; and to support wiki-style collaboration, crowdsourcing, and commentary on the chosen data sets.

The software base is the SMW+ semantic wiki.[6] SMW+ is a set of open-source extensions to the popular Semantic MediaWiki [6] software which render it more appropriate for enterprise-scale use. We added a new Linked Data Integration Framework to SMW+, building on a number of open-source components for Web data access, vocabulary mapping, identity resolution, and quality evaluation of Linked Data. In SMW-LDE, data from multiple Linked Data sources can be flexibly transformed into a unified ontology that allows researchers to pose queries over data spanning multiple domains and data sets. The first phase

[5] http://www.brain-map.org/
[6] http://wiki.ontoprise.de/

of the project will bring ABA, Uniprot[7], KEGG Pathway[8], PharmGKB[9] and Linking Open Drug Data [5] data sets together in order to solve the challenge of finding drugs that target elements within a disease pathway, but are not yet used to treat the disease. The genes associated with the found drugs may then be compared for commonalities through an integrated analysis of the related ABA structure expression data.

2 Integrating Linked Data from the Web

In this chapter, we discuss the architecture of the SMW-LDE deployment, which is depicted in figure 1.

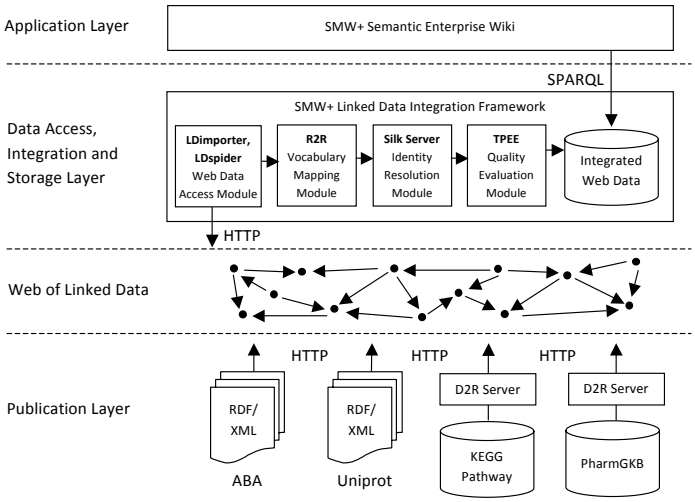


Fig. 1. Overall architecture of a Semantic MediaWiki using SMW-LDE to integrate Linked Data from the Web

All base data sets for the project, including Brain Atlas data, are published on the Web according to the Linked Data principles, thereby becoming part of a *giant global graph* – the Web of Linked Data. This logical graph is depicted in the *Web of Linked Data* layer in figure 1. Non-RDF data sources such as KEGG Pathway and PharmGKB are published using D2R Server[10]. The SMW+ Linked Data Integration Framework takes the role of a data access, integration and storage layer that makes Web data available to the application (in this case,

[7] http://www.uniprot.org/
[8] http://www.genome.jp/kegg/pathway.html
[9] http://www.pharmgkb.org/
[10] http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

the semantic wiki) in a unified manner. It consists of an integrated chain of Web data access, integration, and storage modules implemented in Java and Scala using the Jena framework. The modules and their tasks are described in the following.

1. **Web Data Access Module** The basic means to access Linked Data on the Web is to dereference HTTP URIs and to discover additional data sources by traversing RDF links, are realized using the LDspider[11] Linked Data crawler. In addition, our LDimporter module can also import data from various RDF dump formats or SPARQL endpoints using data set descriptions based on voiD descriptions [1] and Semantic Sitemaps [4].
2. **Vocabulary Mapping Module** Different Linked Data sources often use different vocabularies to represent the same type of information. For instance, both ABA and Uniprot ontologies contain the concept of a gene, but associate different relations and properties with it. In order to make Web data understandable to wiki users, the R2R Framework[12] is employed as the vocabulary mapping module in order to translate terms from different vocabularies into the wiki ontology.
3. **Identity Resolution Module** Different Linked Data sources use different URIs to identify the same entity, such as genes or proteins. The framework integrates Silk Server[13] as its identity resolution module, which interlinks newly discovered data about entities with data about them that is already defined in the wiki or other connected data sources.
4. **Quality Evaluation Module** In order to prefer data from sources known for good quality and to resolve data conflicts [2], the framework includes a data quality evaluation module, the Trust Policy Evaluation Engine (TPEE).
5. **Integrated Web Data** The repository finally stores the Web data together with provenance information to be used by the application layer. We employ the Named Graphs data model [3] for representing Web data together with provenance information as an integrated model.

3 Using Linked Data in the Wiki

Making the integrated neurogenomics-relevant data available in a semantic wiki empowers users to access it in various ways:

- An ontology browser (cf. figure 2) of SMW+ enables the exploration of the integrated data, including representation of provenance information. In this way, correlations between data from different sources, as well as data conflicts can be identified in an interactive manner.
- Inline queries (cf. figure 2) and the interactive query interface offer means for ad hoc queries against the set of integrated Linked Data sources. The query results can be organized in wiki pages and complemented with textual descriptions or interpretations.

[11] http://code.google.com/p/ldspider/
[12] http://www4.wiwiss.fu-berlin.de/bizer/r2r/
[13] http://www4.wiwiss.fu-berlin.de/bizer/silk/server/

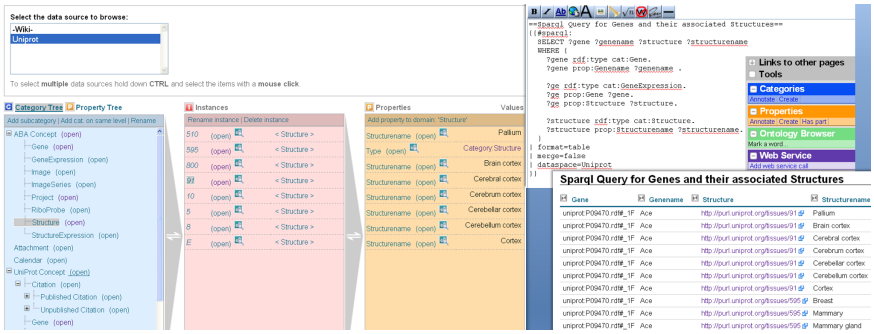


Fig. 2. SMW+ Ontology Browser showing mapped Uniprot data (left), a related query within a wiki article (top right) and rendered query result in that article (bottom right)

The most prominent benefit of using a semantic wiki environment in this project is the collaborative creation of semantic meta-data via annotations. Rather than just *querying* integrated Web data, users of the wiki can make use of the data by referring to imported genes or proteins in their own articles. By doing so, further statements about Web data resources are created that are in turn exported as Linked Data. This includes cross data source connections – for instance, the Allen Institute can publish their own data in the wiki and cross-reference it to Uniprot proteins.

Acknowledgement

This work was supported in part by Vulcan Inc. as part of its Project Halo (www.projecthalo.com).

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J. Describing linked datasets. In *Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009)*, 2009.
2. Bleiholder, J., Naumann, F. Data fusion. *ACM Computing Surveys*, 41(1):1–41, 2008.
3. Carroll, J., Bizer, C., Hayes, P., Stickler, P. Named graphs. *Journal of Web Semantics*, 3(4):247–267, 2005.
4. Cyganiak, R., Delbru, R., Stenzhorn, H., Tummarello, G., Decker, S. Semantic sitemaps: Efficient and flexible access to datasets on the semantic web. In *Proceedings of the 5th European Semantic Web Conference (ESWC2008)*, 2008.
5. Jentzsch, A., Hassanzadeh, O., Bizer, C., Andersson, B., Stephens, S. Enabling tailored therapeutics with linked data. In *Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009)*, 2009.
6. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R. Semantic wikipedia. *Journal of Web Semantics*, 5:251–261, September 2007.

Learning Co-reference Relations for FOAF Instances*

Jennifer Sleeman and Tim Finin
University of Maryland, Baltimore County
Baltimore. MD 21250 USA

Abstract. FOAF is widely used on the Web to describe people, groups and organizations and their properties. Since FOAF does not require unique IDs, it is often unclear when two FOAF instances are co-referent, i.e., denote the same entity in the world. We describe a prototype system that identifies sets of co-referent FOAF instances using logical constraints (e.g., IFPs), strong heuristics (e.g., FOAF agents described in the same file are not co-referent), and a Support Vector Machine (SVM) generated classifier.

Keywords: FOAF, machine learning, linked data

1 Introduction

The FOAF (Friend of a Friend) vocabulary has been one of the most widely used ontologies since the beginning of the Semantic Web. It defines classes and properties for describing entities (people, organizations and groups), their attributes, and their relations. What FOAF does not require is a property that represents a globally unique identifier (GUID) that can be used to recognize when two foaf:Agent individuals are co-referent, i.e., refer to the same individual whether real or fictional.

A traditional approach for identifying co-reference entities on the Semantic Web is the process of smushing [4] FOAF instances combining the information from various sources that are determined to represent the same person. One can choose to rely solely on the presence of owl:sameAs, however, its presence is not always found and it can also be represented inaccurately. There are multiple techniques used to both identify co-referent FOAF profiles and that perform some type of smushing as mentioned in our previous work [5]. We describe a hybrid approach for deciding when RDF descriptions of two FOAF agents are likely to be co-referent that combines rules and an SVM-based classifier.

While the `owl:sameAs` relation is typically used to assert that two FOAF instances refer to the same individual, this can lead to unwarranted and problematic inferences [2, 3]. For this reason, we use the weaker predicates, `coref` and `notCoref` to represent that two instances are or are not thought to be coreferential. For coreferential instances, we can merge their descriptions for some, but not all, uses. Figure 1 gives some axioms in N3 for the `coref` and `notCoref` properties. The `coref` property is transitive and symmetric and has

* Partial support for this research was provided by NSF award 0910838 and the Johns Hopkins University Human Language Technology Center of Excellence.

```
:coref a owl:TransitiveProperty, a owl:SymmetricProperty.
owl:sameAs rdfs:subPropertyOf :coref.
:notCoref a owl:SymmetricProperty.
owl:differentFrom rdfs:subPropertyOf :notCoref.
{?a :notCoref ?b. ?b :coref ?c.}  => {?a :notCoref ?c}
{?a foaf:knows ?b.}  => {?a :notCoref ?b}
```

Fig. 1: Definitions of the :coref and :notCoref properties uses instead of owl:sameAs.

`owl:sameAs` as a sub-property. `notCoref` is symmetric, but not transitive and has `owl:differentFrom` as a sub-property. The first rule states that if two instances, a and b, are not coreferent, then every instance coreferent with a is `:notCoref` with every instance coreferent with b. The second, which is really a heuristic, states that if a knows b, then they are assumed to be distinct individuals and thus `:notCoref`. Note that `owl:sameAs` implies `coref` and `owl:differentFrom` implies `notCoref`, so reasoners that can derive `sameAs` and `differentFrom` properties will also contribute to computing coreference relations.

2 Approach

Given a collection of FOAF instances to compare, we would like to cluster them into sets that we believe refer to the same person in the world. This process is divided into five stages: (i) generating candidate pairs, (ii) generating a rules-based model, (iii) classification, (iv) designating pairs as co-referent or not, and (v) creating clusters. Figure 2 shows a high level architecture of our system. We describe our approach below as defined in [6].

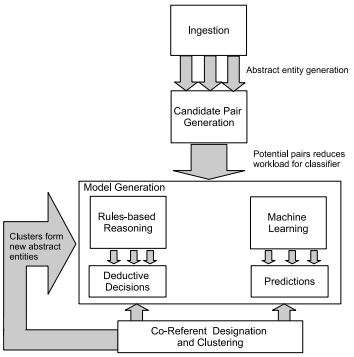


Fig. 2: Our system architecture starts by selecting pairs of foaf instances to compare and applies both rule-based reasoning and an SVM-based classifier to determine likely co-reference relations that are then use to form clusters.

Pair Generation. With a potentially large collection of FOAF instances we could proceed by testing each of the $O(N^2)$ possible pairs to see which are co-referent. Since the vast majority of the pairs will not be matched and the co-reference test will be relatively expensive, we start by filtering the possible pairs to produce a smaller set of candidates using a simple string matching heuristic test for each pair.

Rule-based Model and Classification. Filtered pairs are evaluated by rules that provide a result that indicates whether the pair is or is not co-referent. Rules can include properties such as owl:sameAs, property attributes such as inverse functional properties and rules deduced from the data itself. For example, persons defined by a persons knows graph are likely not to be co-referent

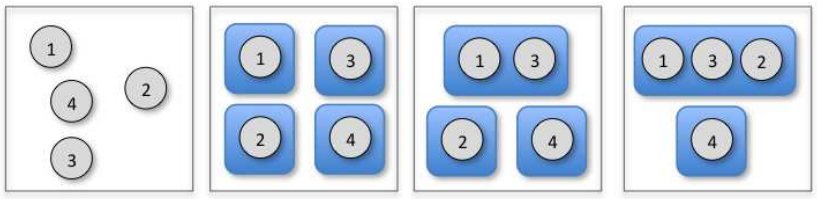


Fig. 3: After assigning each foaf individual to a singleton coreference set, we use a greedy algorithm to merge the sets that are judged to be similar.

with that person. The co-referent classifier takes a pair of FOAF instances and decides if they are co-referent or not. Property-specific features used by the classifier include distance measures of properties common to both instances, inverse functional properties, more complex distance measures, which might include unpacking semantic information (e.g., the geographical distance between to geotags) and resolving entity mentions (e.g., Baltimore) to linked data nodes, and partial analysis of the graphs centered on the instances, such as the immediate (one-hop) social networks formed by foaf:knows properties. Refer to [6] for a more detailed discussion.

Co-referent Designation. The results of the classifier are used by the co-referent process when determining co-referent pairs. The rules-based model is used to determine if a pair of FOAF instances are co-referent. If the results of the rules-based model are indeterminate, then the prediction generated by the classifier is used. Pairs that are designated as co-referent form a new cluster. Clusters are then processed by the system in addition to pairs.

Clustering. New co-referent pairs can be considered as a graph where the nodes are the original set of FOAF instances and an edge exists between two nodes that are a candidate pair as determined by the co-referent processing. Co-referent triples become part of larger cluster groups and are used for future pair matching.

Figure 3 depicts the greedy process for four foaf individuals. We begin by putting each in a singleton coreference set. A merging process continues as long as two candidate sets are judged to be similar enough to be merged into a new one that replaces its ancestors and stops when there are no pairs that can be merged. In this figure, the four foaf individuals end up in two coreference sets.

3 Evaluation

We ran two experiments, the first experiment resulted in about 50,000 triples with over 500 entity mentions. We applied the rules which resulted in 900 pairs that were designated as a non-match and the majority was undetermined. The classification portion of this process consisted of 600 pairs used for training and 3 tests consisting of 200 pairs each. The third test also included a single post-clustering run.

Our latest experiment contained about 250,000 triples with over 3500 entity mentions. We applied the rules which resulted in positive co-referent cases based

	True Positive Rate	False Positive Rate	Precision	Recall	F-Measure
E1	0.933	0.267	0.930	0.933	0.930
E2	0.959	0.128	0.958	0.959	0.958

Table 1: The results of a 10-fold cross-validation classification test show good results fir both precision and recall.

on the inverse functional property. The majority of the rules resulted in an undetermined state. As expected, the `foaf:knows` rule returned a number of pairs that resulted in a non-co-referent state. The classification training set consisted of over 1800 classes. We conducted a 10-fold cross-validation with results conveyed in table 1. Table 1 shows that our classification step is likely predicting accurately co-referent and non-co-referent pairs.

During our E2 clustering phase, the first phase of clustering resulted in a 90% accuracy. The error occurred in pairs that should have been clustered but were not. A second round of clustering did not yield any new relationship pairs among instances but cluster to cluster pairing did occur. Additional tests and evaluations are outlined in [6].

4 Conclusions and Future Work

We have described an approach to predicting coreferent pairs of FOAF instances that uses a small set of rules, a classifier developed by supervised machine learning process and clustering co-referent pairs. We have been working with FOAF data as an instance of a larger problem: automatically linking RDF instances based on their descriptions.

References

1. Artiles, J., Gonzalo, J., Sekine, S.: Weps 2 evaluation campaign: Overview of the web people search clustering task. In: 18th WWW Conf. Madrid (2009)
2. Ding, L., Shinavier, J., Finin, T., McGuinness, D.L.: OWL:sameAs and linked data: an empirical study. In: Proc. 2nd Web Science Conf. (April 2010)
3. Halpin, H., Hayes, P., McCusker, J., McGuinness, D., Thompson, H.: When owl:sameas isn't the same: An analysis of identity in linked data. In: Proc. 9th Int. Semantic Web Conf. (2010)
4. Foaf-project.org definition of smushing. http://wiki.foaf-project.org/w/Smushing (2010), accessed January 2010
5. Sleeman, J., Finin, T.: A Machine Learning Approach to Linking FOAF Instances. In: Spring Symposium on Linked Data Meets AI. AAAI (January 2010)
6. Sleeman, J., Finin, T.: Computing FOAF Co-reference Relations with Rules and Machine Learning. In: Proceedings of the Third International Workshop on Social Data on the Web (November 2010)

Silk – Generating RDF Links while publishing or consuming Linked Data

Anja Jentzsch, Robert Isele, Christian Bizer

Freie Universität Berlin, Web-based Systems Group
mail@anjajentzsch.de, robertisele@googlemail.com, chris@bizer.de

Abstract. The central idea of the Web of Data is to interlink data items using RDF links. However, in practice most data sources are not sufficiently interlinked with related data sources. The Silk Link Discovery Framework addresses this problem by providing tools to generate links between data items based on user-provided link specifications. It can be used by data publishers to generate links between data sets as well as by Linked Data consumers to augment Web data with additional RDF links. In this poster we present the Silk Link Discovery Framework and report on two usage examples in which we employed Silk to generate links between two data sets about movies as well as to find duplicate persons in a stream of data items that is crawled from the Web.

Keywords: Linked Data, Link Discovery, Identity Resolution

1 Introduction

The Web of Data is built on the fundamental idea that data items are published using dereferencable URIs wherein related data items are connected using RDF links [1]. While the Web of Data is constantly growing[1], it still forms a weakly interlinked graph and contains only a small fraction of the RDF links that could in theory be set [2]. In order to tackle this problem, we provide the Silk Link Discovery Framework. Silk generates RDF links between data items based on user-provided link specifications. Silk specifications are expressed using a declarative language and define the conditions that data items must fulfill in order to be interlinked.

The Silk Link Discovery Framework addresses the following two use cases: It can be used by data providers to generate RDF links pointing at existing datasets on the Web. These RDF links can then be published together with the primary data sets on the Web. Furthermore, Silk can be used as an identity resolution component within applications that consume Linked Data from the Web in order to augment Web data with additional RDF links which have not been set by the data providers.

Silk is provided in three different variants which address different use cases:

[1] http://lod-cloud.net/

- *Silk - Single Machine* is used to generate RDF links on a single machine. The datasets that should be interlinked can either reside on the same machine or on remote machines which are accessed via the SPARQL protocol. *Silk - Single Machine* provides multithreading and caching. In addition, the performance can be further enhanced using an optional blocking feature.
- *Silk - Map Reduce* is used to generate RDF links between data sets using a cluster of multiple machines. *Silk - Map Reduce* is based on Hadoop and can for instance be run on Amazon Elastic MapReduce. *Silk - Map Reduce* enables Silk to scale out to very big datasets by distributing the link generation to multiple machines.
- *Silk - Server* can be used as an identity resolution component within applications that consume Linked Data from the Web. *Silk - Server* provides an HTTP API for matching instances from an incoming stream of RDF data against a local set of known instances. It can be used for instance together with a Linked Data crawler to populate a local duplicate-free cache with data from the Web.

The Silk Link Discovery Framework is implemented in Scala[2] and can be downloaded from the project homepage[3] under the terms of the Apache Software License.

In the following, we will give an overview of the Silk Link Discovery Framework, report on two usage examples in which we employed the framework and present planned future work.

2 The Silk Link Discovery Framework

The Silk Link Discovery Framework consists of a console application used to interlink two data sets as well as of the Silk Server, an HTTP server, which receives an incoming RDF stream and creates links between data items. Both applications provide a flexible configuration language, the Silk Link Specification Language (Silk-LSL), to specify the conditions data items must fulfill in order to be interlinked [3]. For this purpose, the user may apply similarity metrics, such as string, date or URI comparison methods, to multiple property values of an entity or related entities. The resulting similarity scores can be combined and weighted using various similarity aggregation functions. A Silk link configuration may contain several link specifications if links for different types of data items should be generated.

The central part of the Silk Link Discovery Framework is the Silk Linking Engine, which generates the links between data items according to user-provided link specifications. The Silk Linking Engine processes the incoming data items, which are usually originating from a SPARQL endpoint, in subsequent phases:

The optional **Blocking** phase partitions the incoming data items into clusters. Since comparing every source resource to every single target resource results

[2] `http://scala-lang.org`
[3] `http://www4.wiwiss.fu-berlin.de/bizer/silk/`

in a number of $n * m$ comparisons which might be time-consuming, blocking can be used to reduce the number of comparisons. Blocking partitions similar data items into clusters limiting the comparisons to items in the same cluster. For example, given a set of books to be compared, in order to reduce the number of comparisons, one could block the books by publisher.

The **Link Generation** phase reads the incoming data items and computes a similarity value for each pair. The incoming data items, which might be allocated to a cluster by the preceding blocking phase, are written to an internal cache. From the cache, pairs of data items are generated. If blocking is disabled, this will generate the complete cartesian product of the two data sets. If blocking is enabled, only data items from the same cluster are compared. For each pair of data items, the link condition is evaluated, which computes a similarity value between 0 and 1. Each pair generates a preliminary link with a confidence value according to the similarity of the source and target data item.

The **Filtering** phase filters the incoming links in two stages: In the first stage, all links with a lower confidence than the user-defined threshold are removed. In the second stage, all links which originate from the same data item are grouped together. The number of links per source item which are forwarded to the output, is specified by an optional link limit. If a link limit is defined, only the links with the highest confidence are forwarded.

3 Silk Server

Silk Server is an identity resolution component, that can be used within Linked Data application architectures to add missing RDF links to data that is consumed from the Web of Linked Data. It is designed to be used with an incoming stream of RDF instances, produced for example by a Linked Data crawler such as LDspider. Silk Server matches data describing incoming instances against a local set of known instances and discovers missing links between them. Based on this assessment, an application can store data about newly discovered instances in its repository or fuse data that is already known about an entity with additional data about the entity from the Web.

4 Usage Examples

Silk has been employed in several scenarios to generate links between data sets on the Web of Data. In the following we report the results of employing *Silk - Single Machine* and *Silk - Server* within two usage scenarios.

4.1 Interlinking DBpedia movies with LinkedMDB directors

We employed *Silk - Single Machine* to interlink movies in DBpedia with the corresponding director in LinkedMDB. For this purpose, Silk was fed with the 50000 movies from DBpedia and 2500 directors from LinkedMDB. For each

movie, Silk was configured to set a `dbpedia:director` link from the movie to its director.[4] A single PC with a Core 2 Duo CPU and 4GB of RAM needed 55 minutes to match the complete cartesian product resulting in 125 000 000 comparisions. Silk successfully identified 5900 links between a movie and its director. In order to increase the performance, the Linking Specification was extended to employ blocking on the director names to reduce the number of comparisions. As blocking may decrease the recall of the matching, we compared the generate links. With blocking enabled Silk was still able to identify 5857 links, resulting in a loss of less than one percent, while reducing the runtime of the matching considerable to only 7 minutes.

4.2 Identifying duplicate person descriptions in a data stream

In the Web of Data we can usually find different URIs which effectively identify the same entity, e.g. `<http://tomheath.com/id/me>` and `<http://www.eswc2006.org/people/#tom-heath>` describe the same person. We employed a Linked Data crawler to crawl the FOAF web and stream the traversed profiles to the Silk Server in order to identify duplicate persons and generate `owl:sameAs` links between them. For evaluation, we used the Semantic Web Dog Food data set[5] which already interlinks some of the contained persons with their corresponding FOAF profile. Among the 56 persons for which the Semantic Web Dog Food data set provides links to their FOAF profile, Silk Server was able to reconstruct 51 links from the stream. In addition, it was able to identify the FOAF profile of another 132 persons for which Semantic Web Dog Food did not provide a link to their profile yet.

5 Acknowledgments

This work was supported in part by Vulcan Inc. as part of its Project Halo (www.projecthalo.com) and by the EU FP7 project LOD2 - Creating Knowledge out of Interlinked Data (Grant No. 257943, http://lod2.eu/).

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
2. M. A. Rodriguez. A graph analysis of the linked data cloud. *CoRR*, abs/0903.0194, 2009.
3. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *International Semantic Web Conference*, pages 650–665, 2009.

[4] The link specification can be found on `http://www4.wiwiss.fu-berlin.de/bizer/silk/linkspecs/dbpedia_linkedmdb_directors.xml`

[5] `http://data.semanticweb.org/`

Hybrid Graph based Keyword Query Interpretation on RDF

Kaifeng Xu[1], Junquan Chen[1], Haofen Wang[1], and Yong Yu[1]

Apex Data and Knowledge Management Lab
Shanghai Jiao Tong University, Shanghai, 200240, China
{kaifengxu,jqchen,whfcarter,yyu}@apex.sjtu.edu.cn

Abstract. Adopting keyword query interface to semantic search on RDF data can help users keep away from learning the SPARQL query syntax and understanding the complex and fast evolving data schema. The existing approaches are divided into two categories: instance-based approaches and schema-based approaches. The instance-based approaches relying on the original RDF graph can generate precise answers but take a long processing time. In contrast, the schema-based approaches relying on the reduced summary graph require much less processing time but cannot always generate correct answers. In this paper, we propose a novel approach based on a hybrid graph which can achieve significant improvements on processing time with a limited accuracy drop compared with instance-based approaches, and meanwhile, can achieve promising accuracy gains at an affordable time cost compared with schema-based approaches.

1 Introduction

On the way to Semantic Web, Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web. The ever growing semantic data in RDF format provides fertile soil for semantic search, and formal query languages (e.g. SPARQL) are adopted by most current semantic search systems[1, 2] to accurately express complex information needs. However, the disadvantages of formal queries are: (1) *Complex Syntax*: It is hard to learn and remember complex syntax of formal queries for ordinary users. (2) *Priori Knowledge*: Users have to know the schema of the underlying semantic data beforehand. In contrast, keyword queries cater to user habits since keywords (or known as keyword phrases) are easier to be understood and convenient to use. An approach that can leverage the advantages of both query types is to provide a keyword user interface and then translate keyword queries into formal queries.

In XML and database communities, bridging the gap between keyword queries and formal queries has been widely studied. However, there exists a limited amount of work on how to answer keyword queries on semantic data in RDF format. As an early attempt to build a semantic search system, SemSearch [3] employed a template-based approach to capture the restricted interpretations

of given keywords. Later, improved approaches [4, 5] have been proposed to address the problem of finding all possible interpretations. In particular, Thanh et al. [5] employed the RDF graph (*instance-based approaches*) to discover the connections between nodes matching the input keywords, through which the interpretation accuracy can be ensured, but at the cost of a longer processing time. This problem has been recently tackled by [6, 7], where keyword queries are translated using a summary graph extracted from the RDF data (*schema-based approaches*). Although schema-based approaches significantly speed up the processing, the schema-graph loses too much connectivity information of the corresponding RDF graph to guarantee the interpretation accuracy.

In this paper, we propose a novel effective and efficient keyword query interpretation approach based on a hybrid graph carefully constructed from the original RDF graph. A hybrid graph is much smaller than the original RDF graph, and meanwhile it can preserve as much connectivity information as possible. In this way, we construct the hybrid graph under the guidance of a graph score which reflects the best tradeoff between effectiveness and efficiency of keyword query interpretation.

2 The Hybrid Graph based Approach

Figure 1 describes the entire process (both offline and online stages) of keyword query interpretation. In the offline stage, a hybrid graph is constructed from the origin RDF graph. After that, a keyword index is built for the mapping of keywords to corresponding nodes in the hybrid graph. During the online process, keywords input by end users are first mapped to the nodes in the hybrid graph using the keyword index, and then we search on the hybrid graph to construct top-k potential tree-shaped conjunctive queries (i.e., formal queries). While our focus is the construction of the hybrid graph, we implement a similar keyword mapping, query construction and ranking as mentioned in Q2Semantic [6]. Due to space limitations, we refer you to [6] for details.

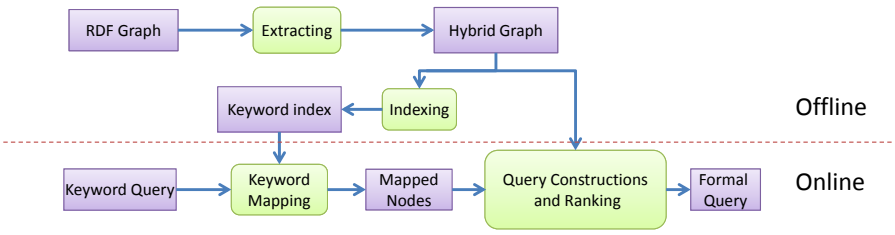


Fig. 1. The flow chart of keyword query interpretation

The construction of hybrid graph is an iterative process which extracts and refines a qualified subgraph from the original RDF graph by means of a graph

score. The graph score is used to define the overall interpretation performance, which plays an important role from the starting point to the ending point of each interaction in the whole construction process. More precisely, the graph score is the linear combination of the size of the hybrid graph and the amount of the connectivity information contained in the graph. The workflow of a hybrid graph construction is illustrated in Figure 2(a) which takes the original RDF graph as input. A *construction unit* (CU) is employed to carry out the refinement on the given RDF graph to generate a qualified hybrid subgraph. The CUs are additionally used several times for further refinement on the remaining unqualified RDF subgraphs. Finally, several qualified hybrid subgraphs are returned, and combined together to form an overall hybrid graph.

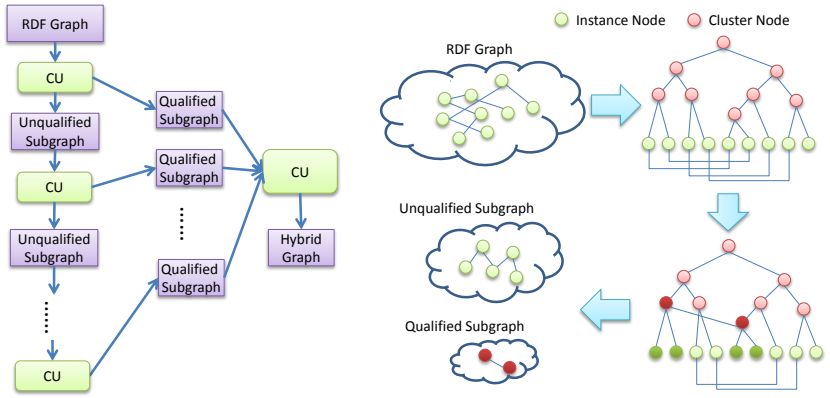


Fig. 2. (a) The work flow of hybrid graph construction. (b)The work flow of CU.

Figure 2(b) shows the detailed components inside a CU. Given a RDF graph G, CU will work through the following streamline: (1) *Instance Clustering*: This phase aims to generate the nodes of hybrid graph. Each instance in G is represented by the feature set of its relations and regarded as a trivial cluster initially. After that, the cluster pairs with the highest similarity are hierarchically clustered until a cluster tree T is derived. (2) *Relation Refinement*: This phase aims to generate the relations of hybrid graph. It tries to substitute the relations in the given RDF graph with those between the high level nodes in the cluster tree. The relation replacement will increase the score of the graph, and the goal is to get a hybrid graph with the highest score. (3) *Graph Detachment* The score of the whole graph generated after relation refinement might be too low. Thus, we extract a part of the graph (called a qualified subgraph) whose score is above the given score threshold, and feed the remaining part (called an unqualified subgraph) into the CU for next iterations.

3 Preliminary Results

We compare our approach with instance-based approches and schema-based counterparts on three different datasets (i.e., semanticweb.org, DBpedia, DBLP) in terms of processing time and interpretation accuracy. Table 1 lists The statistics of the three data sets. We manually construct 42 scenarios (17 from semanticweb.org, 10 from DBpedia, and 5 from DBLP) for the comparison.

Table 1. Statistics of semanticweb.org, DBLP and DBpedia.

Data set	♯Category	♯Instance	♯Relation	♯Inst.degree	♯Rel.kind	♯Rel/kind
semanticweb.org	5.06×10^2	7.483×10^3	1.628×10^4	2.18	4.77×10^2	3.413×10^1
DBLP	1.0×10^1	1.640×10^6	3.176×10^6	1.94	1.0×10^1	3.176×10^5
DBpedia	2.694×10^5	2.520×10^6	6.868×10^6	2.73	1.128×10^4	6.088×10^2

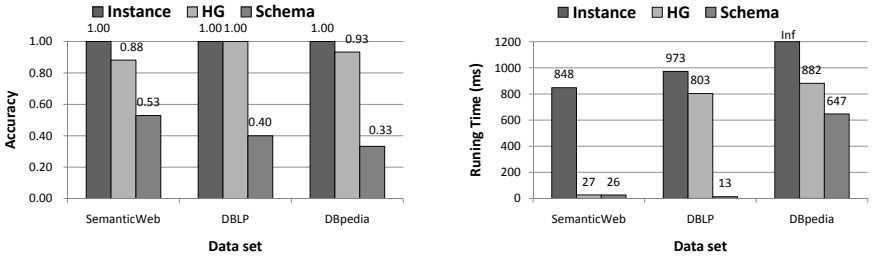


Fig. 3. The practical interpretation accuracy and efficiency on different data sets

Figure 3 shows that our approach has achieved a 61.66% efficiency improvement with a 6.17% accuracy drop over instance-based approaches on average. On the other hand, our approach achieves a 132.30% accuracy gain with a 20.08% time increase compared with schema-based approaches.

References

1. Broekstra, J., Kampman, A., Van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: ISWC. (2002) 54–68
2. Lu, J., Ma, L., Zhang, L., Brunner, J., Wang, C., Pan, Y., Yu, Y.: SOR: a practical system for ontology storage, reasoning and search. In: VLDB. (2007) 1402–1405
3. Lei, Y., Uren, V., Motta, E.: Semsearch: A search engine for the semantic web. In: EKAW. (2006) 238
4. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: Spark: Adapting keyword query to semantic search. In: ISWC/ASWC. (2007) 694
5. Tran, T., Cimiano, P., Rudolph, S., Studer, R.: Ontology-based interpretation of keywords for semantic search. In: ISWC/ASWC. (2007) 523
6. Wang, H., Zhang, K., Liu, Q., Tran, T., Yu, Y.: Q2Semantic: A lightweight keyword interface to semantic search. In: ESWC. (2008) 584
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: ICDE. (2009) 405–416

A Semantic Web Repository for Managing and Querying Aligned Knowledge

James P. McGlothlin, Latifur Khan

The University of Texas at Dallas
Richardson, TX USA
{jpmcglothlin, lkhan}@ utdallas.edu

Abstract. Ontology alignment is the task of matching concepts and terminology from multiple ontologies. Ontology alignment is especially relevant in the semantic web domain as RDF documents and OWL ontologies are quite heterogenous, yet often describe related concepts. The end goal for ontology matching is to allow the knowledge sets to interoperate. To achieve this goal, it is necessary for queries to return results that include knowledge, and inferred knowledge, from multiple datasets and terminologies, using the alignment information. Furthermore, ontology alignment is not an exact science, and concept matchings often involve uncertainty. The goal of this paper is to provide a semantic web repository that supports applying alignments to the dataset and reasoning with alignments. Our goal is to provide high performance queries that return results that include inference across alignment matchings, and rank results using certainty information. Our semantic web repository uses distributed inference and probabilistic reasoning to allow datasets to be efficiently updated with ontology alignments. We materialize the inferred, aligned data and make it available in efficient queries.

1 Introduction

We shall begin this paper by defining the high level problem we wish to solve and the system goals. Given multiple datasets (RDF documents) and ontologies, the goal is to allow queries against the complete knowledge set. The queries should be able to be specified using the terminologies from any of the ontologies, or from a new common global terminology. The queries should return all relevant knowledge, including inferred triples and triples that are specified using different, but corresponding, terminologies. The queries should rank results based on confidence values, and should enable selection based on probability conditionals. Furthermore, these queries should return such knowledge in a very timely manner.

Determining the alignments or matchings is the obvious first task. If two datasets use different terminology, then the concepts must be aligned to enable queries across both knowledge sets. Ontology alignment is already a well-researched area. Our contribution is to allow these alignments to be applied to the dataset, to transform the data accordingly, and to allow queries against the aligned results.

In [2], we first introduced RDFKB (Resource Description Framework Knowledge Base), a bit vector schema that is uniquely able to materialize inferred triples without a performance penalty. In [6], we introduced a multiple bit vector schema using thresholds to support efficient queries involving probability. RDFKB is the only semantic web repository to materialize uncertain information and to support queries involving probabilistic inference.

Our solution in this paper builds on these technologies. We propose to materialize all inferred triples including those inferred by reasoning with alignment matchings. In fact, we propose to implement all alignment matches as inference rules. We also propose to use the threshold vector schema as our solution for propagating and querying similarity measurements.

However, this paper involves more than just utilizing previous versions of RDFKB to apply alignments. We must be able to support distributed inference across multiple datasets, which changes our data management schema. We must be able to add inference rules to an existing dataset, which changes our inference rules specifications. Ontology alignment is a fluid process. Similarity measures can fluctuate through user feedback, additional machine learning, etc. Therefore, we must be able to change or even delete inference rules from the system. Also, many times the alignment goal is to transform the data into new terminology, thus replacing the original data. Finally, trust factors should be able to be associated with data origins, in order to facilitate handling conflicting information during dataset merges.

In Section 2, we will briefly overview the RDFKB schemata and our previous work. In Section 3, we will specify the new features that enable us to apply alignments. In Section 4, we will present experimental results, and, in Section 5, we make some conclusions and define some future work.

2 Background

RDFKB uses two schemata, one for data management, and one for queries. The data management schema is centered around a Triples table. This allows us to associate additional information (probability) with each triple, to optimize the addition and deletion processes, and to encapsulate the query schema from the user.

Our query schema includes two bit vector tables: POTable and PSTable. POTable contains 5 columns: the property, object, subjectbitvector, bitcount and threshold. The subject bit vector has a 1 for each subject that appears in a triple with the corresponding property and object and with probability >= threshold. PSTable contains 5 columns: property, subject, objectbitvector, bitcount and threshold. Bit vectors allow us to access entire collections of triples by reading a single tuple. Furthermore, joins and unions can be performed using efficient bit operations.

All inferred triples are materialized during addition time using forward chaining. [2] details how our bit vector schema is able to support inference materialization without a performance penalty, and describes our implementation of OWL inference rules. [6] details our probability solution using multiple bit vectors and thresholds, and demonstrates that adding probability does not reduce query performance.

3 New Features

Inference Rules. RDFKB uses inference rules to materialize inferred data. The actual instantiated inference rules are registered with RDFKB rather than implemented by RDFKB. At a high level, an inference rule defines that given a set of 1 or more triples ($A_1,\ldots,A_n$), we can conclude an additional triple B. For example, given <Professor1 type AssistantProfessor>, we can conclude <Professor1 type

Professor>. Using our high level definition above, it is obvious we can use inference rules to perform any alignment transformation.

The inference rule must define two methods *Infer()* and *Infer(triple T)*. *Infer(triple T)* returns the set of inferred triples that this inference rule can infer if T is added to the dataset. *Infer()* returns all triples that can be inferred across an entire dataset. This new method allows an inference rule to be added to an existing dataset, one of our requirements to support applying ontology alignments.

Provenance. In [6], RDFKB uses an InferenceCount value and forward chaining to enable updates and deletes. However, this will not allow us to update or delete an inference rule, which is required to support changing alignments and similarity measures. Therefore, we replace the InferenceCount field in the Triples table with Lineage, a foreign key into one of our provenance tables. The provenance tables are Users, Datasets, Events, and Dependencies. The Events table defines what inference event materialized an inferred triple, including the Inference Rule. This allows us query all triples materialized by an inference rule, so we are able to delete inference rules or update probabilities associated with inference rules.

The provenance tables also solve several of our other requirements. The Datasets table and Users table provide our trust factors. A concrete triple specifies its Dataset in the lineage column and we have replaced the Triples table with a collection of Triples tables. The Datasets table allows us to traverse over these tables, and inference rules can now query the entire collection of triples tables during the forward chaining process. This enables distributed inference, a core requirement for reasoning with alignment. We can apply any inference rule across multiple datasets and ontologies, which enables all possible alignment updates.

Remove(). Often, the goal is to transform the dataset to a new terminology rather than to just enable queries using the new terminology. The difference between these two scenarios is whether the original instances, using the original terminology, persist in the dataset. Inference rules can transform the dataset by instantiating inferred triples using the new terminology. To support deleting the original triples, we add a function *Remove(triple T)*. Remove, unlike *delete(triple T)* , removes the concrete triple without removing the inferred (transformed) triples.

4 Experimental Results

In [6], we show that we are faster than all existing semantic web repositories using all 26 queries defined by LUBM[3] and Barton Dataset[4]. In fact, we are faster than the next fastest solution, RDF-3X[1], by almost 3x times. There is no standard dataset and set of alignments for us to test our new features with. Our claim therefore is that since we are faster than other repositories for basic queries, and the current technologies for applying alignments (bridges, transformation APIs, etc) are all query time functions, we will also be much faster for queries involving alignment.

We have defined some simple experiments to demonstrate that we can apply alignment matchings to the dataset, and that queries are still efficient. For these experiments, we continued to use LUBM (with 67 million triples) and Barton dataset. Certain relationships between these datasets are natural since university students and

professors commonly author publications. Our tests utilize only the most common alignment matchings: sameAs and subClassOf. However, these tests verify that each of the requirements in Section 1 is satisfied. Performance times are listed in seconds.

The first experiment we performed was simply to align type Person by applying a sameAs inference rule matching the two Person classes. We applied this rule to the two datasets in 0.91s. We were then able to query *?s type Person* in 0.08s.

In the second experiment, we aligned instances. We choose 100,000 specific professors from LUBM and arbitrarily aligned them with 100,000 authors from Barton Dataset in 10.8s. We then queried all members of University0 who wrote a item of type text (0.03s), all professors who wrote a conference publication (0.24s), and all conferences which published a work by a professor (0.13s).

The third experiment we performed was to align the LUBM type Publication. We first aligned this with Item. This alignment took 0.91s. We queried all Items (0.07s) and Publications (0.08s). We then deleted this alignment and added a new alignment between Publication and Text (1.27s to delete and add the alignments). We changed the probability on this alignment from 1.0 to 0.97 (0.38s). We then queried and ranked items of type text(0.06s). The results from the Barton Dataset were first (p=1.0) followed by all items of type Publication from LUBM (p=0.97).

Finally, we used the subject types in Barton to align with the publicationResearch in LUBM. We took this a step further and used this information to guess the subjects associated with research groups and departments in LUBM. For example, if 6/7 publications written by members of a research group were on the topic of data mining, we concluded the research group related to data mining with p= 0.86. The main point in this experiment was to validate our ability to adjust alignments on the fly. For example, matching four researchers with authors increases the number of known topics for a research group and alters the similarity measure of the research group.

5 Conclusion and Future Work

We have defined a set of specific requirements necessary to allow ontology alignment to be applied to data in a semantic web repository. No existing semantic web repositories provide these features. We have presented experiments validating that we do provide these features, and we have presented performance numbers documenting the times required to apply various alignments and to query the results.

For future work, we would like to develop a complete benchmark and perform more elaborate experiments and comparisons. We also plan to use cloud computing solutions such as HBase and Hadoop[5] to further increase RDFKB's scalability.

6 References

1. Neumann, T., Weikum, G.: RDF-3X: a RISC-style engine for RDF. PVLDB(2008) 647-659
2. McGlothlin, J.P., Khan, L.R.: RDFKB: efficient support for RDF inference queries and knowledge management. In IDEAS(2009) 259-266
3. Lehigh University Benchmark (LUBM), http://swat.cse.lehigh.edu/projects/lubm
4. The Barton dataset, http://simile.mit.edu/wiki/Dataset:_Barton
5. Hadoop, http://hadoop.apache.org
6. Mcglothlin, J., Khan, L.: Materializing and Persisting Inferred and Uncertain Knowledge in RDF Datasets. In AAAI(2010)

Ontology Mapping Neural Network: An Approach to Learning and Inferring Correspondences among Ontologies

Yefei Peng[1]*, Paul Munro[1], and Ming Mao[2]

[1] University of Pittsburgh, Pittsburgh PA 15206, USA,
yep3@pitt.edu, pmunro@pitt.edu,
[2] SAP Labs, Palo Alto CA 94304, USA,
ming.mao@sap.com,

Abstract. An ontology mapping neural network (OMNN) is proposed in order to learn and infer correspondences among ontologies. It extends the Identical Elements Neural Network (IENN)'s ability to represent and map complex relationships. The learning dynamics of simultaneous (interlaced) training of similar tasks interact at the shared connections of the networks. The output of one network in response to a stimulus to another network can be interpreted as an analogical mapping. In a similar fashion, the networks can be explicitly trained to map specific items in one domain to specific items in another domain. Representation layer helps the network learn relationship mapping with direct training method.
OMNN is applied to several OAEI benchmark test cases to test its performance on ontology mapping. Results show that OMNN approach is competitive to the top performing systems that participated in OAEI 2009.

Keywords: neural network, ontology mapping, analogy, learning

1 Introduction

Ontology mapping is important to the emerging Semantic Web. The pervasive usage of agents and web services is a characteristic of the Semantic Web. However agents might use different protocols that are independently designed. That means when agents meet they have little chance to understand each other without an"interpreter". Ontology mapping is "a necessary precondition to establish interoperability between agents or services using different ontologies." [2]

The Ontology Mapping Neural Network (OMNN) extends the Identical Elements Neural Network(IENN)'s [3, 4, 1, 5] ability to represent and map complex relationships. The network can learn high-level features common to different tasks, and use them to infer correspondence between the tasks. The learning dynamics of simultaneous (interlaced) training of similar tasks interact at the

* The author is working at Google now. Email: yefeip@google.com

shared connections of the networks. The output of one network in response to a stimulus to another network can be interpreted as an analogical mapping. In a similar fashion, the networks can be explicitly trained to map specific items in one domain to specific items in another domain.

2 Network Architecture

The network architecture is shown in Figure 1. A_{in} and B_{in} are input subvectors for nodes from ontology A and ontology B respectively. They share one representation layer AB_r. RA_{in} represents relationships from graph A; RB_{in} represents relationships from graph B. They share one representation layer R_r.

In this network, each to-be-mapped node in graph is represented by a single active unit in input layers (A_{in}, B_{in}) and output layers (A_{out}, B_{out}). For relationships representation in input layer (RA_{in}, RB_{in}), each relationship is represented by a single active unit.

The network shown in Figure 1 has multiple sub networks shown in the following list.

1. Net_{AAA} : {A_{in}-AB_r-X_{AB}; RA_{in}-R_{RA}-X_R }-H_1-W-H_2-V_A-A_{out};
2. Net_{BBB} : {B_{in}-AB_r-X_{AB}; RB_{in}-R_{RB}-X_R }-H_1-W-H_2-V_B-B_{out};
3. Net_{AAB} : {A_{in}-AB_r-X_{AB}; RA_{in}-R_{RA}-X_R }-H_1-W-H_2-V_B-B_{out};
4. Net_{BBA} : {B_{in}-AB_r-X_{AB}; RB_{in}-R_{RB}-X_R }-H_1-W-H_2-V_A-A_{out};

An explicit cross training method is proposed to train the correspondence of two relationships by directly making their representations more similar. Only a portion of the neural network is involved in this cross training method: the input subvectors and representation layer. For example, we want to train the relationship correspondence of $< R_1, R_2 >$, where R_1 belongs to ontology A and R_2 belongs to ontology B. R_1 will be presented at RA_{in}. The output at R_r will be recorded, which we will name as RR_1. Then R_2 is presented at RB_{in}. RR_1 will be treated as target value at R_r for R_2. Weights RU_B will be modified so that R_1 and R_2 have more similar representation at R_r with standard back propagation method. Then $< R_1, R_2 >$ will be trained so that weight RU_A will be modified to make R_1's representation at R_r similar to that of R_2. The sub networks involved in this training method will be named as $RNet_{AB}$ and $RNet_{BA}$.

Network is initialized by setting the weights to small random values from a uniform distribution. The network is trained with two vertical training tasks (Net_{AAA} and Net_{BBB}), two cross training tasks (Net_{AAB} and Net_{BBA}), and two explicit training tasks ($RNet_{AB}$ and $RNet_{BA}$).

3 Results

Selected OAEI [3] benchmark tests are used to evaluate OMNN approach. All test cases share the same reference ontology, while the test ontology is different.

[3] http://oaei.ontologymatching.org/

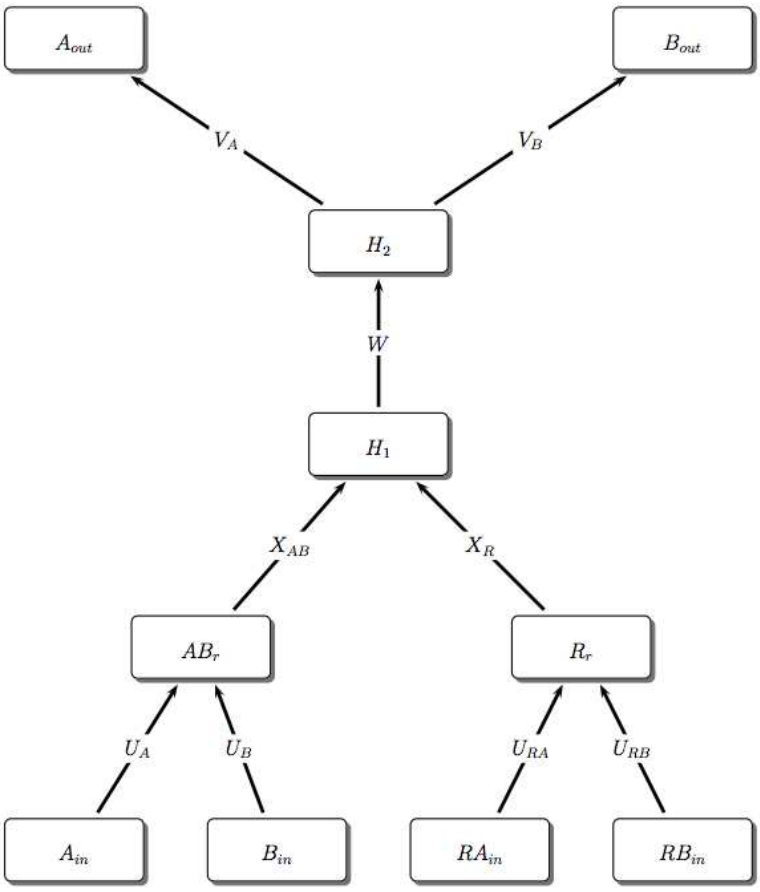


Fig. 1. Proposed network architecture

The reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. In the OMNN approach, classes are treated as items; object properties and data properties are treated as relationships; lastly individuals are not used.

Texture information is used to generate high confident mappings which are then used as cross-training data in OMNN. However OMNN does not focus on how well texture information is used.

In order to compare with other approaches that heavily use texture information, 19 test cases with limited texture information are selected to be used in our experiments. They are test case 249, 257, 258, 265, 259, 266 and their sub-cases.

To get a meaningful comparison, the Wilcox test is performed to compare OMNN with the other 12 systems participated in OAEI 2009 on precision, recall and f-measure. The result is shown in Table 1. It shows that OMNN has better F-measure than 9 of the 12 systems, OMNN's recall is significantly better than 10

of the systems. It should be noted that p-value< 0.05 means there is significant difference between two systems compared, then detailed data is visited to reveal which is one is better than the other.

Table 1. p-value from Wilcox test for 19 benchmark test cases. The green color means that OMNN is significantly better than the system; red color means the system is significantly better than OMNN; yellow means no significant difference. Significance is defined as p-value< 0.05.

System	Precision	Recall	F-Measure
aflood	0.000	0.570	0.182
AgrMaker	0.014	0.000	0.000
aroma	0.420	0.000	0.000
ASMOV	0.000	0.046	0.679
DSSim	0.027	0.000	0.000
GeRoMe	0.042	0.000	0.000
kosimap	0.008	0.000	0.000
Lily	0.000	0.306	0.000
MapPSO	0.000	0.000	0.000
RiMOM	0.136	0.002	0.032
SOBOM	0.811	0.000	0.000
TaxoMap	0.011	0.000	0.000

.

References

1. Bao, J., Munro, P.W.: Structural mapping with identical elements neural network. In: Proceedings of the International Joint Conference on Neural Networks - IJCNN 2006. pp. 870–874. IEEE (2006)
2. Ehrig, M.: Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
3. Munro, P.: Shared network resources and shared task properties. In: Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society (1996)
4. Munro, P., Bao, J.: A connectionist implementation of identical elements. In: In Proceedings of the Twenty Seventh Ann. Conf. Cognitive Science Society Proceedings. Lawerence Erlbaum: Mahwah NJ (2005)
5. Munro, P.W.: Learning structurally analogous tasks. In: Kurková, V., Neruda, R., Koutník, J. (eds.) Artificial Neural Networks - ICANN 2008, 18th International Conference. Lecture Notes in Computer Science, vol. 5164, pp. 406–412. Springer: Berlin/Heidelberg (2008)

Lily-LOM: An Efficient System for Matching Large Ontologies with Non-Partitioned Method

Peng Wang

School of Computer Science and Engineering, Southeast University, China
pwang@seu.edu.cn

Abstract. Since the high time and space complexity, most existing ontology matching systems are not well scalable to solve the large ontology matching problem. Moreover, the popular divide-and-conquer matching solution faces two disadvantages: First, partitioning ontology is a complicate process; Second, it will lead to loss of semantic information during matching. To avoid these drawbacks, this paper presents an efficient large ontology matching system Lily-LOM, which uses a non-partitioned method. Lily-LOM is based on two kinds of reduction anchors, i.e. positive and negative reduction anchors, to reduce the time complexity problem. Some empirical strategies for reducing the space complexity are also discussed. The experiments show that Lily-LOM is effective.

1 Introduction

Since high time and space complexity, most ontology matching systems cannot deal with large ontology matching (LOM) problem. First, matching process requires a large amount of memory space, which would cause the system to crash due to the out of memory error. The space complexity of a matching system usually is $O(n^2)$. Second, most ontology matching algorithms are $O(n^2)$ time complexity, i.e. it needs n^2 times similarity calculations.

Divide-and-conquer strategy is a feasible solution for LOM problem. However, it also has two main issues to be resolved. First, we notice that some ontology partitioning approach cannot control the size of blocks, which may be too small or too large for matching. Second, the ontology partitioning idea also would cause another considerable issue, namely, the partitioning would make the elements on the boundaries of blocks lose some semantic information, that would in turn affect the quality of final matching results.

This paper presents Lily-LOM, a system for matching large ontologies, which is based on a non-partitioned method. Compared with the existing work, Lily-LOM has two distinct advantages: First, it needs not to partition large ontologies but it also has the high performance. Second, it is a general solution for LOM problem, namely, it can adopt most existing matching techniques.

2 Matching Large Ontologies Based on Reduction Anchors

During matching large ontologies, we notice two interesting facts: (1) a large ontology is often composed of the hierarchies organized by *is-a* or *part-of* properties, and a correct alignment should not be inconsistent with such hierarchies; (2) an alignment between two large ontologies has locality, i.e., most elements of region D_i in ontology O_1 will match to the elements of region D_j in ontology O_2. The two facts provide new ways for finding efficient solution about LOM.

In Fig. 1. (a), if high similarity values exist between a_i and b_p or b_q, we can decide that a_i matches b_p or b_q. This decision will bring a direct benefit: the subsequent similarity calculations between sub-concepts(/super-concepts) of a_i and super-concepts(/sub-concepts) of b_p or b_q can be skipped. This paper calls such concept pairs like (a_i, b_p) the positive reduction anchors(P-Anchors), which employ ontology hierarchy feature to reduce the time complexity in LOM.

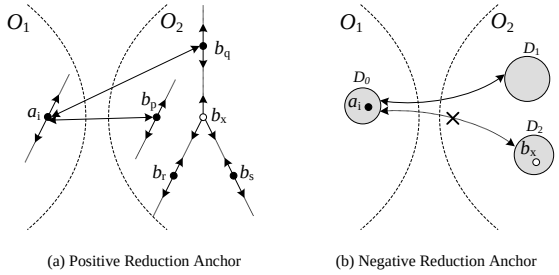


Fig. 1. Reduction anchors in large ontology matching

Fig. 1. (b) shows the locality phenomenon in LOM, where D_i refers to a region in the ontology. Suppose a_i in D_0 does not match b_x in D_2, then we can infer that the neighbors of a_i do not match b_x too, i.e., the similarity values between them are very low. As a result, we can skip the subsequent similarity calculations between the neighbors of a_i and b_x, which can also reduce the times of similarity calculations. This paper calls such concept pairs like (a_i, b_x) the negative reduction anchors(N-Anchors).

P-Anchors and N-Anchors provide two ways to design new efficient solutions for LOM. Based on the two kinds of anchors, matching process can skip many times of similarity calculations to reduce the time complexity significantly. Obviously, P-Anchors and N-Anchors cannot be identified in advance, so it needs to discover them dynamically in matching, then uses the anchors to predict the ignorable similarity calculations.

Let the P-Anchors of a_i is $PA(a_i) = \{b_1, b_2, ..., b_k\}$. We call all the ignorable similarity calculations predicted by $PA(a_i)$ the positive reduction set of a_i. The corresponding reduction set can be calculated by following formula, in which lub denotes least upper bound and glb is greatest lower bound.

$$PS(a_i) = [sub(a_i) \otimes sup(lub(b_1, ..., b_k))] \cup [sup(a_i) \otimes sub(glb(b_1, ..., b_k))]$$

We can prove that when the order of similarity calculations can divide the hierarchy path L into equal parts continually, the P-Anchors can generate the

maximum valid positive reduction set with $|L| * (|L| - 2)$ size [1]. It means the algorithm has the best time complexity $O(2n)$. Generally, the algorithm has $O((1 - \frac{\bar{d}}{n})n^2)$ time complexity, where $\bar{d}$ is the average depth of the ontology.

N-Anchors can also predict the ignorable similarity calculations, which are called the positive reduction set. If (a_i, b_j) is a N-Anchor, we can predict that neighbors of a_i are also irrelevant to b_j. The set of all ignorable similarity calculations predicted by this way are called the negative reduction set.

Let $NA(a_i)$ refer to the N-Anchors about a_i, the neighbors with $nScale$ distance to a_i constitute a set $Nb(a_i) = \{a_x | d(a_x, a_i) <= nScale\}$, the negative reduction set generated by a_i is:$NS(a_i) = NA(a_i) \otimes Nb(a_i)$. The time complexity of the algorithm is $O(\alpha n^2)$, where α is in $[0, 1]$ and is determined by size of negative reduction set.

3 Empirical Space Complexity Processing

Besides the time complexity, the space complexity is another challenge in LOM. We present some empirical methods for handling the space complexity problem, and it may be useful for other matching systems. The number of elements in large ontology is large, so we should avoid allocating a $n \times n$ similarity matrix. Considering the similarity matrix is a typical sparse matrix, it can adopt the compression techniques to replace it. It usually compresses a similarity matrix into several MBs. In our LOM algorithms, the size of reduction set will become bigger and bigger, which takes a large amount of space. We first replace the two dimension reduction set with one dimension style, then merge the continuous number of elements as a link. Memory space resource is valuable in LOM, so if a variable or a data structure is unused, we should free its space immediately. This principle will reduce the possibility of out of memory error.

4 Experimental Evaluations

All algorithms proposed in this paper are implemented in ontology matching system Lily-LOM. More information about Lily can be found at http://cse.seu.edu .cn/people/pwang/lily.htm.

We get some matching results on several real large ontologies by participating in OAEI[1]. Here we present the results of our LOM algorithms on three LOM tasks (*Anatomy*, *Fao*, and *Library*) in OAEI2008.

From 2007 to 2008 years, there are 13 systems participated in the anatomy task, but only three systems: Lily, Falcon-AO, and TaxoMap, used the special large ontology matching method. Falcon-AO proposed a divide-and-conquer method called PBM algorithm. TaxoMap uses the PBM algorithm, so it is a re-implement of PBM. We measure quality of the results with the classic F1-measure, and use *Recall+* [2] to measure how many non trivial correct alignments can be found.

Table 1 shows the results of three LOM systems. According to the results, we have four conclusions: (1) Lily is one of the LOM system can perform well in *Anatomy* task. (2) For the three LOM systems, Lily and Falcon-AO have similar quality, which are better than TaxoMap. (3) The running time of Lily has two parts: the special matcher used in Lily takes 3.1 hours for the preprocessing, but the matching computing and postprocessing only spend 13 minutes. It indicates that if we use other literal-based matchers, we would have close running time

[1] Ontology Alignment Evaluation Initiative http://oaei.ontologymatching.org/

Table 1. Matching results of systems on Anatomy

System	Runtime	Precision	Recall	F-Measure	Recall+
Label Eq.	–	0.981	0.613	0.755	0.000
Lily	3.1h+13min	0.796	0.693	0.741	0.470
Falcon-AO	12min	0.963	0.599	0.738	0.127
TaxoMap	25min	0.460	0.764	0.574	0.470

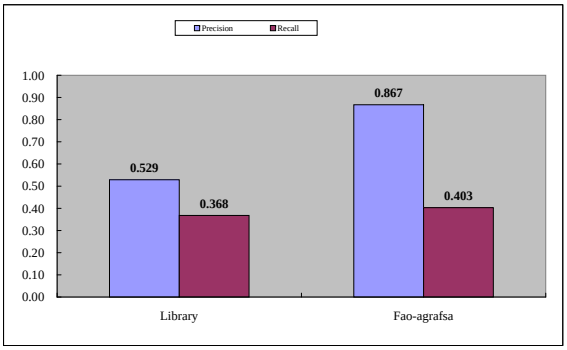


Fig. 2. Matching results of Lily on the real large ontologies

to other systems. (4) Lily and Taxomap have high Recall+ value, it means that they have the ability to discover the difficult alignments, but Lily has better F-measure.

The results of Lily on the Library and Fao tasks are showed as Fig. 2, which also demonstrates that it can discover some alignments in the two tasks.

5 Conclusion

This paper present a system Lily-LOM, which proposes a new large ontology matching method based on reduction anchors. The reduction anchors are useful to predict the ignorable similarity calculations during matching, that can reduce the high time complexity problem.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (61003156).

References

1. Peng Wang and Baowen Xu. Matching large ontologies. Technical Report WIP-TR-2009-02, Southeast university, http://cse.seu.edu.cn/people/pwang/publication/WIP-TR-2009-02.pdf, 2009.
2. Caterina Caracciolo, Jrme Euzenat, Laura Hollink, Ryutaro Ichise, and et al. Results of the ontology alignment evaluation initiative 2008. In *The Third International Workshop on Ontology Matching (OM2008), Karlsruhe, Germany.*, 2008.

Toward Seoul Road Sign Management on the LarKC Platform

Tony Lee[1], Stanley Park[1], Zhisheng Huang[2] and Emanuele Della Valle[3 4]

[1] Saltlux Inc., Seoul, Korea
[2] Vrije University Amsterdam, Netherlands
[3] CEFRIEL – Politecnico of Milano, Via Fucini 2, 20133 Milano, Italy
[4] Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy
{tony, hgpark}@saltlux.com, huang@cs.vu.nl, emanuele.dellavalle@polimi.it

Abstract. Geo Semantic Technology is evaluated as the core technology for supporting interoperability of geospatial data and building urban computing environment. We made semantic integrations of LOD's Linked Geo Data and Open Street Map with Korean POI data set, and have researched for developing intelligent road sign management system based on the LarKC platform.

Keywords: Geo Semantics, Semantic Web, Reasoning,

1 Introduction

The LarKC[5] project's main goal is to develop a platform for reasoning using massive amounts of heterogeneous information [1]. The platform has a pluggable architecture to exploit techniques and heuristics from diverse areas such as databases, machine learning, and the Semantic Web.

Within LarKC, we are running an Urban Computing [2] use case, with the aim of proving the commercial feasibility of the LarKC platform and its architectures when applied to huge geo semantic and urban data sets. In particular, in this poster, we briefly present our efforts toward developing real-world Road Sign Management systems (RSM) for Seoul.

2 Motivation

A typical building in South Korea is described by the administrative divisions[6] in which it lies rather than street names. If the address is written in Korean, the largest division will be written first, followed by the smaller divisions, and finally the

[5] http://www.larkc.eu
[6] http://en.wikipedia.org/wiki/Administrative_divisions_of_South_Korea

building and the recipient, in accordance with the East Asian addressing system. Divisions could be identified after the name of the nearest point of interest (POI).

The problem is that Korean cities grow and evolve much faster than western cities. POIs may move, new roads may be built, and road signs may be changed accordingly. Effectively managing road signs, in particular validating if a sequence of road signs leads to a given address, is a major problem. For this reason the Korean Road Traffic Authority maintains a database of all Seoul road signs. The directions given on each road sign are formally described together with their actual location.

3 Research Challenges

Effective management of road signs requires processing the directions that are given on each road sign together with a large amount of urban-related information. Until a couple of years ago the cost of obtaining and maintaining up-to-date urban-related information was high; but nowadays it is much less expensive thanks to collaborative projects such as Open Street Map[7] (OSM), which creates free editable maps of the world, or Wikipedia, where POI descriptions can be found.

In the LarKC project, we have investigated a data integration solution for urban information [3] that can provide a basis for intelligent road sign management. The solution supports data modeling and the integration of massive amounts of linked geo-data, POI data, and road sign data, as well as scalable querying and reasoning.

4 Data Integration

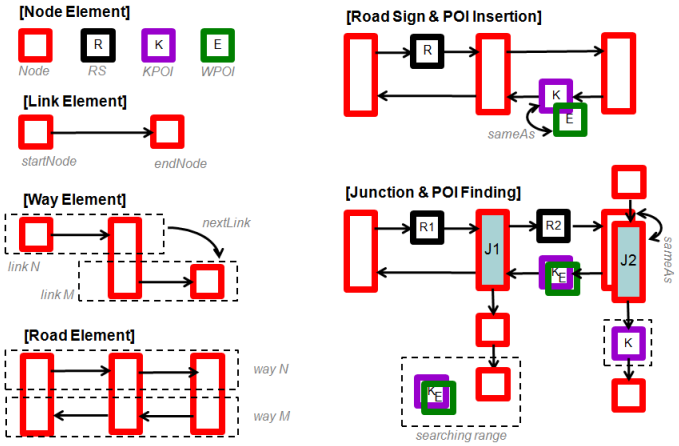


Fig. 1. The mediation ontology we use in our system.

[7] http://www.openstreetmap.org/

The data set we are manipulating contains about 1.1 billion triples. 2 million triples describe the streets of Seoul, and were directly extracted from OSM. 4 million triples describe POIs related to road signs and come from the Korean Road Traffic Authority database. Half million triples describe road signs and also come from the Korean Road Traffic Authority database.

The data were integrated using the mediation ontology illustrated in Figure 1. Roads are modeled as a sequence of nodes and links. Four types of node are modeled: the generic nodes that can identify either a junction between multiple roads or a bend in a road; the road sign (RS) nodes that indicate the presence of a road sign; the Korean POIs (KPOI) that indicate POIs from the Korean Road Traffic Authority database; and the Wikipedia POIs (WPOI) that indicate POIs from Wikipedia (obtained through DBpedia). A way is composed of links. A road is composed of ways. Road signs and points of interest are placed along the roads. If KPOIs and WPOIs are understood to be same, owl:sameAs is used to state it. Due to quality issues in the OSM data set, not all the junctions are explicitly stated; where necessary owl:sameAs is also used to state that two nodes are the same node and, thus, that a junction is present among multiple roads. Finally, note that not all POIs are directly on the roads - some of them may be places nearby a node in the road.

5 Queries and Reasoning for RS validation

In Figure 2, we give an idea of the queries and reasoning required for the intelligent management of the road signs that we are developing. Sign R1 indicates that two POIs (i.e., G1 and G2) can be found straight ahead. R2 indicates that POIs G2 and G3 are straight ahead while G1 can be reached by turning right. R3 indicates that G3 is straight ahead, while G2 can be reached by turning right.

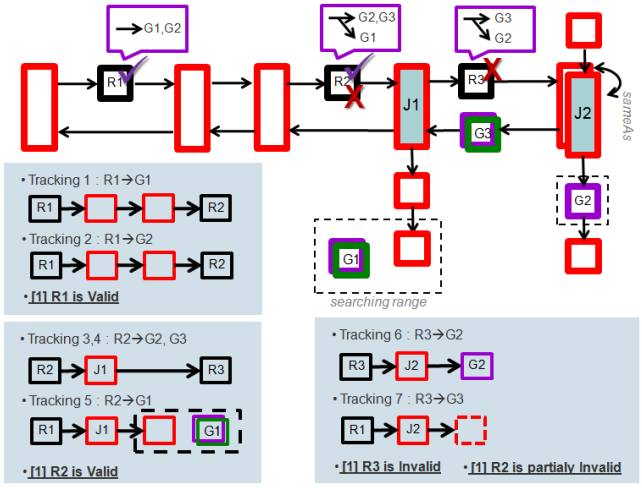


Fig. 2. Road sign validations by using SPARQL and OWL Horst Reasoning

In the boxes we show the validation process of the three road signs. The directions for G1 and G2 on R1 are valid because going straight in two nodes we can find road sign R2 that contains further indications for G1 and G2. Similarly the direction for G2 and G3 on R2 are valid because going straight at the junction we can find road sign R3 that contains further indications for G2 and G3. The direction for G1 on R2 is also valid, because turning right, G1 can be found near the second node of the street. The direction for G2 on R3 is valid, but the direction for G3 is not valid: G3 is reachable only by executing a U-turn. This also means that the direction for G3 on R2, which we previously stated to be valid, is not valid because it refers to a road sign R3 which is not valid.

Current implementation of the RSM system is based on 40 SPARQL queries executed under the Owl Horst entailment regime including some axioms of rdfs:subClassOf, rdfs:subPropertyOf, owl:inverseOf and owl:sameAS. They encode 30 Korean road sign regulations related to positioning and naming.

4 Conclusions and Future work

In the work done so far we have found several data quality issues, which are due to the presence in OSM of a lot of useless information, together with poor accuracy and missing data. We have partially solved this issue by cleaning up data manually; automatic cleaning support is under investigation. Data errors are also present in the Korean Road Traffic Authority database data set in terms of direction and location errors. An important issue is that different names for the same POI are present both in OSM and in the Korean Road Traffic Authority database, e.g., Seoul Univ. and Seoul National University. We used a semi-automatic technique to assert owl:sameAs relationships. Finally, we are investigating techniques for rewriting SPARQL queries in SQL/MM Spatial for efficiently evaluating SPARQL queries that manipulate geographic knowledge [4] including real world reasoning with noisy data.

Acknowledgments. The work described in this poster has been partially supported by the European LarKC project (FP7-215535).

References

1. Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Valle, E.D., Fischer, F., Huang, Z., Kiryakov, A., il Lee, T.K., Schooler, L., Tresp, V., Wesner, S., Witbrock, M., Zhong, N.: Towards larKC: A platform for web-scale reasoning. In: ICSC, IEEE Computer Society (2008) 524–529
2. Kindberg, T., Chalmers, M., Paulos, E.: Guest editors' introduction: Urban com- puting. IEEE Pervasive Computing 6(3) (2007) 18–20
3. Tony Lee, Albert Ahn and Saung hoon Lee : Semantic Search and Data Interoperability for GeoWeb: The 14th International Seminar on GIS (2010)
4. Della Valle, E., Celino, I., Dell'Aglio, D.: The experience of realizing a semantic web urban computing application. T. GIS 14(2) (2010) 163–181

MoKi: a Wiki-Based Conceptual Modeling Tool

Chiara Ghidini, Marco Rospocher, and Luciano Serafini

FBK–irst, Via Sommarive 18 Povo, I-38123, Trento, Italy

Abstract. The success of wikis for collaborative knowledge construction is triggering the development of a number of tools for collaborative conceptual modeling based on them. In this paper we present a completely revised version of MoKi, a tool for modelling ontologies and business process models in an integrated way.

1 Introduction

MoKi [1] is a collaborative MediaWiki-based[2] tool for modeling ontological and procedural knowledge in an integrated manner. The main idea behind MoKi is to associate a wiki page, containing both unstructured and structured information, to each entity of the ontology and process model. In this section we present a completely revised version of MoKi, which extends the first release of the tool (see [1]). The main changes w.r.t. [1] are (i) the redesign of the content organisation of the MoKi page, which now comprises an unstructured part and a structured part (this extends and replaces the simple representational languages used in [1]); and (ii) the multi-mode access to the page content, to support easy usage both by domain experts and knowledge engineers, thus facilitating them to play an equally central role in the modelling activities (this extends and replaces the single template-based access mode provided in [1]).

1.1 The MoKi page

Being a tool supporting the description of ontological and procedural knowledge according to the OWL Web Ontology Language and the Business Processes Modelling Notation (BPMN), the basic element for MoKi are *concepts*, *properties*, and *individuals* in the ontology, and *processes* in the process model. Each instance of these elements is therefore associated to a MoKi page, composed of an unstructured part and a structured part.

The unstructured part This part contains text written following the standard MediaWiki markup format: in particular, it can contain plain text, possibly enriched by formatting information, links to other MoKi pages or to external resources, uploaded images, and so on. The format of this part of the page is the same for all the different elements of the models.

The structured part This part, which is delimited by specific tags to separate it from the unstructured text, contains knowledge stored according to the modelling language adopted. In the current implementation, the structured part of a page describing an ontology element contains a RDF/XML serialisation of a set of OWL statements formalising the element, while, similarly, the structured part of a page describing a BPMN process contains an XML serialisation of the process diagram.

[1] See `http://moki.fbk.eu`.

[2] See `http://www.mediawiki.org`.

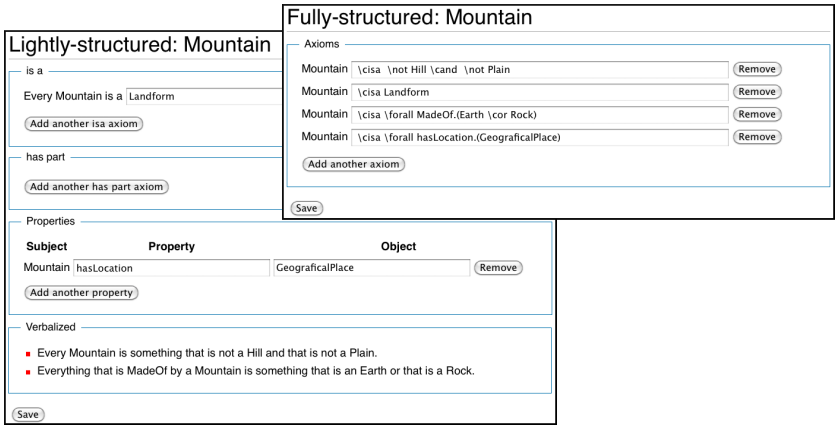


Fig. 1. Lightly-structured access mode and Fully-structured access mode for ontology concepts.

1.2 Supporting multi-mode access in MoKi

Users can access the ontological and procedural knowledge contained in MoKi using three different access modes: one mode, the *unstructured access mode*, to access the unstructured part of a MoKi page, and two different modes, the *fully-structured access mode* and the *lightly-structured access mode*, to access the structured part.

The unstructured access mode This access mode allows the user to edit/view the content of the unstructured part of the MoKi page of a model element. The editing/viewing of this part occurs in the standard MediaWiki way.

The fully-structured access mode This access mode allows the user to edit/view the content of the structured part of a MoKi page using the full expressivity of the modelling language adopted. For ontological knowledge the fully-structured access mode allows the user to view/edit formal statements (axioms) describing the element associated to the page. Axioms are written according to the *latex2owl* syntax[3], an intuitive latex-style format for writing ontologies using a text-editor, format which can be automatically translated into (an RDF/XML serialisation of) OWL. The user can easily edit the list of axioms in a form based interface, as shown in Figure 1 (right). For procedural knowledge we have implemented an access mode that, by tightly integrating in MoKi the Oryx editor[4], a full-fledged business process editor, allows the user to edit the BPMN process diagram described in the page, as shown in Figure 2 (up).

The lightly-structured access mode The purpose of this access mode is to allow users with limited knowledge engineering skills, to edit/view the content of the structured part of the MoKi page in a simplified and less formal way. For ontological knowledge the lightly-structured access mode is provided through a form made of two components, as depicted in Figure 1 (left). In the top half part the user can view and edit simple statements which can be easily converted to/from OWL statements. If the OWL version of any of these statements is already contained in the structured part of the page, then the

[3] See `http://dkm.fbk.eu/index.php/Latex2owl`
[4] See `http://bpt.hpi.uni-potsdam.de/Oryx/`

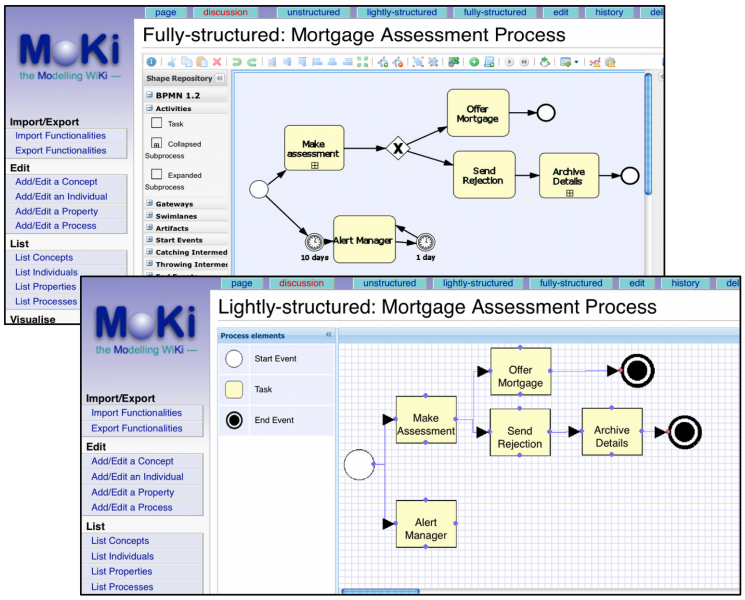


Fig. 2. Fully-structured access mode and lightly-structured access mode for processes.

corresponding fields are pre-filled with the appropriate content. Similarly, when any of these simple statements is modified in the lightly-structured access mode, the changes are propagated to the content of the structural part of the page. The bottom half of the form provides a description of those OWL statements which cannot be intuitively translated/edited as simple statements as the ones in the top half of the page. In the current implementation, this part contains the translation of those statements in Attempto Controlled English, provided by the OWL 2 Verbalizer[5]. For procedural knowledge we have implemented an access mode based on a light-weight graphical process editor which uses a restricted subset of process building blocks, as shown in Figure 2 (down).

1.3 Model Overview pages in MoKi

Model Overview pages are MoKi special pages dynamically created from the (structured) content of the pages describing model elements. For ontological knowledge, the model overview pages allow to explore the generalisation and part/subparts decomposition hierarchies of ontology concepts, as well as the classification of the ontology individuals. In particular, MoKi provides two kinds of model overview pages. In the tabular-based one, the user can access a table listing every concept (resp. individual) of the ontology together with the concepts of which it is a specialisation and the concepts in which it decomposes according to the part of relation (resp. the concepts to which the individual belongs to). In the graphical-based one, a tree-like view shows the hierarchy of concepts according to either the subclass or the part-of relation, or the membership of individuals to concepts. Drag and drop editing facilities are also provided to rearrange

[5] See `http://attempto.ifi.uzh.ch`.

the tree. For procedural knowledge, the model overview page provides an overview of the process/sub-process decomposition mechanism by means of a table listing every process defined in MoKi together with the processes in which it decomposes.

1.4 Usages of MoKi

The different versions of MoKi have been applied in several scenarios. Focusing on the usages with real domain experts, MoKi has been successfully applied by four application partners within the FP6 EU-project APOSDLE[6] to develop enterprise models (composed of a domain ontology and a process model) in six different domains, and it is currently used by a team of knowledge engineers and domain experts to collaboratively build an Organic Agriculture and Agroecology Ontology within the FP7 EU-project Organic.Edunet[7].

Although the version of the tool here presented is tailored to the development of ontologies and business processes, the tool can be customized to support modelling other kinds of knowledge. For example, a preliminary version of the tool which support modelling of clinical protocols according the ASBRU language is described in [2].

2 Conclusions

In this paper we have presented a completely revised version of MoKi, a tool for modelling ontologies and business process models in an integrated way. The main novelties of the tool w.r.t. the previous version presented ([1]), are (i) a different content organisation of the page, which now comprises an unstructured part and a structured part, and (ii) the implementation of a multi-mode access to the page content, to support easy usage both by domain experts and knowledge engineers, thus facilitating them to play an equally central role in the modelling activities.

In our future work, we aim at improving the support for process modeling, in particular in providing an extensive automatic support for aligning the fully-structured access mode and lightly-structured access mode in case of procedural knowledge. We also aim at evaluating our tool further, on larger case studies.

Acknowledgements

The work described in this paper has been partially funded by the European Commission under the contract number FP7-248594.

References

1. Ghidini, C., Kump, B., Lindstaedt, S., Mahbub, N., Pammer, V., Rospocher, M., Serafini, L.: MoKi: The Enterprise Modelling Wiki. In: Proceedings of ESWC 2009. Volume 5554 of LNCS., Springer (2009) 831–835 Demo Session.
2. Eccher, C., Ferro, A., Seyfang, A., Rospocher, M., Miksch, S.: Modeling clinical protocols using semantic MediaWiki: the case of the Oncocure project. In: ECAI workshop on Knowledge Management for Healthcare Processes (K4HelP). (2008)

[6] See `http://www.aposdle.org/`
[7] See `http://www.organic-edunet.eu/`

Publishing Bibliographic Data on the Semantic Web using BibBase

Reynold S. Xin[×], Oktie Hassanzadeh[+], Christian Fritz[§]
Shirin Sohrabi[+], Yang Yang[+], Minghua Zhao[+], Renée J. Miller[+]

[+]Department of Computer Science, University of Toronto
{oktie,sohrabi,c7yangya,mzhao,miller}@cs.toronto.edu
[×]Department of EECS, University of California, Berkeley
rxin@berkeley.edu
[§]Information Sciences Institute, University of Southern California
fritz@isi.edu

Abstract. We present BibBase, a system for publishing and managing bibliographic data available in BibTeX files on the Semantic Web. BibBase uses a powerful yet light-weight approach to transform BibTeX files into rich *Linked Data* as well as custom HTML and RSS code that can readily be integrated within a user's website. The data can instantly be queried online on the system's SPARQL endpoint. In this demo, we present a brief overview of the features of our system and outline a few challenges in the design and implementation of such a system.

Keywords: Bibliographic Data Management, Linked Data, Data Integration

1 Introduction

Management of bibliographic data has received significant attention in the research community. Many online systems have been designed specifically for this purpose, e.g., BibSonomy [9] and CiteSeer [10]. The work in the semantic web community in this area has also resulted in several tools (such as BiBTeX to RDF conversion tools [5]), ontologies (such as SWRC [7] and the Bibliographic Ontology [6]) and data sources (such as DBLP Berlin [8]). These systems, tools, and data sources are widely being used and have considerably simplified and enhanced many bibliographic data management tasks such as data curation, storage, retrieval, and sharing of bibliographic data.

Despite the success of the above-mentioned systems, very few individuals and research groups publish their bibliographic data on their websites in a structured format, particularly following the principles of Linked Data [1] which mandate the use of HTTP dereferenceable URIs and structured (RDF) data to convey the semantics of the data This is mainly due to the fact that existing systems either are not designed to be used within an external website, or they require expert users to set up complex software systems on machines that meet the requirements of this software. BibBase aims to fill this gap by providing several distinctive features that our demo will illustrate.

2 Light-weight Linked Data publication

BibBase makes it easy for scientists to maintain publication lists on their personal web site. Scientists simply maintain a BiBTeX file of their publications, and BibBase does the rest. When a user visits a publication page, BibBase dynamically generates an up-to-date HTML page from the BiBTeX file, as well as rich Linked Data with resolvable URIs that can be queried instantly on the system's SPARQL endpoint. We have chosen to use an augmented version of MIT's BiBTeX ontology definition to publish data in RDF[1].

Compared to existing Linked Data publication tools, this approach is notably easy-to-use and light-weight, and allows non-expert users to create a rich linked data source without any specific server requirements, the need to set up a new system, or define complex mapping rules. All they need to know is how to create and maintain a BiBTeX file and there are tools to help with that.

It is important to note that this ease of use does not sacrifice the quality of the published data. In fact, although the system is light-weight on the users' side, BibBase performs complex processing of the data in the back-end. When a new or updated BiBTeX file arrives, the system transforms the data into several structured formats using our ontology, assigns URIs to all the objects (authors, papers, venues, etc.), performs duplicate detection and semantic linkage, and maintains and publishes provenance information.

3 Duplicate Detection

BibBase needs to deal with several issues related to the heterogeneity of records in a single BiBTeX file, and across multiple BiBTeX files. BibBase uses existing duplicate detection techniques in addition to a novel way of managing duplicated data following the Linked Data principles.

Within a single BiBTeX file, the system uses a set of rules to identify duplicates and fix errors. For example, if a BiBTeX file has two occurrences of author names "J. B. Smith" and "John B. Smith", the system matches the two author names and creates only a single author object. In this example, the assumption is that the combination of the first letter of first name, middle name, and last name, "JBSmith", is a unique identifier for a person in a single file.

For identification of duplicates across multiple BiBTeX files, the assumptions made for local duplicate detection may not hold. Within different publication lists, "JBSmith" may (or may not) refer to the same author. BibBase deals with this type of uncertainty by having a *disambiguation* page on the HTML interface that informs the users looking for author name "J. B. Smith" (by looking up the URI `http://data.bibbase.org/author/j-b-smith`) of the existence of all the entities with the same identifier, and having `rdfs:seeAlso` properties that link to related author entities on the RDF interface.

[1] Notably, the MIT's BiBTeX ontology (`http://zeitkunst.org/bibtex/0.1/`) is extended to allow description of the order of authors, unlike some widely-used bibliographic ontologies. We also provide `owl:sameAs` and `umbel:isLike` links to the other existing bibliographic ontologies. The new ontology definition is available at `http://data.bibbase.org/ontology`.

Duplicate detection, also known as *entity resolution*, *record linkage*, or *reference reconciliation* is a well-studied problem and an active research area [3]. We use some of the existing techniques to define local and global duplicate detection rules, for example using fuzzy string similarity measures [2] or semantic knowledge for matching conference names and paper titles [4].

4 Discovering semantic links to external data sources

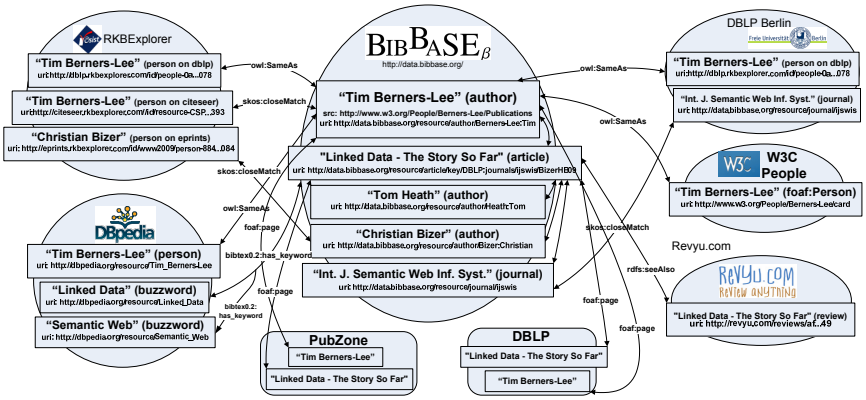


Fig. 1. Sample entities in BibBase interlinked with several related data sources.

In order to publish our data *in the Web*, not just *on the Web*, to avoid creation of an isolated data silo, we need to discover links from the entities in BibBase to entities from external data sources. Figure 1 shows a sample of entities in BibBase and several possible links to related Linked Data sources and web pages. In order to discover such links, similar to our duplicate detection approach, we can leverage online and offline solutions. The online approach mainly uses a dictionary of terms and strings that can be mapped to external data sets. A similar approach is used to match abbreviated venues, such as "ISWC" to "International Semantic Web Conference". The dictionaries (or ontology tables) are maintained inside BibBase, and derived from sources such as DBpedia, Wordnet, and DBLP. We also allow the users to extend the dictionaries by `@string` definitions in their BiBTeX files. Offline link discovery is performed using existing link discovery tools [4]. Our demo will allow users to interactively add BiBTeX entries, then view and query the semantically annotated entry and discovered links.

5 Additional Features

The success of BibBase as a Linked Data source depends on scientists using BibBase for their publications pages. To further entice scientists to do so, BibBase sports a number of additional features that make it an attractive proposition.

- Storage and publication of provenance information, i.e., metadata about the source of each entity and each link in the data.
- Dynamic grouping of entities based on attributes (e.g., by year or keyword).
- An RSS feed, allowing anyone to receive notifications whenever a specified scientist publishes a new paper.
- A DBLP fetch tool that allows scientists who do not yet have a BiBTeX file to obtain their DBLP publications to start using BibBase right away.
- Statistics regarding users, page views, and paper downloads.

We enable users to provide feedback on the quality of data and links. By providing feedback, users will not only improve the quality of the data published on their own websites, they will also help create a very high-quality data source in the long run that could become a benchmark for the notoriously hard task of evaluating duplicate detection and semantic link discovery systems.

6 Conclusion

In this demonstration, we will present BibBase, a system for light-weight publication of bibliographic data on personal or research group websites, and management of the data using existing semantic technologies as a result of the complex *triplification* performed inside the system. BibBase extends the Linked Data cloud with a data source that unlike existing bibliographic data sources, allows online manipulation of the data by non-expert users. We plan to continue to extend the features of BibBase. A list of currently implemented and upcoming experimental features is available at `http://wiki.bibbase.org`.

References

1. T. Berners-Lee. Linked Data - Design Issues. `http://www.w3.org/DesignIssues/LinkedData.html`, 2006. [Online; accessed 14-June-2010].
2. A. Chandel, O. Hassanzadeh, N. Koudas, M. Sadoghi, and D. Srivastava. Benchmarking Declarative Approximate Selection Predicates. In *ACM SIGMOD Int'l Conf. on the Mgmt. of Data*, pages 353–364, 2007.
3. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.
4. O. Hassanzadeh, A. Kementsietsidis, L. Lim, R. J. Miller, and M. Wang. A Framework for Semantic Link Discovery over Relational Data. In *Proc. of the Conf. on Information and Knowledge Management (CIKM)*, pages 1027–1036, 2009.
5. I. Herman. BibTeX in RDF. `http://ivan-herman.name/2007/01/13/bibtex-in-rdf/`, 2007. [Online; accessed 14-June-2010].
6. `http://bibliontology.com/`.
7. `http://ontoware.org/swrc/`.
8. `http://www4.wiwiss.fu-berlin.de/dblp/`.
9. `http://www.bibsonomy.org/`.
10. `http://citeseer.ist.psu.edu/`.

Visualizing Populated Ontologies with OntoTrix

Benjamin Bach[1,3], Gennady Legostaev[2,1], and Emmanuel Pietriga[1]

[1] INRIA and LRI (Université Paris-Sud & CNRS), Orsay, France
benjamin.bach@lri.fr, emmanuel.pietriga@inria.fr
[2] St Petersburg State University, Saint Petersburg, Russia
glegostaev@gmail.com
[3] Dresden University of Technology, Germany

Abstract. Most tools for visualizing Semantic Web data structure the representation according to the concept definitions and interrelations that constitute the ontology's vocabulary. Instances are often treated as somewhat peripheral information, when considered at all. The visualization of instance-level data poses different but significant challenges as instances will often be orders of magnitude more numerous than the concept definitions that give them machine-processable meaning. We present a visualization technique designed to visualize large instance sets and the relations that connect them. This visualization uses both node-link and adjacency matrix representations of graphs to visualize different parts of the data depending on their semantic and local structural properties, exploiting ontological knowledge to drive the layout of, and navigation in, the visualization.

1 Introduction

By making use of the rich capabilities of graphical representations and by abstracting from the complex syntactic details of textual formats, visual tools aim at providing better cognitive support to users, from knowledge engineers to domain-expert end-users. They provide them with interactive representations of the data based upon state-of-the art information visualization techniques, better supporting tasks such as ontology understanding, discovery, search, comparison and mapping [3].

Most tools structure the visualization according to the concept definitions and interrelations that constitute the ontology's vocabulary. While many of them do support the visualization of instance data, instances are often treated as somewhat peripheral information. Employing Description Logics terminology, the visualization is mainly structured according to the TBox, instances that constitute the ABox being treated as leaf nodes in this tree or graph structure. Exceptions to this general observation exist, but either give a limited view of the ABox [1,6] or use conventional node-link diagram representations that hardly scale beyond a few hundred nodes at best [4].

Instances represent an essential part of the overall knowledge base. Compared to the definition of concepts based on OWL constructs, instance data are at a lower level of abstraction. But more importantly, instance datasets are often orders of magnitude larger (see, e.g., many of the datasets currently part of the *Linking Open Data* graph). As such, the visualization of instance-level data poses different but real challenges that remain to be addressed. We present OntoTrix, a visualization technique designed to enable users to visualize, and navigate in, large instance sets and their relations.

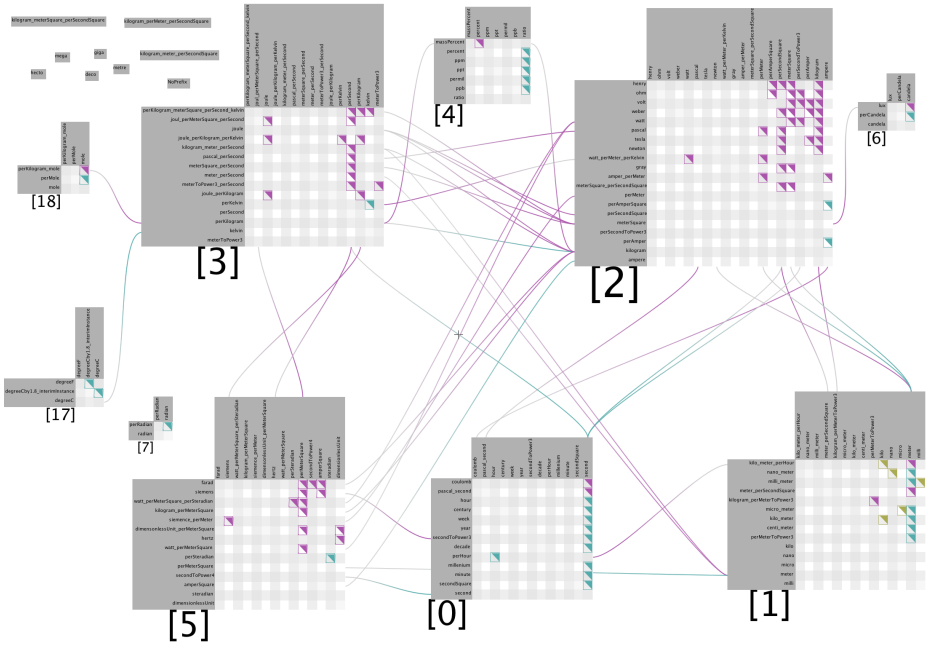


Fig. 1. Overview of SWEET's units.owl ontology with OntoTrix.

2 OntoTrix

Ontology graphs contain many nodes and edges, and are often non planar. Two main issues with node-link diagram representations of such graphs are their inefficient use of screen real-estate and edge crossings that make dense regions difficult to read, both eventually causing scalability problems. A well-known alternative to node-link diagrams for graph visualization are adjacency matrices. Nodes are represented as rows and columns, and edges as filled cells at the intersection of connected rows and columns. While node-link diagrams are good at showing the structure of relatively small and sparse graphs, adjacency matrices are very effective at showing large (better use of screen real-estate) and dense (no edge crossing) graphs. However, adjacency matrix representations are much less familiar to users than node-link diagrams, and make tasks that involve following paths in the graph more difficult [2].

OntoTrix is inspired by a recent hybrid network visualization technique, NodeTrix [2], that uses both node-link and adjacency matrix representations to visualize different parts of the data depending on their semantic and structural properties (Figure 1). The technique has proven successful at handling large networks, being very efficient at visualizing locally dense but globally sparse networks. It displays the overall structure of the network using a node-link diagram, and the dense subgraphs that represent communities using matrices. While the graph structure of ontologies might not always share the small-world characteristics of social networks, such a hybrid representation, combined with appropriate interaction techniques, can be an efficient means to perform exploratory visualization of large ontology instance sets.

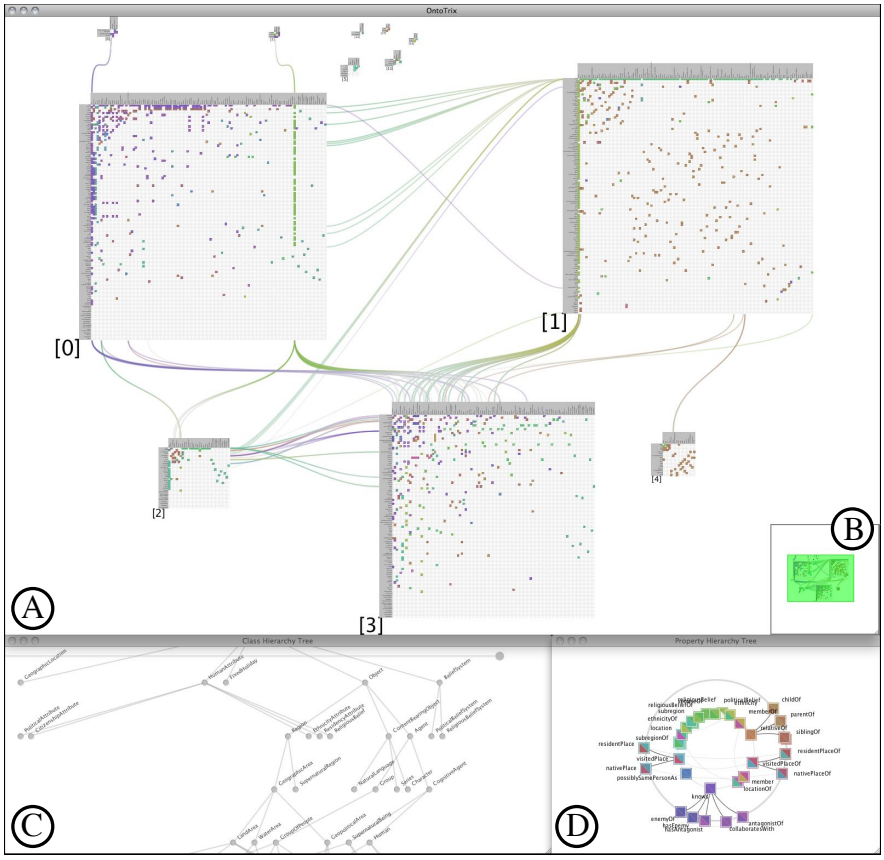


Fig. 2. OntoTrix Interface Overview: (A) Main NodeTrix view, (B) Bird's eye view, (C) Class hierarchy view, (D), Property hierarchy view.

NodeTrix was originally devised for undirected social network structures that only feature one type of node and one type of relation. We extend NodeTrix to handle the much richer and complex graph structure of populated ontologies, exploiting ontological (TBox) knowledge to drive the layout of, and navigation in, the representation. Embedded in a smooth zoomable interface [5], and coupled with visual representations of the class and property hierarchies that enable interactive navigation and filtering of instance data, this visualization produces more compact and legible representations than node-link diagram approaches, thus better scaling to large instance sets.

The OntoTrix environment features four main views (Figure 2). The main view (A), contains the OntoTrix representation of the instance set. (B) provides an interactive bird's eye view of (A). The class hierarchy is visualized in (C), the property hierarchy in (D). These views are highly synchronized. For instance, hovering a node in the class hierarchy view highlights all corresponding instances in the OntoTrix view. Hovering a node in the property hierarchy view highlights all corresponding elements in matrices, as well as corresponding edges between matrices. Classes declared as being in the domain or range of the property are highlighted in the class hierarchy, and conversely.

Matrices in NodeTrix basically correspond to highly-connected groups of actors, i.e., dense subgraphs that represent social communities. In OntoTrix, we propose different methods for grouping instances into matrices, yielding very different perspectives on the dataset. Instances can be clustered in matrices based on edge density, taking into account all types of relations between instance nodes (Figure 1). A second method groups instances into matrices according to class membership. A third method represents a tradeoff between the previous two. From an initial grouping based on density, all nodes are grouped together on a per-matrix basis according to class membership as described above (reordering columns and rows). Each of the original matrices can then be split into smaller matrices corresponding to class membership groups. The last method groups all instances involved in statements based on selected object property type(s).

Beyond improvements in terms of readability of a graph's structure, NodeTrix features an interesting property that is related to the above view of matrices as aggregates of instance nodes for the layout process. Often, these aggregates will by themselves represent interesting entities, not explicitly represented in the ontology, but that bear semantics. When grouping by class membership in OntoTrix, matrices obviously represent groups of similar instances. We thus label each matrix with the associated class name. When grouping by other methods, matrices cannot easily be tagged as there is no explicit information about the grouping that stems from the purely structural clustering of the graph. Matrices might still represent interesting entities, but will have to be labeled manually and are currently assigned a random identifier. For instance, the matrices in Figure 1 illustrate interesting grouping patterns: matrix [0] contains mostly time-related units, matrix [1] mostly energy-related units, matrix [6] units related to measuring light, etc. These groupings are very different from what is obtained when grouping by class membership, and give an interesting perspective on the dataset. By supporting dynamic, smoothly-animated transitions between grouping methods, OntoTrix allows for rapid switching between these perspectives.

OntoTrix is implemented in Java, using the ZVTM zoomable user interface toolkit, LinLogLayout for layout and clustering, Jena 2 for parsing ontologies, and the TDB backend for storing. Jena's OWL transitive reasoner provides a complete classification of the ontology. Additional reasoning can be performed using other reasoners.

References

1. Fluit, C., van Harmelen, F., Sabou, M.: Supporting user tasks through visualisation of lightweight ontologies. In: Handbook on Ontologies in Info. Systems. Springer-Verlag (2003)
2. Henry, N., Fekete, J.D., McGuffin, M.J.: Nodetrix: a hybrid visualization of social networks. IEEE Transactions on Visualization and Computer Graphics 13(6), 1302–1309 (2007)
3. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods—a survey. ACM Computing Surveys 39(4), 10:1–10:42 (2007)
4. Noppens, O., Liebig, T.: Interactive Visualization of Large OWL Instance Sets. In: International Workshop on the Semantic Web and User Interaction (SWUI) (2006)
5. Pietriga, E.: A toolkit for addressing hci issues in visual language environments. In: Symp. on Visual Languages and Human-Centric Computing (VL/HCC'05). pp. 145–152. IEEE (2005)
6. Tu, K., Xiong, M., Zhang, L., Zhu, H., Zhang, J., Yu, Y.: Towards imaging large-scale ontologies for quick understanding and analysis. In: ISWC. pp. 702–715. Springer-Verlag (2005)

BRAMBLE: A Web-based Framework for Interactive RDF-Graph Visualisation

Nikolas Schmitt, Mathias Niepert, and Heiner Stuckenschmidt

KR & KM Research Group
University of Mannheim

Abstract. Most graph visualisation tools for RDF data are desktop applications focused on loading complete ontologies and metadata from a file and allowing users to filter out information if needed. Recently both scientific and commercial frameworks have started to shift their focus to the web, however they still rely on plugins such as Java and rarely handle larger collections of RDF statements efficiently.
In this abstract we present a framework which visualises RDF graphs in a native browser environment, leveraging both the SVG standard and JavaScript technology to provide a responsive user interface. Graphs can be directly expanded, modified and explored. Users select nodes and edges from a central data repository containing millions of statements. The resulting graph can be shared with other users retaining full interactivity for collaborative work or presentation purposes.

1 Introduction

With the growth of datasets such as the Linked Open Data project[1], finding and presenting relevant information becomes an increasingly complex process. Graphs are well suited to browse and visualise linked data and a wide range of tools exist that provide graph drawing functionality on various platforms and in a myriad of formats. In particular there are commercial products[2] integrating graphs with web applications, allowing users to view and interact with a graph using their web-browser. So far these solutions rely on either a heavy server back-end generating the graph visualisation as images and sending them to the client, which imposes heavy restrictions with regard to interactivity of the graph, or they use a Java Applet or similar plugin which has to be downloaded first and does not run natively in the browser. The *WiGis* framework [1] shifts the entire workload onto the server and sends images to the browser at a constant frame rate after every user interaction. While this does solve some performance and scalability problems with large graphs, it makes the user interface feel slow and unresponsive since any actions taken by the user have to be evaluated by the server before there can be a response. All of the graph visualisation frameworks

[1] http://linkeddata.org/
[2] Tom Sawyer Software: Tom Sawyer Visualization (http://www.tomsawyer.com) TouchGraph: TouchGraph Navigator (http://www.touchgraph.com)

rely on the "Overview - Filter" approach where the user is presented a graph over the entire collection of nodes and edges and is then expected to filter out the information he needs. This method has some inherent disadvantages: The whole graph has to be loaded and drawn up front. Thus frameworks employing this approach can only process a few hundred nodes and even those developed for large graphs have difficulties handling more than 10,000 nodes at an acceptable speed.

In this abstract we demonstrate how the *Bramble Framework*[3] implements web-based graph visualisation aiming for fully interactive graphs displayed in a native, browser-based environment. After the user has selected graph nodes and edges through a browser-based graphical user interface, the data is retrieved from a server-based RDF repository. Graphs are presented directly in a browser using Scalable Vector Graphics (SVG). The user may expand the graph by adding new nodes that are connected to existing nodes or explore the graph dynamically by moving from node to node. Attributes are displayed for nodes and edges and the user may influence their visualisation (colour, text) or delete them from the graph.

2 User Interface

After connecting to the data repository of choice, the user can utilise the *Chain Manager* dialog to retrieve specific information. The process is started by defining a *Node Set*: one or more entities from the datastore that share common features.

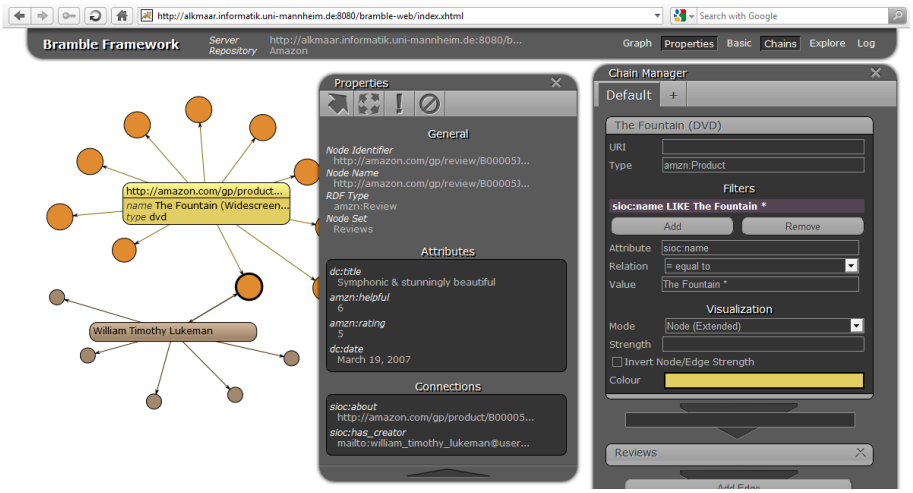


Fig. 1. The framework interface with Chain Manager and Properties dialog open.

[3] http://alkmaar.informatik.uni-mannheim.de:8080/bramble

Entities will be represented in the graph as nodes. Figure 1 shows a node set which contains entities whose `rdf:type` conforms to a product from the Amazon.com database and whose `sioc:name` matches the pattern "The Fountain*". Relevant classes and relations are suggested by the interface based on the RDF schema definitions present in the datastore. A set of nodes can be chained to another set either by specifying a relation that connects the two or leaving the field blank, in which case the framework will retrieve any relations between both sets. In Fig. 1 reviews (orange) are added to the graph by attaching a node set to the chain which automatically selects entities linked to the product (yellow).

To enhance the visualisation of the graph the user may customise the style of each individual node set and the edges that connect them. Customisation includes changing the colour and size of nodes, the strength of edges and adding textual descriptions to entities. The graph layout is determined by choosing one of the available algorithms such as the Kamada-Kawai [2] algorithm and optionally tweaking their parameters.

Once the graph is generated the user may select any graph element to retrieve additional information. Clicking nodes opens a property dialog displaying all links to other resources. These links can be added to the graph with their predicate becoming an edge and their object becoming a new node. Any graph element can be transformed into a more detailed version of itself, displaying information directly in the graph, or be removed from the graph.

3 Architecture and Implementation

The server-client architecture depicted in fig. 2 offers many advantages for a graph visualisation framework: Users upload data they want to analyse to a central repository. Any users with access to the repository can now use their browser to construct and customise graphs. Saving their work to the server allows them to continue editing from another computer or platform and sharing fully interactive graphs with other users.

A Java web service is used as the interface between the OpenRDF Sesame repository and the client web application. It connects to the repository, retrieves data and relays them to the web application using JSON. The web service takes *Chains* as input and translates them into SeRQL queries. The query results are parsed and transformed into node and edge objects which the web application can use to draw the graph.

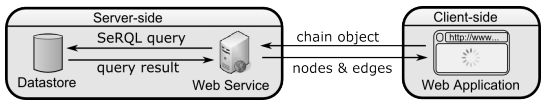


Fig. 2. Server-Client architecture with data query.

The web application is served as XHTML markup with a large collection of JavaScript aided by popular libraries such as jQuery. No server page technology is used, the communication to the server relies sololy on AJAX requests to the web service. The entire process of evaluating the data, applying custom style rules, calculating the graph layout using the algorithm of choice and drawing

the graph is done on the client-side. The graph rendering engine generates the SVG markup (which uses XML syntax) and embeds scripts and triggers to enable interactivity as well as RDF metadata.

4 Use Case

In order to test and evaluate the framework an example repository has been created using the data gathered from the Amazon.com database by the Multi-Domain Sentiment Data Analysis project [3]. The repository contains millions of product reviews which provide information on how highly certain products are regarded by different groups of users.

The data on products from the category DVD make up for almost 100,000 reviews written by 50,000 users about 13,000 products. Visualising all the information in one graph is not only technically infeasible but also useless as a medium for research. The *Bramble Framework* enables analysts to search for information about one certain product and visualise how well it is received by users. Different styles can be used to identify quickly which customers are in favour of a product and which are not. Analysts are able to display additional information about a customer who dislikes a certain product and find out if and why another product is preferred. This leads to identifying strong market contenders and fields to improve the product in.

5 Conclusion

The *Bramble Framework* demonstrates how graphs can be visualised over the web while retaining full interactivity and a responsive user interface. Using SVG to render nodes and edges has several advantages over plugins such as Java or Adobe Flash: graphs can be exported as SVG to be modified on a per-element basis by external programs and scaled without loss of quality. In addition, metadata can be embedded into the XML markup and the technology is supported by all major browsers and platforms. The client-server approach with a web-based frontend makes data in existing web repositories more accessible. Furthermore, complex relationships in datasets consisting of millions of statements can be visualised using the frameworks flexible selection method.

References

1. B. Gretarsson, S. Bostandjiev, J. O'Donovan and T. Höllerer: WiGis: A Framework for Scalable Web-based Interactive Graph Visualizations. (2009)
2. T. Kamada and S. Kawai: An algorithm for drawing general undirected graphs. Inf. Process. Lett., 31, 7–15 (1989)
3. J. Blitzer, M. Dredze and F. Pereira: Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In ACL, 187–205 (2007)

A web-based Evaluation Service for Ontology Matching

Jérôme Euzenat[1], Christian Meilicke[2],
Heiner Stuckenschmidt[2], Cássia Trojahn[1]

[1] INRIA & LIG, Grenoble
[2] University of Mannheim

Abstract. Evaluation of semantic web technologies at large scale, including ontology matching, is an important topic of semantic web research. This paper presents a web-based evaluation service for automatically executing the evaluation of ontology matching systems. This service is based on the use of a web service interface wrapping the functionality of a matching tool to be evaluated and allows developers to launch evaluations of their tool at any time on their own. Furthermore, the service can be used to visualise and manipulate the evaluation results. The approach allows the execution of the tool on the machine of the tool developer without the need for a runtime environment.

1 Introduction

Evaluation of matching tools aims at helping designers and developers of such tools to improve them and to help users to evaluate the suitability of the proposed methods to their needs. The Ontology Alignment Evaluation Initiative (OAEI)[3] has been the basis for evaluation over the last years [1]. It is an annual evaluation campaign that offers several data sets organized by different groups of researchers. However, additional effort has to be made in order to catch up with the growth of ontology matching technology. The SEALS project[4] aims at providing standardized datasets, evaluation campaigns for typical semantic web tools and, in particular, a software infrastructure for automatically executing evaluations. In this context, we have developed a web-based evaluation service that allows developers to launch their own evaluations at any time while using a set of approved datasets. It is based on the use of a web service interface wrapping the functionality of a matching tool to be evaluated. In the following, we describe the main components of our service and present a complete evaluation example.

2 Evaluation Service Architecture

The evaluation service is composed of three main components: a web user interface, a BPEL workflow and a set of web services. The web user interface is the

[3] `http://oaei.ontologymatching.org/`
[4] Semantic Evaluation at Large Scale `http://about.seals-project.eu/`

entry point to the application. This interface is deployed as a web application in a Tomcat application-server behind an Apache web server. It invokes the BPEL workflow, which is executed on the ODE[5] engine. This engine runs as a web application inside the application server.

The BPEL process accesses several services that provide different functionalities. The *validation service* ensures that the matcher web service is available and fullfills the minimal requirements to generate an alignment in the correct format. A *redirect service* is used to redirect the request for running a matching task to the matcher service endpoint. The *test iterator service* is responsible for iterating over test cases and providing a reference to the required files. The *evaluation service* provides measures such as precision and recall for evaluating the alignments generated by the matching system. A *result service* is used for storing evaluation results in a relational database.

Once the web service matcher implementation has been deployed and published at a stable endpoint by the tool developer, its matching method can be invoked within the BPEL workflow. For that reason, an evaluation starts by specifying the web service endpoint via the web interface. This data is then forwarded to BPEL as input parameters. The complete evaluation workflow is executed as a series of calls to the services listed above. The specification of the web service endpoint becomes relevant for the invocation of the validation and redirect services. They implement internally web service clients that connect to the URL specified in the web user interface.

Test and result services used in the BPEL process require to access additional data resources. For accessing the test data, the test web service can access the metadata describing the test suite, extracts the relevant information and forwards the URLs of the required documents via the redirect service to the matcher, which is currently evaluated. These documents can then be accessed directly via a standard HTTP GET-request by the matching system. The result web service uses a connection to the database to store the results for each execution of an evaluation workflow. For visualizing and manipulating the stored results we use an OLAP (Online Analytical Processing) application, which accesses the database for retrieving the evaluation results. Results can be re-accessed at any time e.g., for comparing different tool versions against each other.

3 Evaluating a Matching Tool

For demonstration purposes, we have extended the Anchor-Flood [2] system with the web service interface[6]. This system has participated in the two previous OAEI campaigns and is thus a typical evaluation target. The current version of the web application described in the following is available at `http://seals.inrialpes.fr/platform/`. At `http://alignapi.gforge.inria.fr/`

[5] ODE BPEL Engine `http://ode.apache.org/`

[6] Available at `http://mindblast.informatik.uni-mannheim.de:8080/sealstools/aflood/matcherWS?wsdl`

tutorial/tutorial5/, there is complete information about how to create a valid matcher.

In order to start an evaluation, one must specify the URL of the matcher service, the class implementing the required interface and the name of the matching system to be evaluated (Figure 1). Three of the OAEI datasets have been selected, namely Anatomy, Benchmark and Conference. In this specific example, we have used the conference test case.

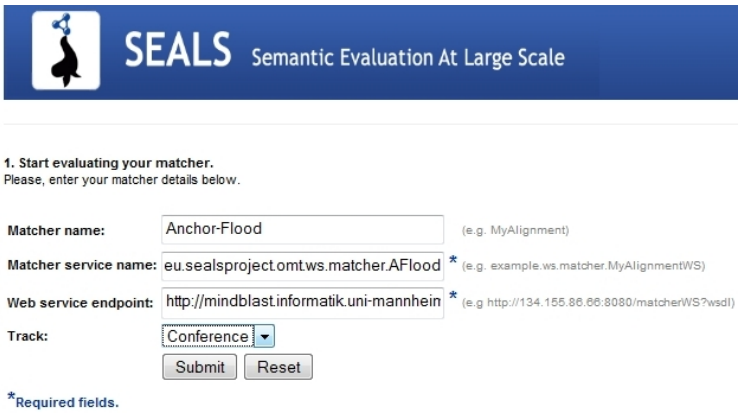


Fig. 1. Specifying a matcher endpoint as evaluation target.

Submitting the form data, the BPEL workflow is invoked. It first validates the specified web service as well as its output format. In case of a problem, the concrete validation error is displayed to the user as direct feedback. In case of a successfully completed validation, the system returns a confirmation message and continues with the evaluation process. Every time an evaluation is conducted, results are stored under the enpoint address of the deployed matcher (Figure 2).

The results are displayed as a table (Figure 3), when clicking on one of the three evaluation IDs in Figure 2. The results table is (partially) available while the evaluation itself is still running. By reloading the page from time to time, the user can see the progress of an evaluation that is still running. In the results table, for each test case, precision and recall are listed. Moreover, a detailed view on the alignment results is available (Figure 4), when clicking on the alignment icon in Figure 3.

4 Final Remarks

Automatic evaluation of matching tools is a key issue for promoting the development of ontology matching. In this demo we have presented a web-based tool for automatic evaluation of matching systems that is available for the research community at any time. The major benefit of this service is to allow developers to debug their systems, run their own evaluations, and manipulate the results immediately in a direct feedback cycle.

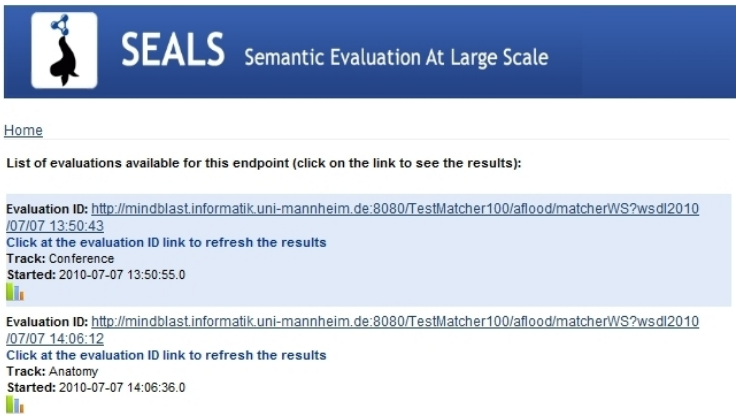


Fig. 2. Listing of available evaluation results.

Test	Precision	Recall	Status	Alignment
cmt-conference	0.30	0.38	completed	
cmt-confOf	0.45	0.31	completed	
cmt-edas			not completed	
cmt-ekaw			not completed	
cmt-iasted			not completed	
cmt-sigkdd			not completed	
conference-confOf			not completed	
conference-edas			not completed	
conference-ekaw			not completed	

Fig. 3. Display results of an evaluation.

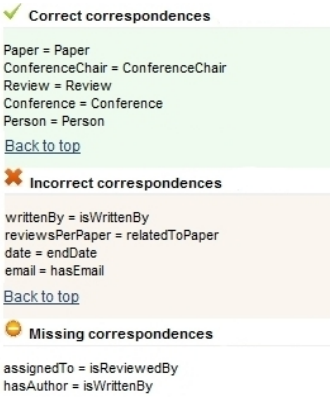


Fig. 4. Detailed view on an alignment.

Acknowledgements The authors are partially supported by the SEALS project (IST-2009-238975).

References

1. J. Euzenat, A. Ferrara, L. Hollink, V. Malaisé, C. Meilicke, A. Nikolov, J. Pane, F. Scharffe, P. Shvaiko, V. Spiliopoulos, H. Stuckenschmidt, O. Sváb-Zamazal, V. Svátek, C. T. dos Santos, and G. Vouros. Results of the ontology alignment evaluation initiative 2009. In *Ontology Matching Workshop*, 2009.
2. H. Seddiqui and M. Aono. Anchor-flood: results for OAEI 2009. In *Proceedings of the ISWC 2009 workshop on ontology matching*, Washington DC, USA, 2009.

SemWebVid - Making Video a First Class Semantic Web Citizen and a First Class Web Bourgeois*

Thomas Steiner,

Google Germany GmbH, ABC-Straße 19, 20354 Hamburg, Germany
tsteiner@{google.com, lsi.upc.edu}¶

Abstract. SemWebVid[1] is an online Ajax application that allows for the automatic generation of Resource Description Framework (RDF) video descriptions. These descriptions are based on two pillars: first, on a combination of user-generated metadata such as title, summary, and tags; and second, on closed captions which can be user-generated, or be auto-generated via speech recognition. The plaintext contents of both pillars are being analyzed using multiple Natural Language Processing (NLP) Web services in parallel whose results are then merged and where possible matched back to concepts in the sense of Linking Open Data (LOD). The final result is a deep-linkable RDF description of the video, and a "scroll-along" view of the video as an example of video visualization formats.

Keywords: RDF, LOD, Linked Data, Semantic Web, NLP, Video

1 Introduction

Over recent years the use of Resource Description Framework (RDF) in documents has gained massive popularity with even mainstream media[2] picking up stories of big companies deploying RDF on their Web presence. However, these efforts have mainly concentrated on textual documents in order to annotate concepts like shop opening hours, prices, or contact data. Far fewer occurrences can be noted for RDF video description on the public Web. Related efforts are automatic video content extraction, or the W3C Ontology for Media Resource.

The development of SemWebVid was driven by the following objectives:

* This work is partly funded by the EU FP7 I-SEARCH project (project reference 248296)

¶ The author is currently a PhD student at Universitat Politècnica de Catalunya, Department LSI, Campus Nord, Edifici Omega, Jordi Girona 1-3, 08034 Barcelona, Spain. We would like to thank Michael Hausenblas from DERI Galway for the review of this work and paper.

[1] Live demo at http://tomayac.com/semwebvid/, username: iswc2010, password: iswc2010

[2] http://www.nytimes.com/external/readwriteweb/2010/07/01/01readwriteweb-how-best-buy-is-using-the-semantic-web-23031.html

- Improve **searchability** of video content by extraction of contained entities and disambiguation of those entities (for queries like *videos of Barack Obama where he talks about Afghanistan while being abroad*).
- Enable **graphical representations** of video content through symbolization of entities (for e.g. video archives of keynote speeches where one could *graphically skim through long video sections at a glance*).

These goals can be reached through RDF video descriptions and we thus developed SemWebVid to create RDF video descriptions in a potentially automatable way based on live data found on YouTube.

2 SemWebVid Dataflow

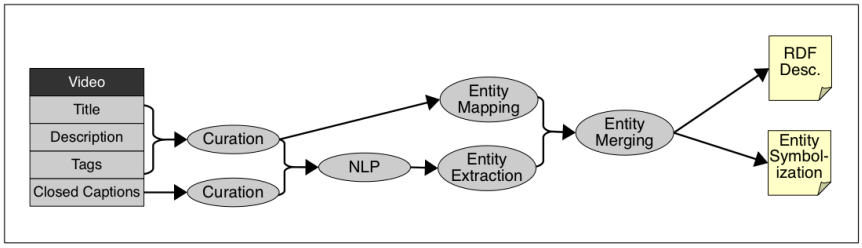


Fig. 1. SemWebVid Dataflow Diagram

The raw data for the RDF video descriptions consist of the beforementioned two pillars: the user-generated video title, video (plaintext) description, and tags on the one hand, and the user- or auto-generated[3] closed captions on the other.

3 Curation of Raw Data

Before the entity mapping and extraction steps, the raw data need to be curated. While the video titles are typically trouble-free as they are usually very descriptive, the main problem with the plaintext video descriptions is that sometimes they get abused for non-related spam-like messages or comments rather than providing a proper summary of the video content. Unfortunately this is hard to detect, so in the end we decided to simply use them as is. With regards to tags the main issue are different tagging styles. As an example see potential tags for the concept of the person Barack Obama:

- `"barack", "obama"` (all words separated, 2 tags)
- `"barack obama"` (space-separated, 1 tag)
- `"barackobama"` (separate words concatenated, 1 tag)

The first split style is especially critical if complete phrase segments are expressed in tag form:

[3] YouTube allows for auto-generation of closed captions through speech recognition: http://youtube-global.blogspot.com/2010/03/future-will-be-captioned-improving.html

- `"one", "small", "step", "for", "a", "man"`

For our demonstration we use an API from Bueda[4] in order to split combined tags into their components and try to make sense of split tags. We use Common Tag to represent tags in the application.

The curation step for closed captions mainly consists of removing speaker and hearable events syntax noise from the plaintext contents, and obviously the cues (time markers for each caption). This can be easily done using regular expressions, the syntax being a variation of `">>Speaker:"` and `"[Hearable Event]"`.

4 Entity Extraction and Mapping

We try to map the list of curated tags back to entities using plaintext entity mapping Web services[5] from DBpedia [1], Sindice [2], Uberblic, and Freebase. This works quite well for very popular tags (samples below from the DBpedia URI Lookup Web service, all results are prefixed with `http://dbpedia.org/resource/`"):

- `"barack obama" => Barack_Obama`

It somewhat succeeds for very generic tags (though with obvious ambiguity issues):

- `"obama" => Obama,_Fukui`

It fails for specific tags (`"ggj09"` was a tag for the event "Global Game Jam 2009"):

- `"ggj09" => N/A`

It is thus very important to preserve **provenance** data in order to judge and estimate the quality of the mapped entities. With regards to the curated closed captions, description, and title we work with NLP Web services[6], namely OpenCalais, Zemanta, and AlchemyAPI. For the test cases we used (famous speeches, keynotes) results were relatively accurate from our judgments. In a final step the detected entities are merged, and a symbolization for each entity gets retrieved by means of a heuristic approach, including Google image search.

Fig. 2. Graphical symbolizations of several entities (TimBL, Semantic Web, etc.)

5 Description of the SemWebVid Demonstration

SemWebVid is designed to be an online Ajax application for interactive use. Unfortunately the terms and conditions of some of the NLP Web services involved do

[4] http://www.bueda.com/developers
[5] http://platform.uberblic.org, http://www.freebase.com, http://sindice.com, http://dbpedia.org
[6] http://opencalais.com, http://zemanta.com, http://alchemyapi.com

not allow for a SemWebVid API, however, due to its design both on-the-fly RDF description generation and permanent linking to previous descriptions are possible.

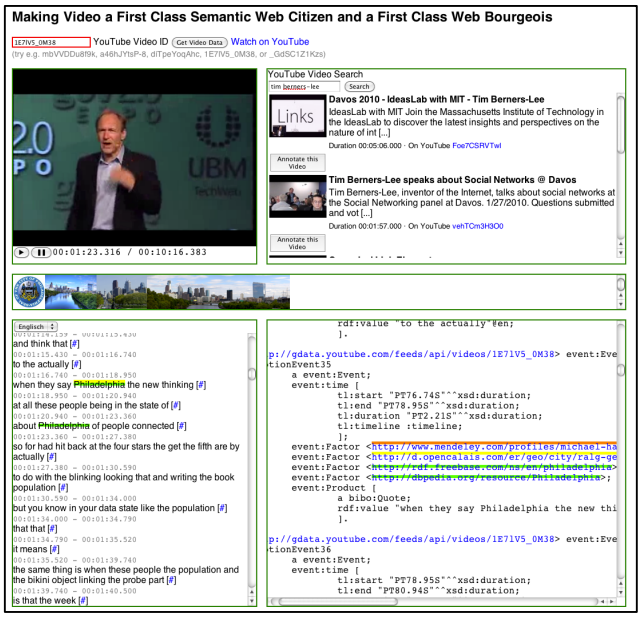


Fig. 3. SemWebVid screenshot showing Tim Berners-Lee's infamous potato chips speech at gov2.0 Expo 2010. Below the video box the concept of the city of Philadelphia is symbolized. The left lower box shows the closed captions directly, the right box the RDF description.

6 Conclusion and Future Work

While we are not the first[7] to connect RDF (and thus Linked Data) with video, SemWebVid's contribution is to present an automatic text-based way to generate RDF video descriptions. Future work is among other things to determine whether the expected searchability improvements pay off the high processing efforts.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Proc. of 6th Int. Semantic Web Conf., 2nd Asian Semantic Web Conf., November 2008, pp. 722–735.
2. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com – A Document-oriented Lookup Index for Open Linked Data. International Journal of Metadata, Semantics and Ontologies, 3 (1), 2008.

[7] Sack, H: http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_ITS/papers/Harald/DSMSA09.pdf

RExplorator - supporting reusable explorations of Semantic Web Linked Data.

Marcelo Cohen, Daniel Schwabe

Pontifical Catholic University of Rio de Janeiro
R. M. S. Vicente 225
Gávea, Rio de Janeiro, RJ, Brazil
+55 21 3527-1500
{mcohen21@gmail.com, dschwabe@inf.puc-rio.br}

Abstract. This demo presents RExplorator, an environment which allows non-technically savvy users, but who understand the problem domain, to explore the data until they understand its structure They employ a combination of search, query and faceted navigation in a direct manipulation, query-by-example style interface. In this process, users can reuse previously found solutions by other users, which may accomplish sub-tasks of the problem at hand. It is also possible to create an end-user friendly interface to allow them to access the information. Once a solution has been found, it can be generalized, and optionally made available for reuse by other users. This enables the establishment of a social network of users that share solutions for problems in particular domains (repositories) of interest.

Keywords. RDF, exploratory search, exploration, ontology, semantic web, reuse, interface, set-based navigation

1 Introduction

The availability of Linked Open Data in the WWW has increased tremendously[1]. Currently, when building a new application, it is becoming increasingly common to first explore available data that can be leveraged to enhance and complete one's own data to provide the desired functionality. The BBC Music website[2] is one visible example of this approach, combining MusicBrainz and DBPedia with their own data.

In previous work we developed Explorator [1], a model for representing information processing by users in exploratory tasks, and its associated tool, which

[1] http://linkeddata.org
[2] http://www.bbc.co.uk/music

provides a browser interface supporting this model. Explorator is based on the metaphor of direct manipulation of information in the interface, with immediate feedback of user actions. In this demo we present RExplorator[3], a significant extension of Explorator, with new features illustrated in this demo.

2 REXPLORATOR

Consider two simple tasks to be carried out over the "Dogfood" data server[4], containing collected publication information for several conferences related to the Semantic Web - Finding all publications of a given author and finding co-workers of a given researcher, and their publications We assume the user has no prior knowledge about the contents of this repository.

For the first task, the user has to

1. Find a class that represents persons
2. Find the desired person, "a".
3. Find a property "p" that relates a person to publications,
4. Find all triples of the form <a p ?pub> and collect all objects from these triples.

In Explorator, this is achieved by first clicking on "Menu"-> All RDF Classes", noticing class Person, mousing over it to click on "All Instances", which reveals a set of all Persons. Double-clicking on a Person (e.g. "Steffen Staab"), a new box appears with all details for this resource (i.e., all triples with this resource as subject). Looking at the details, one notices the property "made", which relates Person to Publications. To get all publications by a Person, one may click on the "Selected Person Details" box, and click on the "S" operand position at the top; click on the "made" box and click on the "P" operand position at the top, and then clicking on the "=" ("compute") operator at the top.

For the second task, the user has to follow a similar set of steps, re-using task 1 once the co-workers have been found.

RExplorator extends Explorator by (we don't detail all for space reasons)

1. Allowing operations to be parameterized;
2. Allowing the results of a query to be fed as input of another query, thus forming graphs of interconnected operations;
3. Allowing keeping such graphs as separate workbenches, while enabling interconnection of graphs across workbenches;

[3] Available at http://www.tecweb.inf.puc-rio.br/rexplorator, which includes a demo video. See also a demo of Explorator - http://blip.tv/play/ AennPpPBIw

[4] http://data.semanticweb.org

4. Allowing the designer to import previously defined query graphs into the current workbench;
5. Allowing the designer to define additional operators beyond the set operations provided;
6. Allowing the designer to define interfaces oriented towards end users, hiding details and customizing the look-and-feel.

The original Explorator metaphor lets users compose operations incrementally, seeing the results at each composition step. Each new query takes its operands from existing query results. In the end, one may regard this set of inter-related operations as a graph, similar to an Excel spreadsheet. However, the operations are all grounded, which would be akin to not having any variables in the formulas of the analogous spreadsheet. Thus, the first generalization made was to allow operations to have its operands parameterized, and to propagate values trough the graph of operations when the value of the parameter is changed. This is equivalent to introducing variables in the expression that denotes the operation.

Consider step 4 in task 1, finding all publications of a Person. In Explorator, this is achieved by selecting an instance of Person (e.g., "Steffen Staab in box "All Persons") in Figure 1, setting it as the subject parameter, selecting the relation "make" as the property parameter, and clicking on the "=" operator to find all triples of the form <<url for Steffen Staab> made? o> clicking on the  icon in each box, as shown in Figure 1 reveals the actual operations and their dependencies .

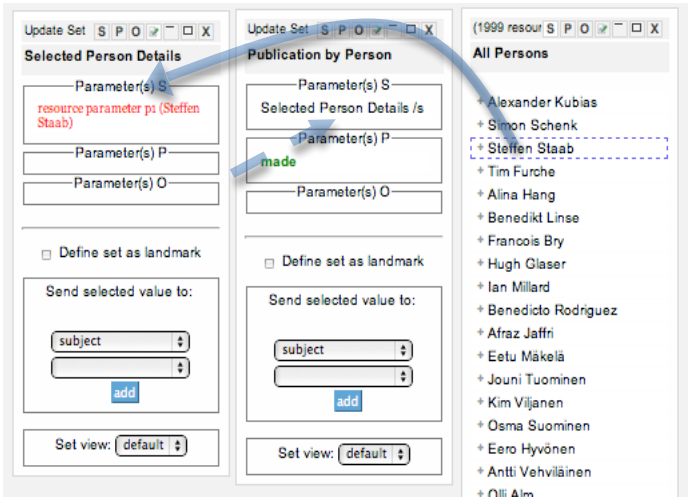


Figure 1 – Query structure and dependencies

The first box, Selected Person Details, represents the query that finds out all triples with a given Person as subject. Notice that the first position, "S", has been

parameterized, and the current parameter value is (the URI for) Steffen Staab. If we drag any person from the rightmost box (All Persons) onto the "S" position in the Selected Person Details box, the value is replaced and the query re-evaluated.

The Publication by Person query (middle box) is defined as taking its "subject" parameter from the "subject" position of the Selected Person Details query. Therefore, if a new value is plugged into the "S" position in the Selected Person Details query, it is automatically propagated to this query, which triggers its reevaluation.

RExplorator organizes the workspace into workbenches, each representing a task. A user may save workbenches for later reuse, and share it with other users as well.

The development interface of RExplorator is best suited to allow users to explore RDF repositories, and requires understanding the RDF model. RExplorator allows expert users to provide an end-user friendly interface – called the Application Interface - to solutions found while exploring datasets. This interface is generated by a combination of views, which are defined using the "views" menu option; generic, pre-defined views are initially available for reuse. Views make full use of CSS, which is also defined in a separate view that can be customized to change the look-and-feel of the generated interface. Figure 2 shows an example of an application interface.

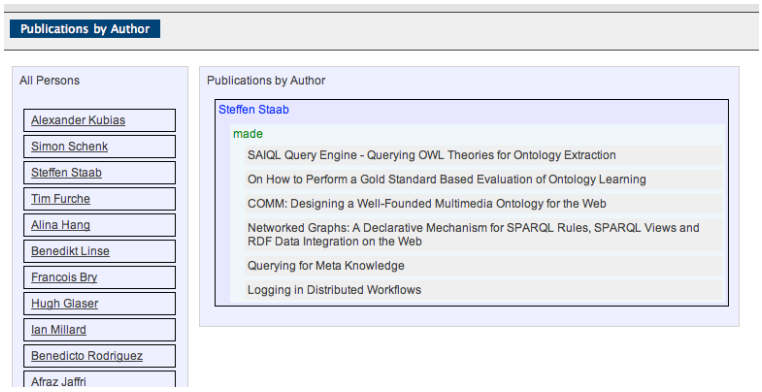


Figure 2 – User-friendly interface with All Persons and Selected Person Co-Workers.

ACKNOWLEDGMENT. Daniel Schwabe was partially supported by a grant from CNPq.

3 REFERENCES

[1] Araújo F. C. S.; Schwabe D.; Explorator: A tool for exploring RDF data through direct manipulation, In: Proceedings of the Linked Data on the Web Workshop (LDOW2009), Madrid, Spain, April 20, 2009, CEUR Workshop Proceedings, ISSN 1613-0073, online http://CEUR-WS.org/Vol-538/ldow2009_paper2.pdf

Generating RDF for Application Testing*

Daniel Blum and Sara Cohen
{daniel.blum@mail,sara@cs}.huji.ac.il

School of Computer Science and Engineering
The Hebrew University of Jerusalem

Abstract. Application testing is a critical component of application development. Testing of Semantic Web applications requires large RDF datasets, conforming to an expected form or schema, and preferably, to an expected data distribution. Finding such datasets often proves impossible, while generating input datasets is often cumbersome. The GRR (Generating Random RDF) system is a convenient, yet powerful, tool for generating random RDF, based on a SPARQL-like syntax. In this poster and demo, we show how large datasets can be easily generated using intuitive commands.

1 Introduction

Testing is a critical step in application development. For Semantic Web applications, testing is a challenge due to both the large volume of input data needed, and the intricate format that this data must have. While many Semantic Web applications focus on varied and unexpected types of data, there are also many others that target specific domains. For such applications, to be useful, datasets used should have at least two properties:

1. The data structure should have the expected structure needed for the target application (e.g., conform to a specific RDF schema).
2. The data should match the expected data distribution of the target application.

Currently, there are several distinct sources for RDF datasets. First, there are *downloadable RDF datasets* that can be found on the web, e.g., Barton libraries, UniProt catalog sequence, and WordNet. RDF Benchmarks, which include both large datasets and sample queries, have also been developed, e.g., the Lehigh University Benchmark (LUBM) [4] (which generates data about universities), the SP^{2}Bench Benchmark [7] (which provides DBLP-style data) and the Berlin SPARQL Benchmark [1] (which is built around an e-commerce use case). Such downloadable RDF datasets are usually an excellent choice when testing the efficiency of an *RDF storage system*. However, they will not be suitable for experimentation and analysis of a particular *RDF application*. Specifically, since these datasets are built for a single given scenario, they may not have either of the two specified properties, for the application at hand.

Data generators are another source for datasets. A data generator is a program that generates data according to user constraints. As such, data generators are usually more flexible than benchmarks. Unfortunately, there are few data generators available for

* This work was partially supported by GIF Grant (2201-1880.6/2008).

RDF (SIMILE [8], RBench [6]) and none of these programs can produce data that conforms to a specific given structure, and thus, again, will not have the specified properties.

In this demo, we present the GRR (Generating Random RDF) system for generating RDF that satisfies both desirable properties given above. Thus, GRR is *not* a benchmark system, but rather, a system to use for Semantic Web application testing. Using intuitive data generation commands with a SPARQL-like syntax, GRR can produce data with a complex graph structure, as well as draw the data values from desirable domains. Data generation commands are translated into a series of SPARQL queries and update commands which are applied directly to an RDF storage system.[1] A video demonstration of GRR is available online,[2] and the system is available upon request.

2 Motivating Example

As a motivating example, we discuss the problem of generating the data described in the LUBM Benchmark. Note that GRR is not limited to creating benchmark data. In our demo, we will demonstrate using GRR to generate other types of data, such as FOAF [3] (Friend of a Friend) datasets, which are used in social network applications.

LUBM [4] is a collection of data describing university classes (i.e., entities), such as departments, faculty members, students, courses, etc. These classes have a plethora of properties (i.e., relations) between them, e.g., faculty members work for departments and head departments, students take courses and are advised by faculty members, etc.

In order to capture a real-world scenario, LUBM defines interdependencies between the entities. For example, the number of students in a department is a function of the number of faculty members. Specifically, LUBM requires there to be a 1:8-14 ratio of faculty members to undergraduate students. As another example, the cardinality of a property may be specified, such as each department must have a single head of department (who must be a full professor). Properties may also be required to satisfy additional constraints, e.g., courses, taught by faculty members, must be pairwise disjoint.

In the next section, we describe the GRR data generation language, and demonstrate commands for producing LUBM benchmark data. Due to space limitations, we do not provide all commands used to reproduce LUBM. However, we note that the number of words needed in all data generation commands (in order to reproduce LUBM), is only about twice as many as used in the intuitive description of LUBM, provided by [4]!

3 Data Generation Commands

Data is generated by a sequence of *data generation commands* (*dg-commands*, for short) $c_1, \ldots, c_n$, when given as input a (possibly empty) RDF dataset R. The first command c_1 is evaluated over R, while each consecutive command c_i is evaluated over the output of the previous command c_{i-1}.

The general syntax of a single dg-command appears below. Note that square brackets are used to denote optional portions, and the "*" indicates a component that can appear any number of times.

[1] The Jena Semantic Web Framework for Java [5] is used in our implementation.

[2] `http://www.cs.huji.ac.il/~danieb12/`

```
(FOR (EACH | sampling-method)
    [WITH (GLOBAL DISTINCT | LOCAL DISTINCT | REPEATABLE)]
    {list of classes}
    [WHERE {list of conditions}] )*
[CREATE i-j {list of classes}]
[CONNECT {list of connections}]
```

A dg-command contains any number of FOR clauses, and then optionally a CREATE and/or CONNECT clause. Intuitively, the FOR clauses choose portions of the RDF input, the CREATE clause creates new nodes in the RDF graph, and the CONNECT clause connects nodes in the RDF graph. We require that at least one among the CREATE and CONNECT clauses be present in every dg-command. We now describe each clause, briefly. (Full language semantics appears in [2]).

- The FOR Clause: Each FOR clause defines *(1)* a query which will applied against the RDF input, as well as *(2)* a method to choose a subset of the query results. For (1), the user provides a list of classes whose instances should be chosen (similar to a SPARQL SELECT clause), as well as any conditions (similar to a SPARQL WHERE clause). The correspondence to SPARQL is not precise as we allow for certain syntactic shortcuts, which avoid explicit variable use, and make dg-commands more readable. For (2), the user defines both the method with which answers should be sampled, as well as whether the sampling process is with/without repetition.
- The CREATE Clause: The CREATE clause defines nodes that should be created. The user provides both a list of RDF classes, and a range determining how many instances of these classes should be created.[3]
- The CONNECT Clause: The CONNECT clause determines the edges that should be generated in the RDF graph, by providing a list of triples.

Several examples of dg-commands appear below. Explanations follow.

```
(c1) CREATE 1-5 {ub:Univ}

(c2) FOR EACH {ub:Univ}
       CREATE 15-25 {ub:Dept}
       CONNECT {ub:Dept ub:subOrg ub:Univ}

(c3) FOR EACH {ub:Faculty, ub:Dept}
     WHERE {ub:Faculty ub:worksFor ub:Dept}
       CREATE 8-14 {ub:Undergrad}
       CONNECT {ub:Undergrad ub:memberOf ub:Dept}

(c4) FOR EACH {ub:Dept}
         FOR 1 {ub:FullProf}
         WHERE {ub:FullProf ub:worksFor ub:Dept}
         CONNECT {ub:FullProf ub:headOf ub:Dept}
```

[3] Dg-commands do not directly define how textual (or other atomic) properties are created and associated with class instances. This information is provided in a simple auxilliary file, e.g., which associates each textual property with a sampling method or dictionary.

```
(c5) FOR 20%-20% {ub:Undergrad, ub:Dept}
     WHERE {ub:Undergrad ub:memberOf ub:Dept}
       FOR 1 {ub:Prof}
       WHERE {ub:Prof ub:memberOf ub:Dept}
       CONNECT {ub:Undergrad ub:advisor ub:Prof}

(c6) FOR EACH {ub:Undergrad}
       FOR 2-4 WITH LOCAL DISTINCT {ub:UndergradCourse}
       CONNECT {ub:Undergrad ub:takeCourse ub:UndergradCourse}

(c7) FOR EACH {foaf:Person ?p1}
       FOR 15-25 {foaf:Person ?p2} WHERE {FILTER( ?p1 != ?p2 )}
       CONNECT {?p1 foaf:knows ?p2}
```

Command c_1 creates between 1 and 5 universities, and command c_2 adds 15–25 departments as suborganizations for each university. Command c_3 iterates over all pairs of faculty members[4] and departments, and adds 8-14 students, per pair to the department (therby achieving the required 1:8-14 ratio of faculty members to undergraduates). Command c_4 chooses one full professor as the head of each department. Command c_5 adds an advisor for 20% of all undergraduates. Command c_6 assigns 2-4 courses for each undergraduate. Note the use of `WITH LOCAL DISTINCT` which ensures that the set of courses chosen per student does not contain repetition, while allowing different students to be assigned the same courses. Finally, c_7 demonstrates advanced features including variables and a filter command, to connect people (in an FOAF RDF dataset) to one another.

In our poster and demo, we will show how to recreate the LUBM benchmark using 24 dg-commands, of the style seen above. In addition, we will show how to create interesting datasets for the FOAF schema. We will also allow those interested to write their own dg-commands, which we will evaluate in GRR to create an RDF dataset.

References

1. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. International Journal of Semantic Web Information Systems 5(2), 1–24 (2009)
2. Blum, D., Cohen, S.: Grr: Generating random RDF. Tech. rep., The Hebrew University of Jerusalem (2010)
3. The friend of a friend (FOAF) project. `http://www.foaf-project.org`
4. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2-3), 158–182 (2005)
5. Jena–a Semantic Web framework for Java. `http://jena.sourceforge.net`
6. RBench website. `http://139.91.183.30:9090/RDF/RBench/index.html`
7. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP^{2}Bench: a SPARQL performance benchmark. In: ICDE. pp. 222–233. Shanghai, China (Mar 2009)
8. Simile website. `http://simile.mit.edu/`

[4] The faculty members were created with an additional dg-command, which was omitted due to lack of space.

Semantic-based Mobile Mashup Platform

Zhipeng Peng[1], Huajun Chen[1], Jinghai Rao[2], Ying Liu[1],
Lei Wang[1], Jian Chen[1],

[1] College of Computer Science, Zhejiang University,
Hangzhou, 310027, China
{pzp, huajunsir}@zju.edu.cn, {islandsolo, blueabysm, crocus.chen}@gmail.com

[2] Nokia Research Center, Beijing Economic and Technological Development Area,
Beijing, 100176, China
{jinghai.rao}@nokia.com

Abstract. Mobile devices contain more personal data such as GPS location, contacts and music, with which users can create innovative and pragmatic mashup applications for different areas such as social networking, E-commerce, and entertainment. We propose a semantic-based mobile mashup platform which enables users to create mashup applications by simply selecting service nodes, linking them together and configuring some connection parameters. Our platform also offers a recommendation mechanism on linkable services by adding semantic annotation to service description, so that users do not need to read specifications of web services in order to find out linkable ones. Therefore, users can focus more on the innovation and practicability of their mashup applications, which will surely result in the emergence of abundant mobile mashup applications.

Keywords: Semantic annotation, mobile mashup, mashup recommendation, web service

1 Introduction

Presently, the number of mobile mashup applications grows rapidly with fast mobile computing development. Current mobile devices' multimedia (e.g. Camera and media playback), sensing (e.g. GPS, Bluetooth, RFID and barcode readers) and communicating (e.g. GPRS, UMTS, Bluetooth, Wi-Fi) capabilities are ideal sentient intermediaries with which users can obtain information for mashup applications. Sentient Graffiti (SG) [1] is one such proposition which offers a platform through which users can contribute content and create integrated mobile applications with mixed information from diverse distributed sources. The TELAR mashup platform [2] is another example which facilitates the creation of adaptive mashups for mobile devices such as the Nokia Internet Tablets. Xu et al. proposed platform architecture based on service oriented architecture (SOA) [3], which focused on how to manage and operate mobile mashup services. Obtaining such mashup applications is not easy because users have to read each service specification and do plenty of programming

work to get what they want. However, most of the mobile mashup users who may have plenty of novel ideas for mobile mashups are not tech-savvy. Our proposition of semantic-based mobile mashup platform meets these users' needs by offering a service recommendation mechanism and a user interface in which users can construct and execute resulting mashups. We choose browser-server architecture so that it will be operating system independent. What users need are simply a mobile device and a flashlite supported web browser. In this paper we present our mobile mashup platform and its semantic-based recommendation mechanism.

2 Semantic-based Mobile Mashup Platform

2.1 Semantic-based Recommendation Mechanism

On our platform, users are able to make a mashup application by selecting a few service nodes and connecting them together. The platform is able to find matched services and the inter-connections among them, which allows users to easily create their applications without knowing every single detail about each service. The recommendation mechanism is implemented by adding semantic annotation to each input or output parameter, so that all services have a common interface. The linkability judgment of two services is shown in Fig. 1.

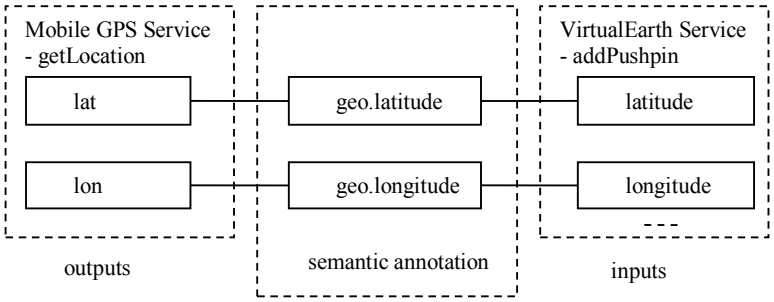


Fig. 1. Since the output parameters ("lat" and "lon") of operation "getLocation" have the same semantic annotations ("geo.latitude" and "geo.longitude") as input parameters ("latitude" and "longitude") of operation "addPushpin", Mobile GPS service is linkable to Virtual Earth service.

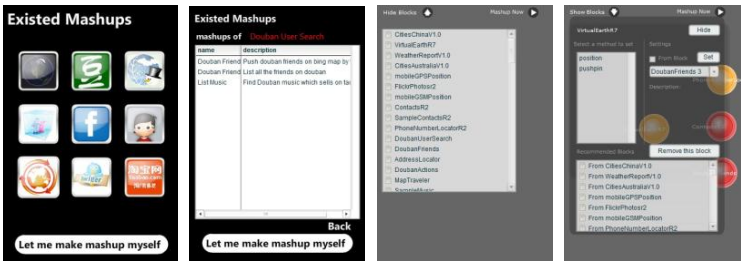
Along with this basic parameter-match recommendation theory, we also include other criteria to filter and rank recommended services. The first criterion is matching degree (the numbers of parameters matched) of two operations. Required input parameters must be met, or the service will be filtered out. After that, with bigger matching degree, the recommended service receives higher rank. The second criterion is conditional probability calculation [4]. The bigger probability that the link of

recommended service and target service is included by any mashup in repository, the higher rank the recommended service gets.

Besides the mashup constructing UI, our platform also includes a mashup execution engine. The execution result is shown in a new page temporarily with a list view or map view. Moreover, advanced users who expect more complicated functions can also encapsulate services by conforming to a specified standard by themselves.

2.2 Mashup Constructing UI

The mashup constructing UI is shown in Fig. 2. On the home screen, users can either execute mashups which are already made by themselves or by others or construct a new one by following a few steps. Firstly, users select a node from the service list. Then they will get a list of recommended services and they can add recommended services to the construction panel. After that, users need to do a little configuration work for connections between service nodes. Lastly, users can execute the resulting mashup, and save it if they are satisfied with the result.



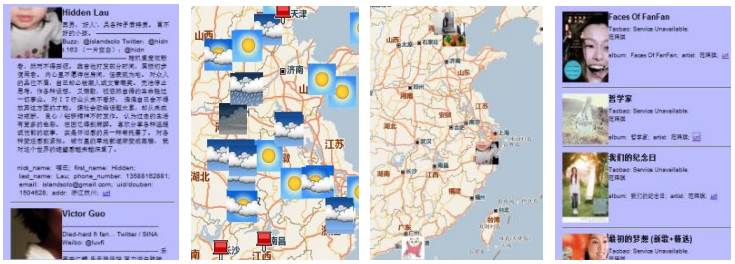
(a) (b) (c) (d)

Fig. 2. User interface of our platform: (a) Some icons represent different classes of mashups, such as location related, or contact related. (b) A list of mashups which are executable. (c) User interface where users can select service nodes to make a mashup. (d) Configuration panel where users can get recommended services and configure links between service nodes.

2.3 Mobile Mashup Examples

With our platform, users can get a variety of innovative and pragmatic mashup applications in different areas such as social networking, E-commerce and entertainment. In Fig. 3 we show several mashups obtained with our platform including a Douban[1] social mashup, a weather checking mashup, a friends' location viewing mashup and a mashup which shows disc information and related purchasing information for music in the mobile[1] device. Users are able to obtain more diverse mobile mashups by selecting linkable services and constructing a mashup graph.

[1] http://www.douban.com/



(a) (b) (c) (d)

Fig. 3. Mobile mashup results: (a) Show Douban information of friends in the contact book. (b) Show weather in major cities. (c) Show friends which are in the contact book on a map. (d) Show disc and purchasing link information of songs in the mobile device.

3 Conclusion

We have presented a mobile mashup platform with a semantic-based recommendation mechanism and a mashup execution engine. The recommendation mechanism makes mashup construction and execution much easier. By following a few simple steps, users are able to obtain a mashup based on their preferences. For future work, we will improve the platform to deal with more complicated situations.

4 Acknowledgments

The authors acknowledge the support provided by CCNT Laboratory of Zhejiang University and Nokia Research Center of Nokia Corporation.

References

1. Brodt, A., Nicklas, D.: The TELAR Mobile Mashup Platform for Nokia Internet Tablets. In: EDBT 2008: Proceedings of the 11th international conference on Extending database technology, 2008
2. Lopez de Ipi˜ na D., Vazquez J. I., and Abaitua J.: A context-aware mobile mash-up plaftorm for ubiquitous web. In: Proc. of 3rd IET Intl. Conf. on Intelligent Environments, pages 116–123, 2007
3. Huiyang Xu, Meina Song, Hui Chen and Junde Song: Research on SOA based mobile mashup platform for telecom networks. The Journal of China Universities of Posts and Telecommunications. Volume 15, Supplement 1, pages 31-36, 2008
4. Huajun Chen, Bin Lu, Yuan Ni, Guotong Xie, Chunying Zhou, Jinhua Mi and Zhaohui Wu: Mashup by Surfing a Web of Data APIs. In: Proc. of the VLDB Endowment. Volume2, Issue 2, pages 1602-1605, 2009

A SILK Graphical UI for Defeasible Reasoning, with a Biology Causal Process Example*

Benjamin Grosof[1], Mark Burstein[2], Mike Dean[2], Carl Andersen[2], Brett Benyo[2], William Ferguson[2], Daniela Inclezan[3], and Richard Shapiro[2]

[1] Vulcan Inc., Seattle, Washington, USA, benjaming@vulcan.com
[2] Raytheon BBN Technologies, Cambridge, Massachusetts, USA, {burstein, mdean canderse, bbenyo, wferguson, rshapiro}@bbn.com
[3] SRI International, Menlo Park, USA, daniela.inclezan@ttu.edu

Abstract. SILK is an expressive Semantic Web rule language and system equipped with scalable reactive higher-order defaults. We present one of its latest novel features: a graphical user interface (GUI) for knowledge entry, query answering, and justification browsing that supports user specification and understanding of advanced courteous prioritized defeasible reasoning. We illustrate the use of the GUI in an example from college-level biology of modeling and reasoning about hierarchically-structured causal processes with interfering multiple causes.

1 Introduction to SILK

SILK[4] (Semantic Inferencing for Large Knowledge) is an expressive Semantic Web rule language and system equipped with scalable reactive higher-order defaults. The system includes capabilities for reasoning, knowledge interchange, and user interface (UI). Part of Project Halo[5], sponsored by Vulcan Inc., the SILK research program addresses fundamental knowledge representation (KR) requirements for scaling the Semantic Web to widely-authored Very Large Knowledge Bases (VLKBs) in business and science that answer questions, proactively supply info, and reason powerfully. The SILK effort has over 15 contributing institutions, including Vulcan, Stony Brook University, Raytheon BBN Technologies, Cycorp, and SRI International.

SILK pushes the frontier of KR by combining expressiveness plus semantics plus scalability. It targets defeasibility, higher-order, and actions — including to support reasoning about complex processes that are described in terms of causality, hierarchical structure, and/or hypothetical scenarios. For example, reasoning about causal processes is a large portion of first-year college biology, often requiring multi-step causal chains and/or multiple grain sizes of description to answer a textbook or exam question. Longer-term, SILK targets widely collaborative KA by subject matter experts (SMEs), such as science students/teachers or business people, not just knowledge engineers (KEs) or programmers.

* This work is part of the SILK project sponsored by Vulcan Inc.
[4] http://silk.semwebcentral.org
[5] http://projecthalo.com

SILK has a new fundamental KR: *hyper* logic programs, which extends normal declarative logic programs (LP). Hyper LP is the first to tightly *combine* several key advanced expressive features: *defaults*, with strong negation and priorities, cf. courteous LP [1] with argumentation theories [2]; (quasi) *higher-order* syntax, reification, and meta-reasoning, cf. HiLog [3] and Common Logic; and procedural attachments to *external* actions (side-effectful), queries (to built-ins, web sources or services), and events (knowledge update flows), cf. situated/production LP [1] (and similar to production rules). KR languages supported for interchange include: SPARQL and RDF(S); SQL and ODBC (e.g., Excel spreadsheets); SILK, RIF (-BLD and -SILK), and OWL (-RL); Cyc (most of its KR and KB); and AURA [4]. AURA is a Project Halo system for question-answering in first-year college science and currently has a KB with tens of thousands of axioms about biology. AURA largely pre-dates SILK and employs a frame-based KR that is considerably less expressive than SILK.

Outline and Contributions: A previous version of SILK was presented in [5]. In the rest of this paper, we present a novel addition to SILK since then: a graphical user interface (GUI) for KA and querying that treats defeasibility.

2 SILK Graphical User Interface & Defeat Justifications

We have developed a graphical user interface (GUI) to the SILK system for knowledge entry, query answering, and justification browsing. The GUI is currently used by KEs and is being extended to support use by subject matter experts (SMEs). The GUI supports user specification and understanding of advanced courteous prioritized defeasible reasoning. It is implemented as a plug-in to the Eclipse Integrated Development Environment (IDE).

The GUI, pictured in Figure 1, offers users a number of capabilities. Entered SILK statements are syntactically validated and statement components (e.g., annotations) are color-coded for clarity. User debugging of rule bases is facilitated by automatic tracking of target queries' results against user changes to the rules. This also allows what-if explorations. The GUI also offers query result justification trees (technically, graphs) that can be explored incrementally, by expanding each tree node to display its children. At each node, the user can specify particular bindings to filter the portion of the justification tree that is displayed. Trees of this sort are also available for negative results (i.e., when a literal *cannot* be inferred), allowing developers to drill down and identify flaws in a desired chain of logical reasoning. This display mechanism also supports the reasoning chains found in courteous defaults by showing *defeated* ground rule instances — rules whose heads are not true, despite their bodies being true, due to conflict with other rules. Figure 1 shows an example of *refutation*-flavor defeat of a rule instance (and thus of its head atom A). The rule instance has a *candidate* argument — i.e., the rule's body is satisfied. But there also is a candidate *counter*argument (whose head is *neg* A) that has a *higher-priority* rule tag.

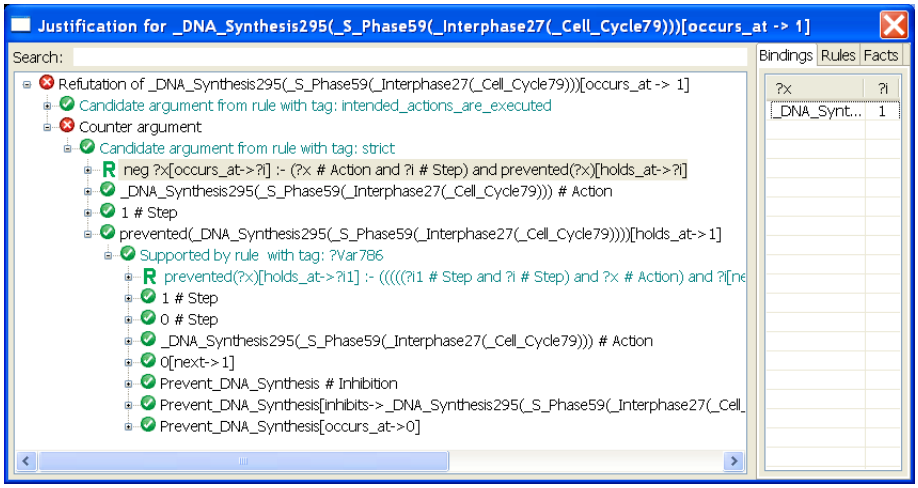


Fig. 1. SILK GUI: exploring the justification of defeat of a biology process rule

In (ASCII) **SILK syntax**, `a # c` means that `a` is an instance of class `c`. Skolems are prefixed by the underscore character ("_"). A courteous rule label term, used for prioritization, is called a *tag*.

Flexible control of selection and layout of items to display is achieved via running rules that are specified (e.g., by power users) in SILK itself.

To our knowledge, the SILK GUI is only the second justification exploration GUI for (prioritized) *defeasible* rules that have (declarative, model-theoretic) *semantics*. The first such system was DR-DEVICE [6], which displays defeat justifications, but with less extensive GUI functionality than SILK provides. Its Defeasible Logic KR is closely related to the courteous feature of hyper LP (see [7] for a comparison), but lacks higher-order and several other advanced expressive features of hyper LP.

3 Example: A Complex Causal Process in Biology

We have developed a novel approach to modeling and reasoning about hierarchically-structured causal processes, that smoothly handles interference/exceptions between multiple causes and elegantly treats the "frame problem" (inertia / persistence of causal fluents). It leverages hyper LP's prioritized defaults. To fully describe the approach is beyond the scope of this paper, however. Instead, we illustrate the approach with an example of college-level biology that shows the SILK GUI's novel capability to explore justifications in the presence of prioritized defeat.

In biology and medicine, a key process is the cell cycle in which a cell grows and then divides. (Control failure in this process causes cancer.) The cell cycle is a complex hierarchically-structured process. It consists of two phases (subprocesses): interphase and mitosis, in that temporal order. Interphase, in turn,

consists of three subphases G1, S, and G2, in that order. Mitosis too has several subphases. Many of the above subphases in turn have sub-subphases, etc. DNA synthesis occurs during S phase, and indeed begins when S phase begins. This is the knowledge required to answer the following first-year college exam question[6]:

```
A researcher treats cells with a chemical that prevents
DNA synthesis from starting.  This treatment traps the cells
in which part of the cell cycle?
  Correct answer: G1.
```

That is, if DNA synthesis does not occur, then S Phase does not occur, and the cell cycle stops in the preceding phase which is G1.

Figure 1 shows a key intermediate step in SILK's inferencing — defeat of a candidate argument that: DNA Synthesis will occur for the question's focal cell cycle (`_Cell_Cycle79`), as is normal (i.e., "intended") in the cell cycle's process's cascade of causal phase steps. That argument is refuted because there is a higher-priority counter argument (itself undefeated) based on the preventive/inhibitory causal effect of the hypothetical scenario's chemical treatment.

4 Conclusions

A key direction in current and future SILK work is to increase SME friendliness of the UI in collaborative KA and querying, using in part controlled natural language. SILK is now being integrated with other portions of Project Halo, particularly AURA. We are refining a translation of Cyc biology etc. knowledge to SILK. See the SILK website for an **extended version of this paper**.

Acknowledgements: Thanks to the SILK team, esp. Paul V. Haley, Terrance Swift, Michael Kifer, David Gunning, Vinay Chaudhri, Michael Gelfond.

References

1. Grosof, B.N.: Representing E-Commerce Rules via Situated Courteous Logic Programs in RuleML. Electronic Commerce Research and Applications **3**(1) (2004)
2. Wan, H., Grosof, B., Kifer, M., et al.: Logic Programming with Defaults and Argumentation Theories. In: Proc. 25th Intl. Conference on Logic Programming. (2009)
3. Chen, W., Kifer, M., Warren, D.: HiLog: A foundation for higher-order logic programming. Journal of Logic Programming **15**(3) (February 1993) 187–230
4. Gunning, D., Chaudhri, V.K., et al.: Project Halo Update: Progress Toward Digital Aristotle. AI Magazine to appear, Fall 2010.
5. Grosof, B.N., Dean, M., Kifer, M.: The SILK System: Scalable and Expressive Semantic Rules. In: Posters and Demos, 8th Intl. Semantic Web Conf. (2009)
6. Bassiliades, N., et al.: Proof explanation in the DR-DEVICE system. In: Proc. 1st Intl. Conf. on Web Reasoning and Rule Systems. (2007)
7. Wan, H., Kifer, M., Grosof, B.: Defeasibility in Answer Set Programs via Argumentation Theories. In: Proc. 4th Intl. Conf. on Web Reasoning and Rule Systems. (2010)
8. Campbell, N.A., Reece, J.B.: Biology. 6th edn. Benjamin Cummings (2002)

[6] chapter 12 self-quiz question 15 in [8]

Semantics for music researchers: How country is my country?

Kevin R. Page [1,2], Benjamin Fields [3], Bart J. Nagel [2], Gianni O'Neill [2], David C. De Roure [1], Tim Crawford [3]

[1] Oxford e-Research Centre, University of Oxford, UK
[2] School of Electronics and Computer Science, University of Southampton, UK
[3] Department of Computing, Goldsmiths University of London, UK

Abstract The Linking Open Data cloud contains several music related datasets that hold great potential for enhancing the process of research in the field of Music Information Retrieval (MIR) and which, in turn, can be enriched by MIR results.
We demonstrate a system with several related aims: to enable MIR researchers to utilise these datasets through incorporation in their research systems and workflows; to publish MIR research output on the Semantic Web linked to existing datasets (thereby also increasing the size and applicability of the datasets for use in MIR); and to present MIR research output, with cross-referencing to other linked data sources, for manipulation and evaluation by researchers and re-use within the wider Semantic Web.
By way of example we gather and publish RDF describing signal collections derived from the country of an artist. Genre analysis over these collections and integration of collection and result metadata enables us to ask: "how country is my country?".

1 Background and motivation

Much of the work of researchers in the field of Music Information Retrieval (MIR) focusses on the algorithmic extraction of information from music. However, there are many problems associated with the design and implementation of distributed systems within which such algorithms might be deployed.

We can broadly describe the process an MIR researcher typically follows in three steps; we also highlight some of the issues and, at an abstract level, how linked data and semantic web technologies might assist in building a complete system.

1. **Assemble a collection of audio input.** To evaluate an algorithm, the researcher must acquire a wide selection of "signal" – typically digital audio files – for the algorithm to process. Music recordings are often restricted from free exchange amongst researchers, either explicitly through copyrights or implicitly through the high overheads of managing detailed and intricate licensing. Even when audio data is freely available and distributable

a difficult balance must be found to avoid "over-fitting" of algorithms to a particular set of signals: whilst a widely shared, understood, and re-usable collection is critical for comparative evaluation, tuning an algorithm to such a collection during development (knowing it will be the benchmark) is likely to detrimentally affect performance against more randomly selected input (i.e. real-world tests). It is therefore useful to create and modify large collections of audio data quickly and flexibly which can be shared between researchers for comparative evaluation. Restrictions on the distribution of actual audio files can be accommodated through the separate description of collections and correctly modelling the relationship between artefacts (e.g. distinguishing between a work, a performances of the work, recordings of the performance, and published media of the recording); metadata exchange can then occur independently and be cross-referenced against any institutional or other private archive of audio. Linking existing metadata for audio files and basing collection generation on this information is desirable for quickly trialling an algorithm against particular musical facets (e.g. a particular period and style derived from the composers).

2. **Apply the algorithm to the audio input.** There are many MIR systems which enable an algorithm to be applied to signal. More recently some systems have begun to adopt practices and tools from the scientific workflow community, for example the Meandre workflow enactment system [1]. Any such system must be able to recognise an input collection and apply the algorithm across it. Where institutionally restricted collections of signal are in use a system must match local audio files to any abstract, metadata based, collection descriptions.

3. **Publish and evaluate algorithm output.** The MIR community has a 7 year history of comparative evaluation in the MIREX competition; the most recent (2010) MIREX adopted a Meandre derived framework for executing the algorithms under test [2]. More generally, evaluation of results requires a common structure into which analytic output can be published for comparison, rather than data structures inherited from the development tool or environment a researcher was using. As faster computational resources become more readily available and can be applied to MIR tasks, the opportunity to undertake analysis on an ever greater scale brings with it the associated problems of managing ever greater quantities of result data. Links from results back to recorded signal (and audio file artefacts) and capturing provenance are equally important: an single algorithm is not normally sufficient to make a definitive assertion, e.g. to classify a recording as jazz. For this reason it is important that the representation of results can be used as input for creating derivative collections of input for further MIR analysis such that information extracted from multiple algorithms can be combined and refined.

2 System overview and "Country/country" example

Employing new RDF encodings for collections and results that utilise existing ontologies (including the Music Ontology, GeoNames, Provenance Vocabulary, and OAI-ORE), and by deploying a linked data audio file repository and services for publishing collections and results, we present a proof-of-concept system that addresses the problems outlined in the previous section. While the principles and design described here can be applied to all MIR systems, for demonstration purposes we have developed a specific use case known as "Country/country". In this section we outline the components of the system, which approximately align to the steps in the previous section (with the addition of a pre-step), detailing the generic purpose of each service, followed by the specific implementation in Country/country (*in italics*).

0. An Audio File Repository which serves audio files and linked data about the audio files using HTTP. *For our public demonstrator a subset copy of the free-licensed Jamendo collection*[1] *has been used.* Using the Music Ontology[3], the relationship to the track it is a recording of, and the "definitive" URI for that track (*as minted by the Jamendo linked data service at dbtune*[2]) is asserted in the linked data.

1. A Collection Builder web application that enables a user to publish sets of tracks described using RDF. The backend uses SPARQL to build collections and takes advantage of links between datasets: *e.g. the Jamendo service incorporates links to geographic locations as defined by GeoNames*[3]*, so the Collection Builder can identify all the tracks offered by Jamendo recorded by artists from a specific country.* An optional second stage of collection builder takes a collection and "grounds" the constituent tracks against available recordings of those tracks by posing SPARQL queries to Audio File Repositories. *In the case of Country/country we "ground" a country derived collection against our Audio File Repository of locally available signal.*

2. The Analysis is performed by a NEMA[2] *genre classification* workflow:

- We have extended the myExperiment[4] scientific collaborative environment to support the Meandre[1] workflows used by NEMA.
- myExperiment has also been modified to accept the collections RDF published in step 1) and marshal the target tracks contained within to the analysis workflow.
- Within the (Meandre-based) *genre classification* workflow a head-end component has been written to dereference each track URI passed to the workflow and, using the linked data published by the signal repository, retrieve both the local copy of the audio file and the reference to the *original Jamendo identifier*. This URI persists through the genre analysis workflow until it reaches a new tail-end component where the analysis is published using RDF – including links back to the *Jamendo* URI.

[1] *http://www.jamendo.com/*
[2] *http://dbtune.org/jamendo/*
[3] *http://www.geonames.org/ontology/*

3. **A Results Viewer** web application retrieves the collections RDF from 1) and results RDF from 2), cross-referencing them via the URIs used throughout the system. The user can identify trends in genre classification within and between collections. Results can be pooled and compared using existing and new collections and inform the creation of new sets. To demonstrate how further links can easily be made to existing datasets and inform derivative collection generation, relevant associations from other linked data sets are shown (*e.g. artists of the same genre and country from DBpedia and the BBC for a particular analysed track*).

3 Online demonstrator

The Country/country demonstrator system is available at:
`http://www.nema.ecs.soton.ac.uk/countrycountry/`

Acknowledgements

Many thanks to Stephen Downie and his team at IMIRSEL University of Illinois for access and source code to their "Son of Blinkie" genre classification workflow and the myExperiment team at the University of Southampton for help and guidance while developing the Meandre extension. This work was carried out through the Networked Environment for Musical Analysis (NEMA) project funded by the Andrew W. Mellon Foundation, and the Structural Analysis of Large Amounts of Musical Information (SALAMI) project funded by the JISC Digitisation and e-Content programme as a part of the Digging into Data challenge.

References

1. X. Llorà, B. Ács, L. Auvil, B. Capitanu, M. Welge, and D. Goldberg, "Meandre: Semantic-Driven Data-Intensive Flows in the Clouds," in *4th IEEE International Conference on eScience*, pp. 238–245, Dec. 2008.
2. K. West, A. Kumar, A. Shirk, G. Zhu, J. Downie, A. Ehmann, and M. Bay, "The Networked Environment for Music Analysis (NEMA)," in *6th IEEE World Congress on Services*, pp. 314–317, July 2010.
3. Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson, "The music ontology," in *Proceedings of the International Conference on Music Information Retrieval*, pp. 417–422, 2007.
4. C. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, *et al.*, "myExperiment: a repository and social network for the sharing of bioinformatics workflows," *Nucleic Acids Research*, 2010.

RDOTE - Transforming Relational Databases into Semantic Web Data

Konstantinos N. Vavliakis[1,2], Theofanis K. Grollios[1], and Pericles A. Mitkas[1,2]

[1]Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, GR541 24, Thessaloniki, Greece
[2]Informatics and Telematics Institute, CERTH, GR570 01, Thessaloniki, Greece
kvavliak@issel.ee.auth.gr,fgroll@auth.gr,mitkas@eng.auth.gr

Abstract. During the last decade, there has been intense research and development in creating methodologies and tools able to map Relational Databases with the Resource Description Framework. Although some systems have gained wider acceptance in the Semantic Web community, they either require users to learn a declarative language for encoding mappings, or have limited expressivity.
Thereupon we present *RDOTE*, a framework for easily transporting data residing in Relational Databases into the Semantic Web. *RDOTE* is available under GNU/GPL license and provides friendly graphical interfaces, as well as enough expressivity for creating custom RDF dumps.

Keywords: Relational Databases to Ontology Transformation, RDB2RDF, RDF Dump

1 Introduction

The large volume of data residing in relational databases led to the creation of systems for instantiating ontology schemata using relational information, with some, like D2RQ, gaining wider acceptance in the Semantic Web community. Unfortunately, all these tools require advanced user skills as they are either built upon complex declarative languages and lack friendly user interfaces or they provide GUIs with limited expressivity.

We present *RDOTE*, a system able to map multiple Relational Databases (RDB) into different ontology schemata and integrate them into a single ontology file. *RDOTE* is online[1,2] available under the GNU/GPL license and provides drag 'n drop operations, drop down lists and recommendation mechanisms, that allow users to define all the necessary mappings between tables/columns and classes/properties, in order to create domain-specific mappings according to a selected ontology schema.

The main contribution of *RDOTE* towards Semantic Web researchers is twofold: a) it can transform datasets currently residing in (one or many) Relational

[1] http://sourceforge.net/projects/rdote/
[2] http://www.youtube.com/watch?v=pk7izhFeuf0

Databases into Semantic Web data through a friendly interface and b) it can quickly instantiate an ontology schema with real data, allowing easy experimentation with large ontology datasets.

2 Background Information - Relevant Work

Throughout related bibliography, one may find numerous methodologies and systems for publishing data residing in Relational Databases into the Semantic Web. D2RQ [1] is the mostly embraced by the Semantic Web Community. D2RQ offers a powerful declarative language for mapping Relational Databases to ontologies, nevertheless, no graphical user interfaces are provided. Less prevalent systems, like RDB2Onto [4] are highly configurable too but they also lack friendly user interfaces. On the other hand, systems like SquirrelRDF [6] offer a simplistic approach to publish RDF data from Relational Databases (still absent GUI), which may not be expressive enough in case of complex databases/mappings. Dartgrid [2] and ODEMapster plugin for the NeOn Toolkit [5] currently offer a graphical user interface, but they have limited scope and applicability, as well as limited expressivity compared to *RDOTE*. Finally Virtuoso RDF View [3] comes with a graphical user interface, which is available only for the Virtuoso database, rather than other popular relational DBMS.

3 *RDOTE* Functionality

An overview of *RDOTE*'s functionality is depicted in Figure 1. *RDOTE* lies in the category of Domain Semantics-driven Mapping Generation tools, all mappings are formulated via graphical user interfaces and stored in text files. *RDOTE* has a generalized application domain and is currently applied on two test cases: one in the bibliographic domain and one in the art object documentation domain, available online.

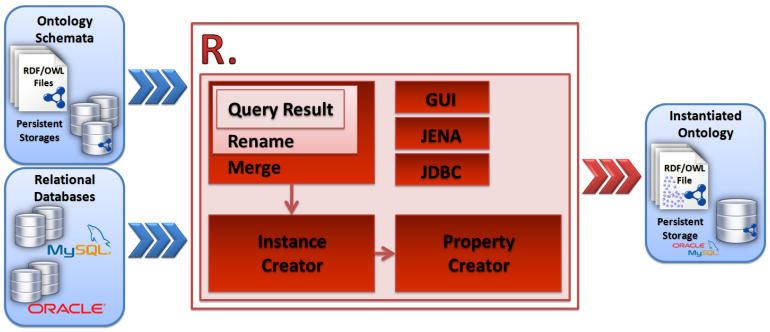


Fig. 1. *RDOTE* Architecture

For the complete transformation of a RDB to an ontology, one has to take the following steps:

1. Connect to the desired RDBMS and load the respective ontology schema: Users first have to connect to the desired RDBs (MySQL and Oracle supported) and load an ontology schema containing the TBox of the desired domain. *RDOTE* can load ontology files in various formats (RDF/XML, N3, N-triples), and/or persistent ontologies. *RDOTE*'s GUI depicts the TBox in a tree representation form, as well as the ABox of the loaded ontology. *RDOTE* also presents the tables contained in each connected RDB and the respective columns.
2. Write SQL queries that select the desired tuples to be processed as an RDF graph: Each query represents a result set that is used to populate an Ontology Class or acts as a dataset of literals for linking instances of a class with a datatype property. For unambiguous creation of instances, the primary key is required in the URI, whereas the user selected columns are required for populating datatype properties. Thus, along with the user defined SQL query, *RDOTE* automatically selects the primary keys that have been defined for the tables participating in the SQL query.
3. Define renaming options and merging strings in case there are queries containing multiple selected columns: This optional step is responsible for renaming result sets, based on regular expression pattern matching. Each tuple matching the defined regular expression will be renamed when used as a literal. Moreover, this step allows users to define merging strings for SQL queries that select data from multiple columns.
4. Connect queries defined in Step 2 with ontology classes (Class Mappings): Next the Class Mappings that are responsible for the creation of all the ontology instances have to be defined. SQL queries are connected to ontology classes and in this way for each tuple of a SQL query, an instance of the respective class is created. In case one wishes to use the actual data instead of just creating URIs, *RDOTE* provides the possibility of copying the actual tuple information into any datatype property of the ontology schema.
5. Define conditional links of Class Mappings with other Class Mappings via object properties or, in case of datatype properties, with SQL queries: For each Property Mapping, users can select a Class Mapping, drag 'n drop a property from the ontology tree and in case of an object property, select a second Class Mapping, whereas in the case of datatype property, select a SQL query. Next a join condition between the first Class Mapping and the second Class Mapping/SQL Query has to be defined, either by the user or *RDOTE*, which can propose possible join conditions as calculated by a greedy graph algorithm traversing the SQL schema. Users can then insert any other conditional restriction SQL92 supports.
6. Instantiate the ontology schema and store it either in text file format or in a persistent repository: Finally users can launch *RDOTE*'s engine, which instantiates the loaded ontology schema and creates an RDF dump of the selected relational data which are stored either in text file format (RDF/XML,

N3, N-triples) or in a persistent storage format (MySQL and Oracle). *RDOTE* first creates all the instances and then it links them with object properties or adds literals using datatype properties.

All steps previously described are realized through *RDOTE*'s friendly interfaces (screenshots are available in *RDOTE*'s homepage), while at any step in the mapping definition process, users can save their project, that is all their mappings and database/ontology connections, and resume later.

The maximum supported RDF dump size *RDOTE* can create depends on the output format and naturally on the numbers of triples. In the case of text format, this size is also significantly dependent on Java maximum heap size. In our case, for maximum Java heap size of 1024MB, *RDOTE* successfully created 2 million triples in RDF/XML-ABBREV format in less than five minutes. Memory limitations do not exist in the case of persistent storage, but time limitations begin to emerge. In this case, *RDOTE* managed to create 10 million triples in less than five hours.

4 Conclusions - Future Work

RDOTE provides the Semantic Web research community and domain experts with the necessary means for easily enriching Ontology schemata with the vast amount of data currently residing in relational databases. It also enables quick instantiations of new ontology schemata for testing and experimentation. By allowing easy transportation of legacy data into semantically aware data structures, *RDOTE* aspires to bring the Semantic Web vision one step closer.

RDOTE is constantly updated. In the near future we expect to incorporate an export/import mechanism for D2RQ compliant mapping files, as well as a query builder graphical user interface together with complementary interfaces and mechanisms that will facilitate and hasten the mapping creation process even further. Finally, further evaluation and testing on large datasets is pending.

References

1. Bizer, C., Seaborne, A.: D2rq - treating non-rdf databases as virtual rdf graphs. In: Poster presented in Internatiotal Semantic Web Conference 2004 (November 2004)
2. Chen, H., Wu, Z.: Dartgrid iii: A semantic grid toolkit for data integration. In: First International Conference on Semantics, Knowledge and Grid. p. 12. IEEE Computer Society (2005)
3. Erling, O., Mikhailov, I.: Rdf support in the virtuoso dbms. In: Conference on Social Semantic Web. LNI, vol. 113, pp. 59–68. GI (2007)
4. Laclavk, M.: Rdb2onto: Relational database data to ontology individual mapping in: Tools for acquisition, organisation and presenting of information and knowledge. Tech. rep. (2008)
5. Rodriguez, J.B., Gómez-Pérez, A.: Upgrading relational legacy data to the semantic web. In: WWW '06: Proceedings of the 15th international conference on World Wide Web. pp. 1069–1070. ACM, New York, NY, USA (2006)
6. Steer, D.: Squirrelrdf. Tech. rep., HP (2006)

WSML2Reasoner - A Comprehensive Reasoning Framework for the Semantic Web

Reto Krummenacher, Daniel Winkler, and Adrian Marte

Semantic Technology Institute (STI), University of Innsbruck, Austria
firstname.lastname@sti2.at

1 Introduction

The amount of data on the Internet is rapidly growing. Formal languages are used to annotate such data in order to make it machine-understandable; i.e., allow machines to reason about it, to check consistency, to answer queries, or to infer new facts. Essential for this are formalisms that allow for tractable and efficient reasoning algorithms. Particular care is demanded in efficiently responding to the trade-off between expressivity and usefulness.

The updated Web Ontology Language (OWL 2) provides dialects that are restricted in their semantic expressivity for optimizing the reasoning behavior; e.g., the OWL 2 EL or OWL 2 RL profiles. Such dialects are very important to respond to the aforementioned trade-off. Profiles reflect particular requirements and yield purposeful balance between expressivity and computational complexity. The support for dialects is not only given in OWL 2, but also in the Rule Interchange Format (RIF) standards. RIF specifies formalisms for the knowledge exchange between different rule systems. The same applies for the WSML language that provides variants for Description Logics and rule-based reasoning. The goal remains the same, formalisms that are expressive enough to be useful, while exhibiting reasoning characteristics that can scale to the size of the Web. Leveraging this is exactly the objective of the WSML2Reasoner framework.

In Section 2 we present WSML2Reasoner and our reasoners IRIS and Elly. We show how the Datalog engine IRIS is used as reasoner for RIF-BLD, and how the ELP reasoner Elly supports the OWL 2 EL and RL profiles. In Section 3 we provide a short example of what shall be shown, amongst other things, during the demo session, and we conclude with Section 4.[1]

2 Reasoners for the Semantic Web

The WSML2Reasoner framework serves as entry point for all the supported OWL, RIF and WSML reasoning.[2] It is based on a highly modular architecture and combines validation, normalization and transformation algorithms for working with ontology descriptions in WSML. The framework includes two default reasoning engines termed IRIS and Elly, and two libraries, namely RIF4J and

[1] This work is supported by the EU FP7 IPs SOA4All and LarKC.

[2] http://tools.sti2.at/wsml2reasoner/

WSMO4J,[3] that provide the object models for RIF-BLD and WSML, respectively. The third-party OWL API yields the data model for the manipulation of OWL ontologies.[4] It adds the reasoner interface that is implemented by Elly for supporting OWL 2 EL and RL. Figure 1 depicts the relevant software components of the WSML2Reasoner framework.

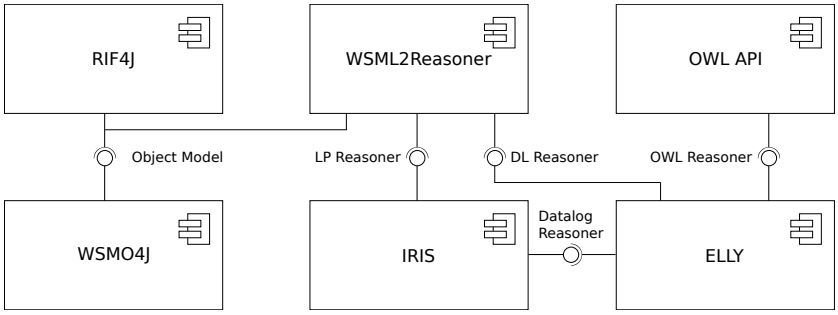


Fig. 1. Reasoner architecture and data model components

2.1 WSML2Reasoner and WSML

WSML is a formal language for the specification of ontologies and the description of Semantic Web services [2]. The latest version WSMLv2.0 provides an alignment of the Logic Programming-based variants WSML-Rule/Flight with RIF, and an updated semantics for the WSML-DL dialect.

Although WSML2Reasoner is designed to support various reasoners, the default release is shipped with IRIS (Section 2.2) and Elly (Section 2.3). These reasoners offer the most complete support for the semantics of WSML, and include the built-ins defined by RIF-DTB. All together, WSML2Reasoner provides a comprehensive reasoning infrastructure for the WSML language family.

2.2 IRIS and RIF Dialects

The Datalog engine IRIS provides the core for both WSML2Reasoner and Elly.[5] In fact, IRIS was initially developed with the WSML stack in mind. The integration of RIF4J and WSML2Reasoner — including the translation modules from RIF rule bases to WSML logical expressions — on top of IRIS realizes the targeted RIF reasoner.

The RIF Datatypes and Built-Ins document specifies a list of datatypes, built-in functions and built-in predicates that are expected to be supported by all RIF dialects [7]. IRIS has been updated to support the full range of built-ins with the exception of the list-related ones.

[3] `http://rif4j.sourceforge.net/`; `http://wsmo4j.sourceforge.net/`
[4] `http://owlapi.sourceforge.net`
[5] `http://www.iris-reasoner.org/`

The expressivity of RIF Core corresponds to Datalog, and due to the very nature of IRIS, RIF Core is fully captured. The RIF-BLD profile matches, in terms of expressivity, the language of definite Horn rules with equality and a standard first-order semantics [1]. IRIS was extended to support the full range of language constructs in RIF-BLD, including equality in rule conclusions. RIF Core and RIF-BLD are at the basis of the W3C recommendation on how to interpret combinations of RIF documents and RDF data, as well as RDFS and OWL ontologies [3]. Consequently, IRIS fulfills the main prerequisites for serving as fully-fledged rule-based reasoner for the (Semantic) Web.

2.3 ELLY and the OWL 2 EL and RL Profiles

ELP is a hybrid between Logic Programming and Description Logics (DL) that combines the tractable DLs $\mathcal{EL}^{++}$ and DLP [4]. These two formalisms yield the logical foundation for the OWL 2 profiles EL and RL, which are thus fully captured by the semantics of ELP [5]. Since ELP does not only define the semantics of the language, but also a tractable reasoning algorithm that translates ELP rule bases into Datalog, a corresponding extension to IRIS could be implemented.

ELLY is a reasoner for entailment and satisfiability checking of ELP rule bases.[6] It includes an object model for ELP and a reasoner based on the translation to Datalog [8]; as such, ELLY is implemented on top of IRIS. As ELP subsumes the semantics of OWL 2 EL and RL, ELLY, in integration with the parsers, object models and reasoning interfaces of the OWL API, becomes a fully-fledged OWL 2 EL and RL reasoner.[7]

3 RIF-BLD Application Scenario

To illustrate how the framework can be leveraged to reason with RIF-BLD rules, we present a scenario based on one of the use cases discussed in [6]. The aim of the chosen example "Publishing Rules for Interlinked Metadata" is to enrich Semantic Web data by application of RIF encoded rules. Table 1 extends the scenario, such that it uses movie metadata that is published on DBPedia[8] and combines it with RIF rules to capture implicit knowledge; e.g., categorizing black and white (B/W) movies depending on their release date.

The RIF-BLD reasoner can then be used for entailment checking against or querying over the specified rule base. For the purpose of this demo, there is a user interface made public at `http://iris.sti2.at/reasoners/rif-reasoner/`. For the modeled scenario, the reasoner returns a variable binding to the movies "Primer" and "The Gold Rush" when querying for low-budget movies (`?- ?Movie#ex:LowBudgetMovie`); the latter is also computed to be a B/W movie. Note that the example uses an abridged RIF presentation syntax and omits namespace declarations; the reasoner solely supports RIF-BLD XML Serialization Syntax, a corresponding example is linked from the Web interface.

[6] `http://elly.sourceforge.net/`
[7] ELLY is listed on `http://www.w3.org/2007/OWL/wiki/Implementations`.
[8] see `http://dbpedia.org`

Table 1. Demo Example: Rules for Interlinked Metadata

```
?Movie#ex:BlackWhiteMovie :-
  ?Movie#dbo:Film
  ?Movie[dbp:released -> ?Date]
  External(pred:date-less-than(?Date "1930-01-01"^^xs:date))
?Movie#ex:LowBudgetMovie :-
  ?Movie#dbo:Film
  ?Movie[dbp:budget -> ?Budget]
  External(pred:numeric-less-than(?Budget "5000000"^^xs:float))
ex:pr#dbo:Film ex:pr[
  rdfs:label -> "Primer"^^xs:string
  dbp:released -> "2004-10-08"^^xs:date
  dbp:budget -> "7000.0"^^xs:float]
ex:gr#dbo:Film ex:gr[
  rdfs:label -> "The Gold Rush"^^xs:string
  dbp:released -> "1925-06-26"^^xs:date
  dbp:budget -> "923000.0"^^xs:float]
```

4 Conclusions

WSML2Reasoner, together with the other presented software components, evolved to a comprehensive reasoning framework for the (Semantic) Web. Emphasized in this respect is a strict conformance to existing Web standards, such as RIF, OWL and in our context WSML too.

With this demonstration, we present the current status of WSML2Reasoner, and emphasize on the application of IRIS as RIF-BLD reasoner; not excluding other examples and online demonstrators for Elly and WSML2Reasoner:

- WSML-DL v2.0, `http://iris.sti2.at/reasoners/wsml-dl-reasoner/`
- WSML-Rule v2.0, `http://iris.sti2.at/reasoners/wsml-rule-reasoner/`
- Datalog, `http://www.iris-reasoner.org/demo`

References

1. Boley, H., Kifer, M.: RIF Basic Logic Dialect. W3C Recommendation (2010)
2. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The Web Service Modeling Language WSML: An Overview. In: 3rd Eur. Semantic Web Conf. pp. 590–604 (2006)
3. de Bruijn, J.: RIF RDF and OWL Compatibility. W3C Recommendation (2010)
4. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable Rules for OWL 2. In: 7th Int'l Semantic Web Conf. pp. 649–664 (2008)
5. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation (2009)
6. Paschke, A., Hirtle, D., Ginsberg, A., Patranjan, P.L., McCabe, F.: RIF Use Cases and Requirements. W3C Working Draft (2008)
7. Polleres, A., Boley, H., Kifer, M.: RIF Datatypes and Built-Ins 1.0. W3C Recommendation (2010)
8. Winkler, D.: ELLY - An ELP Reasoner. Master Thesis, Semantic Technology Institute (STI) Innsbruck, University of Innsbruck (2010)

Linked data from your pocket: The Android RDFContentProvider

Jérôme David, Jérôme Euzenat

INRIA & LIG
Grenoble, France
{Jerome.David,Jerome.Euzenat}@inrialpes.fr

Smartphones are becoming main personal information repositories. Unfortunately this information is stored in independent silos managed by applications. We have seen that already: in the Palm operating system, application "databases" were only accessible when the application schemas were known and worked by opening other application databases.

Our goal is to provide support for applications to deliver their data in RDF. This would allow applications to exploit this information in a uniform way without knowing beforehand application schemas. This would also connect this information to the semantic web and the web of data through reference from and to device information. We present a way to do this in a uniform manner within the Android platform. Moreover, we propose to do it along the linked data principles (provide RDF, describe in ontologies, use URIs, link other sources).

We first consider how the integration of RDF could be further pushed within the context of the Android platform. We demonstrate its feasibility through a linked data browser that allows for browsing the phone information.

1 Providing RDF support in Android

Android is built around different kinds of services, one of which being `ContentProvider` which exposes some data of an application to other applications. `ContentProvider`s manage data structures (usually relational tables) and are able to answer messages of type query, insert, delete and update. A query returns a cursor on a table of tuples. So they offer a typical database interface:

```
Cursor query( Uri id, String[] proj, String select, String[] selectArgs, String order )
Uri insert( Uri id, ContentValues colValueList)
int update( Uri id, ContentValues colValueList, String select, String[] selectArgs )
int delete( Uri id, String select, String[] selectArgs )
String getType( Uri id )
```

This interface uses URIs for identifying the kind of data which is requested: content://contacts/people/ identifies all people in the contact application and content://contacts/people/22 identifies the 22nd instance of these[1].

Calling a `ContentProvider` is driven by the kind of content to be manipulated: the calling application indicates the will to retrieve some content through its type, but does not control which application will provide it. Android calls a `ContentResolver` which further looks into the query (the `id`) to find a suitable content provider on the

[1] Android URIs are not particularly portable, we leave the discussion of this out of this paper.

phone which is able to provide the required content. For that purpose, the resolver maps the query URIs to the declared providers. These providers are declared in application manifests.

In order to exchange RDF within the Android platform, we need `ContentProvider`s providing RDF. For that purpose we have developed a new abstract `RDFContentProvider` extending `ContentProvider`. Answers to queries in an `RDFContentProvider` could be:

- set of triples, which would correspond to the description of one object and the attribute values, this is restricted to queries like: tell me what you know about a particular individual;
- table of tuples, like in `ContentProvider`s or SPARQL which would correspond to values of variable in a SPARQL-like query

These interfaces can be unified since, the former is the answer to the SPARQL query:

```
SELECT ?p ?o WHERE content://contacts/people/22 ?p ?o.
```

The `RDFContentProvider` follows primarily the same kind of interface as `ContentProvider`. The minimal interface for linked data applications is:

- `RDFCursor getRdf( Uri id )`

The Cursor iterates on a table of subject-object-predicate (or object-predicate) which are the triples involving the object given as a URI. A more elaborate semantic web interface could be that of a minimal SPARQL endpoint:

- `Uri[] getTypes( Uri id )`: returns the RDF types of a local URI;
- `Uri[] getOntologies()`: ontologies used by the applications;
- `Uri[] getQueryEntities()`: classes and relation that the application can deliver;
- `Cursor query( SparqlQuery query )`: returns results tuple;
- `Cursor getQueries()`: triple patterns that the application can answer.

So far, we have only developed this interface but the three first primitives.

2 Demonstration: linked data browser

We have implemented a prototype of this architecture described in Figure 1. Precisely, we have implemented:

- `RDFContentProvider`: the provider interface;
- `RDFContentResolver`: which can decide to which class to redirect a query;
- `Pikoid`: a picture annotation application which implements `RDFContentProvider`;
- `AndroidRDFProvider`: an application encapsulating data access to the applications of the Android platform (Calendar, Contact, Map, etc.);
- `RDFBrowser`: a simple client for navigating within RDF data provided by these applications in a generic manner.

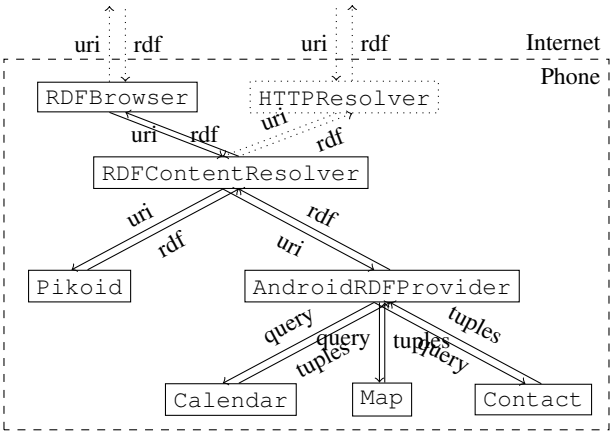


Fig. 1. The four implemented components and the communication between them. Communication with the web (dotted) is not implemented at submission time.

In this demonstration, we will show how to quickly annotate a picture with the help of the standard Android applications (Pikoid), then we will use the RDFBrowser to navigate through these annotations provided as interrelated RDF statements.

The beauty of linked data is that it is easy to understand how to navigate within this data through HTTP requests: each request (a URI) returns RDF from which URIs can be extracted for formulating further requests. The RDF browser acts as a linked data client except that it works over the Android content provider mechanism instead of HTTP: it asks triples about a particular URI, displays these and when clicked on, issue the same URI query. Figure 2 shows the current interface.

3 Conclusion

What we have to demonstrate is only a proof of concept that semantic technologies could be included uniformly in a portable platform with a minimal overhead. This would accomplish one integration step further since the seminal work of [1].

There is room for many developments on the basis of the `RDFContentProvider` interface, bringing our personal data silos closer to the semantic web. Many issues have not been considered in this first development, most importantly the connection to the web. From the Android device to the web, using REST over HTTP to reach (linked) RDF data is not a real problem. From the web to Android, implementing a HTTP server which acts as a REST proxy for Android data accessible to our `RDFBrowser` is not difficult either. The main issue is the conversion of Android local URIs to HTTP URIs.

The system is available at `http://swip.inrialpes.fr` as several applications. It has been tested on the Android emulator and HTC Android 1.6 devices. Specific developments have been made for this version which would not be necessary in newer versions of Android. It is currently being tested on other devices.

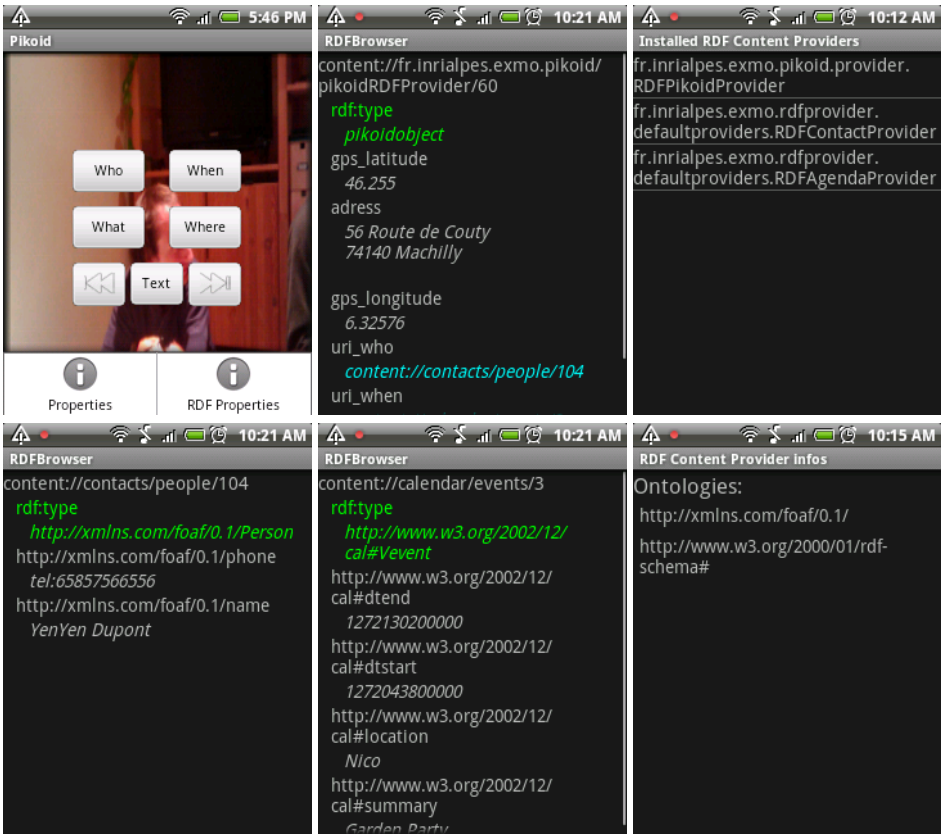


Fig. 2. The Pikoid application annotates images with metadata stored as RDF. RDFBrowser allows for querying this information to the Pikoid RDFContentProvider interface and displaying it. The current picture metadata is shown in the second panel (pikoidRDFprovider/60). From there, it is possible to browse the information available in the address book (people/104) and the calendar (events/3) through the AndroidRDFProvider wrapper. Finally, RDFBrowser also allows for inspecting available RDFContentProviders and the ontologies they manipulate.

References

1. Walsh, N.: Generalized metadata in your palm. In: Proc. 2nd Extreme markup languages conference, Montréal (PQ CA) (2002), `http://conferences.idealliance.org/extreme/html/2002/Walsh01/EML2002Walsh01.html`

4 Acknowledgements

We thank Yu Yang who has programmed part of this demonstration.

Demo: Enriching Text with RDF/OWL Encoded Senses

Delia Rusu, Tadej Štajner, Lorand Dali, Blaž Fortuna, Dunja Mladenić,

Jožef Stefan Institute, Ljubljana, Slovenia
{delia.rusu, tadej.stajner, lorand.dali, blaz.fortuna, dunja.mladenic}@ijs.si

Abstract. This demo paper describes an extension of the Enrycher text enhancement system, which annotates words in context, from a text fragment, with RDF/OWL encoded senses from WordNet and OpenCyc. The extension is based on a general purpose disambiguation algorithm which takes advantage of the structure and/or content of knowledge resources, reaching state-of-the-art performance when compared to other knowledge-lean word sense disambiguation algorithms.

Keywords: RDF/OWL word sense representation.

1 Introduction

A variety of Semantic Web resources in the Linked Open Data (LOD) cloud can serve as knowledge bases for identifying word senses; more general, like DBpedia, W3C WordNet, OpenCyc, or more domain specific like the Gene Ontology, just to name a few of them. Moreover, these resources complement each other, as they span across several domains from music to chemistry and biology.

Enrycher [8] is a service-oriented natural language processing and information extraction framework. It annotates text at various levels, listing: subject – predicate – object triplets (interesting statements) visually interconnected in a semantic graph representation, co-referenced named entities linked to DBpedia, Yago and OpenCyc, keywords and DMOZ categories. In this Demo paper we present an extension of Enrycher[1], relying on a general purpose algorithm which can take advantage of several Semantic Web resources to disambiguate text. This extension annotates words in context with RDF/OWL encoded senses from WordNet [1] and OpenCyc[2]. Given an input text fragment, every word or collocation (word sequence) will be annotated with the appropriate sense in context, and linked to the associated RDF resources defining the sense, in both WordNet and OpenCyc. The motivation behind adding this extension is to provide richer disambiguated annotations of words that are not named entities, and to improve the semantic graph quality, by merging nodes that refer to the same disambiguated concept.

Word sense disambiguation (WSD) is defined as identifying the meaning of words in a given context, and has become a prerequisite for several Semantic Web specific

[1] Demo video: http://marquis.ijs.si/delia/
[2] http://sw.opencyc.org/

tasks like ontology mapping and reasoning. WSD techniques have been previously introduced to validate ontology mappings, by analyzing the semantics of the ontological terms; they exploit ontological context, as well as information provided by WordNet. Aside from WordNet, another knowledge resource, namely Wikipedia has been used for building sense tagged corpora, which have further been employed to train a classifier, obtaining promising results [4]. Wikipedia was also used to automatically extend WordNet with semantic relations (such as synonymy, antonymy, hyponymy, etc.) [6]. However, the existing disambiguation systems mainly retrieve WordNet senses that are not readily usable for Semantic Web applications. Our extension can be easily integrated in other applications that require WSD as a preprocessing step, as word senses are labeled with the corresponding disambiguated RDF/OWL resource. Moreover, we take advantage of ontologies to find word senses, and in future work we plan to add some domain ontologies that can better disambiguate domain specific terminology.

The paper is structured as follows: we start by describing the Enrycher extension integration in Section 2, continue with presenting the disambiguation algorithm in Section 3 and conclude with a section on future work and the demo presentation.

2 Enrycher Extension Integration

Our RDF/OWL word sense annotation extension of Enrycher relies on the Text Preprocessing component which performs sentence splitting, tokenization, part-of-speech tagging and keyword extraction based on a bag-of-words model (see Fig. 1). Both WordNet 3.0 and OpenCyc are processed offline, in order to extract structure and content information. By structure we refer to the semantic relations: synonymy, hypernymy, etc. specific to WordNet, as well as the generalization, specialization, etc. relations encoded in OpenCyc. The content is given by the WordNet *glosses* and the OpenCyc *comments*, and provides descriptions of the word sense.

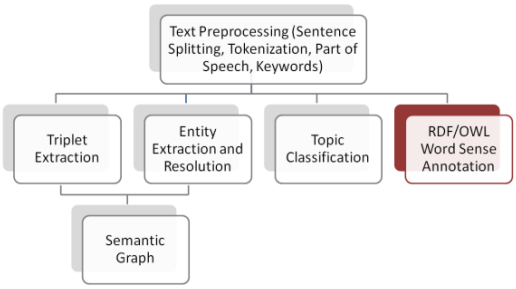


Fig. 1. Enrycher components and their dependencies.

Given an input text fragment, every word or collocation will be annotated with the appropriate sense in context from the aforementioned knowledge resources. If existent, both RDF resources corresponding to WordNet and OpenCyc will be linked. The following section elaborates on the proposed general purpose disambiguation algorithm.

3 Word/Collocation Annotation

We have implemented an unsupervised semantic knowledge based word sense disambiguation algorithm. It relies on the Viterbi algorithm for Hidden Markov Model (HMM) part-of-speech tagging [2], and an initial version was described in [7]. The Viterbi algorithm is a common decoding algorithm for HMM, which was first applied to speech and language processing in the context of speech recognition. We have adapted the algorithm in order to determine, given the senses of words in a sentence, the best sequence of senses that disambiguates the sentence. We start by looking for the senses of nouns, verbs, adjectives and adverbs in one of the two aforementioned knowledge resources. The sequence of observations $O = o_1 o_2 ... o_T$ will represent the T words we disambiguate, while the set of states $Q = q_1 q_2 ... q_N$ define the N senses for a given observation. The sequence of *observation likelihoods* $B=b_i(o_t)$ expresses the probability of an observation o_t being generated from a state i. They are obtained by computing the *cosine similarity* between an ambiguous word description, as defined by the knowledge resource, and information provided by the context (at the level of the sentence, paragraph, etc.). The transition probability matrix $A = a_{11} a_{12} ... a_{n1} ... a_{nm}$ is determined by computing the semantic relatedness between the two senses in state i and j respectively. There have been several relatedness measures proposed in the literature, some of them relying on the knowledge resource structure, others on its content. We have implemented four such relatedness measures, one of which exploiting the resource structure – *Lexical Chains*, while the others take the resource content into account – *Adapted Lesk*, *Vector* and *Vector Pairwise* [5].

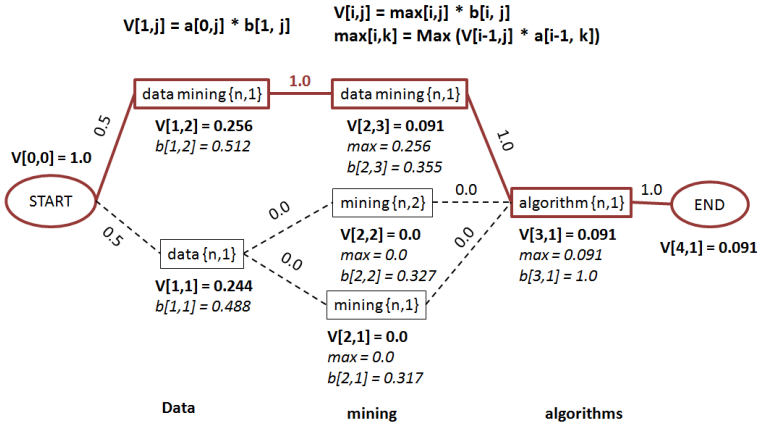


Fig. 2. Disambiguating the phrase *Data mining algorithms* using the proposed algorithm.

We explain the algorithm with the aid of the following example in Fig. 2. To disambiguate the phrase "Data mining algorithms" using WordNet 3.0 as a sense repository, we consider the senses of all words (the word "mining" having the sense of "excavating" or "minelaying"), and in addition the sense of the collocation "data mining" (data processing). We denote the sense part of speech and number in curly brackets. The edges are labeled by state transitions. The collocation is modeled by copying the corresponding sense state, and setting the transition between these two

states to 1.0. There is equal probability to reach any of the sense states of the first word from the start state. Once the final state is reach, we back trace to find the states with the highest associated scores.

We compared our system with others participating in the SemEval 2007 coarse grained all words English disambiguation task based on WordNet senses, obtaining precision/recall/F1 measures of 77.3, lower than the most frequent sense baseline of 78.9, but higher than the best unsupervised disambiguation algorithm participating in the task (SUSSX-FR, based on parsing text and identifying the k nearest neighbors of each word [3]) – 77.0. We also evaluated OpenCyc using a labor-on-demand platform, asking people to determine the correct sense for a given word in context, from a subset of OpenCyc sense definitions, obtaining an average F1 score of 37.55.

4 The Demo and Future Work

The demo will show how the implemented system's web interface annotates words/collocations in a given text fragment with RDF/OWL encoded senses from WordNet and OpenCyc. We are also going to show how to make usage of the system output programmatically, using the LarKC (the Large Knowledge Collider) platform, in order to build Semantic Web applications that rely on WSD.

As for the future work, we plan to integrate other Semantic Web resources from LOD datasets, such as DBpedia, and investigate differences in disambiguation results when using distinct resources and the potential for combining different resources in the same task. Additionally, we aim to apply our WSD algorithm to improve the Enrycher generated semantic graphs.

References

1. Fellbaum, Ch., WordNet: An Electronic Lexical Database. MIT Press (1998)
2. Jurafsky, D., Martin, J. H. Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. Prentice Hall Series in Artificial Intelligence. (2008).
3. Koeling, R. and D. McCarthy. Sussx: WSD using Automatically Acquired Predominant Senses. In Proceedings of the 4th SemEval. pp 314--317. Prague (2007).
4. Mihalcea, R., Using Wikipedia for Automatic Word Sense Disambiguation. In Proceedings of the North American Chapter of the ACL (NAACL), Rochester, NY (2007)
5. Pedersen, T., Patwardhan, S. and Michelizzi, J. WordNet::Similarity - Measuring the Relatedness of Concepts. In Proceedings of NAACL, pp 38--41, Boston, MA (2004).
6. Ponzetto, S.P., Navigli, R., Knowledge-rich Word Sense Disambiguation Rivaling Supervised Systems. In Proceedings of the 48th ACL. pp 1522--1531. Uppsala, (2010).
7. Rusu, D., Fortuna, B. Mladenic, D. Improved Semantic Graphs with Word Sense Disambiguation. Poster. 8th ISWC. Washington, DC (2009).
8. Stajner, T., Rusu, D., Dali, L., Fortuna, B., Mladenic, D., and Grobelnik, M. Enrycher: Service Oriented Text Enrichment. In Proceedings of the 12th Int. Multiconference Information Society. pp. 203--206. Ljubljana, (2009).

4sr - Scalable Decentralized RDFS Backward Chained Reasoning

Manuel Salvadores[1], Gianluca Correndo[1], Steve Harris[2],
Nick Gibbins[1], and Nigel Shadbolt[1]

[1]Electronics and Computer Science,
University of Southampton, Southampton, UK.
{ms8,gc3,nmg,nrs}@ecs.soton.ac.uk
[2]Garlik Ltd, UK. {steve.harris@garlik.com}

http://4sreasoner.ecs.soton.ac.uk/demo.html

Abstract. This poster paper presents the design and implementation of an RDFS reasoner based on a backward chaining approach and implemented on a clustered RDF triplestore. The system presented, called *4sr*, uses 4store as base infrastructure. In order to achieve a highly scalable system we implemented the reasoning at the lowest level of the quad store, the *bind* operation. The *bind* operation in *4sr* traverses the quad store indexes matching or expanding the query variables with awareness of the RDFS semantics.

1 Introduction

The Semantic Web community is promoting RDF stores (or triple stores) as the data storage technology for the Web of Data. RDF stores implement some extra features that make them very attractive for certain type of applications. For instance, data is not bound to a schema and it can be asserted directly from RDF sources (e.g. RDF/XML or Turtle files) due to their native support of Semantic Web data standards. But the most attractive characteristic is the possibility of implementing an entailment regime. Having entailment regimes in a triple store allows us to infer new facts, exploiting the semantics of properties and the information asserted in the knowledge base.

Today, it is still a challenge to query datasets with a few hundred of millions of triples following the RDFS regime for datasets subject to frequent changes. If we want semantic databases to handle big volumes of data transactions then we need to find backward chained approaches that do not add excessive overhead to the query phase.

4sr[1] has been implemented on the *4store* [1] RDF database. *4store* is an efficient, scalable and distributed RDF database which gives us a good platform on which to implement a backward chaining and decentralised approach.

To summarize, the main characteristics of *4sr* are:

[1] 4sr is available from http://4sreasoner.ecs.soton.ac.uk/ under GNU GPL license

– Low level RDFS Backward Chained reasoning implementation.
– Duplicate entailment detection and elimination.
– Named graph management.
– Client applications can disable/enable the RDFS entailment regime via parameters in the HTTP SPARQL endpoint.
– *4sr* doesn't add overhead at the import phase keeping intact *4store*'s import throughput (100kT/s).

2 Background

In the context of distributed techniques, [7] performs Forward Chaining (FC) parallel reasoning to expand the RDFS closure over hundreds of millions of triples. [6] pursues a similar goal and using MapReduce computes the RDFS closure over 865M triples in less than two hours. A continuation of this work has been presented in [5] providing a parallel solution to compute the OWL Horst regime.

[3] presented a novel method based on the fact that Semantic Web data present very skewed distributions among terms. Based on this evidence, the authors present a FC algorithm that works on top of data flows in a p2p-alike infrastructure. This approach reported a materialization of RDFS for 200 million triples in 7.2 minutes on a cluster of 64 nodes.

Obviously, in the last 2-3 years there has been a significant advance on materialization of closure for both RDFS and OWL languages. However very little work has been presented on how to actually query vast amounts of data and how to connect those solutions with SPARQL engines.

3 *4sr* Design and Implementation

The RDFS inferencing in *4sr* is based on two new components that have been incorporated into *4store's* architecture:

– **RDFS Sync:** A new processing node to replicate RDFS statements called *RDFS sync*. This node gathers all RDFS statements from all the storage nodes and keeps a synchronized copy of such information accessible to the *bind* operation in all the segments. After every import, update, or delete, this process extracts the new set of RDFS statements in the KB and sends it to the Storage Nodes. Even for large KBs this synchronization is fast because RDFS statements tend to be a very small proportion of the dataset. This node is also responsible for filtering out entailed duplicates.
– **bind':** The new bind function matches the quads not just taking into account the explicit knowledge but also the extensions from the RDFS semantics.

Figure 1 shows in the architecture how *bind'* and RDFS Sync interact for a hypothetical 2 storage-node deployment.

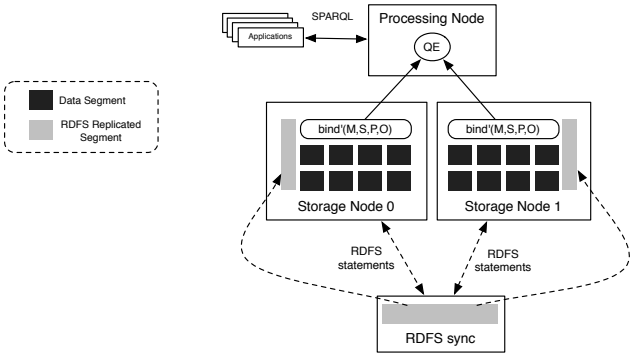


Fig. 1. 4s-reasoner architecture

The current version of *4sr* implements the RDFS rule entailments related to *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain* and *rdfs:range*. These semantics include the rules *rdfs2*, *rdfs3*, *rdfs5*, *rdfs7*, *rdfs9*, *rdfs11*, *ext1*, *ext2*, *ext3* and *ext4* of the RDFS Rule Entailment Regime (see section 7.3 in [2]).

The *bind'* takes a super set of 4 elements $< M, S, P, O >$ to match the segment quads. The first goal in *bind'* is to expand P and O sets so as to broaden the index coverage. After this, the algorithm iterates over the M, S, P, O patterns applying not just an explicit match but a match where the members of the quads are compared taking into account RDFS semantics. Every matched solution with p or/and o unbound will be expanded according to the RDFS entailment regime. As a final step, duplicates are detected and eliminated. This approach has been presented in [4].

This architecture has been implemented in ANSI C 99 using a custom TCP/IP protocol to communicate Storage Nodes and the RDFS Sync Node. Figure 2 gives an overview of how the previous phases are implemented.

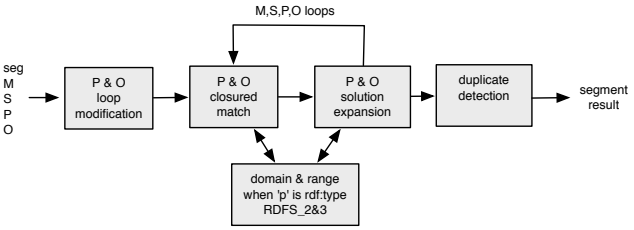


Fig. 2. Bind' processing

4 Evaluation

Due to the lack of space, we refer the committee to [4] where we preliminary tested 4sr using the Berlin SPARQL Benchmark.

5 Conclusions

In this poster paper we present a backward chained decentralized implementation of the RDFS entailment. The novelty of our work is to implement such reasoning in the bind operation of an RDF decentralized database. *4sr* offers a good balance between import throughput and query performance over the RDFS entailment regime. In that sense, *4sr* will support the development of Semantic Web applications where data can change frequently and RDFS inference is required. This poster paper will be accompanied with a demo on site similar to the one available online at:

`http://4sreasoner.ecs.soton.ac.uk/demo.html`

6 Acknowledgements

This work was supported by the EnAKTing project funded by the Engineering and Physical Sciences Research Council under contract EP/G008493/1.

References

1. Harris, S., Lamb, N., Shadbol, N.: 4store: The design and implementation of a clustered rdf store. In: Scalable Semantic Web Knowledge Base Systems - SSWS2009. pp. (p. 94–109) (2009)
2. Hayes, P., McBride, B.: Rdf semantics, w3c recommendation 10 february 2004, `http://www.w3.org/TR/rdf-mt/`
3. Kotoulas, S., Oren, E., van Harmelen, F.: Mind the data skew: Distributed inferencing by speeddating in elastic regions. In: Proceedings of the WWW 2010, Raleigh NC, USA (2010), `http://www.few.vu.nl/~kot/papers/www2010.pdf`
4. Salvadores, M., Correndo, G., Omitola, T., Gibbins, N., Harris, S., Shadbolt, N.: 4s-reasoner: Rdfs backward chained reasoning support in 4store. In: Web-scale Knowledge Representation, Retrieval, and Reasoning (Web-KR3) (September 2010), `http://eprints.ecs.soton.ac.uk/21255/`
5. Urbani, J., Kotoulas, S., Maassen, J., van Harmelen, F., Bal, H.E.: Owl reasoning with webpie: Calculating the closure of 100 billion triples. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC (1). Lecture Notes in Computer Science, vol. 6088, pp. 213–227. Springer (2010)
6. Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F.: Scalable distributed reasoning using mapreduce. In: 8th International Semantic Web Conference (ISWC2009) (October 2009), `http://data.semanticweb.org/conference/iswc/2009/paper/research/374`
7. Weaver, J., Hendler, J.A.: Parallel materialization of the finite rdfs closure for hundreds of millions of triples. In: International Semantic Web Conference. pp. 682–697 (2009)

RightField: Embedding Ontology Term Selection into Spreadsheets for the Annotation of Biological Data

Katy Wolstencroft[1], Matthew Horridge[1], Stuart Owen[1], Wolfgang Mueller[2], Finn Bacall[1], Jacky Snoep[1], Olga Krebs[2], Carole Goble[1]

[1] School of Computer Science, University of Manchester, Manchester, M13 9PL, UK
[2] HITS gGmbH, Schloss-Wolfsbrunnenweg 35, Heidelberg, Germany
<given.family@manchester.ac.uk, given.family @h-its.org>

Abstract. RightField is an open source application that provides a mechanism for embedding ontology annotation support for Life Science data in Microsoft Excel spreadsheets. Individual cells, columns, or rows can be restricted to particular ranges of allowed classes or instances from chosen ontologies. Informaticians, with experience in ontologies and data annotation prepare RightField-enabled spreadsheets with embedded ontology term selection for use by a wider community of laboratory scientists. The RightField-enabled spreadsheet presents selected ontology terms to the users as a simple drop-down list, enabling scientists to consistently annotate their data without the need to understand the numerous metadata standards and ontologies available to them. The spreadsheets are self-contained and remain "vanilla" Excel so that they can be readily exchanged, processed offline and are usable by regular Excel tooling. The result is semantic annotation by stealth, with an annotation process that is less error-prone, more efficient, and more consistent with community standards. RightField has been developed and deployed for a consortium of some 300 Systems Biologists. RightField is open source under a BSD license and freely available from http://www.sysmo-db.org/RightField.

Keywords: ontology annotation, biology, metadata standards, spreadsheets

1 Introduction

In the post-genomic era, the quantity and complexity of biological data produced during standard laboratory projects has increased. New techniques and technologies, in areas such as transcriptomics and proteomics, enable scientists to produce high volumes of data in single experiments. In order to compare and reuse this data, however, rich metadata annotation is also required. The cost of this annotation is high and it is a time-consuming and undervalued process.

In the biological sciences, guidelines and checklists describing what metadata is required for the interpretation and reuse of data are emerging. They are often specified as *minimum information models* [1] with associated controlled vocabularies or ontologies that define the terms that should be used to describe these metadata elements. In some cases, for example, for microarray data, publication submissions are

not accepted unless the accompanying data is compliant with the relevant minimum information model (for microarrays, this is MIAME, the Minimum Information about a Microarray Experiment). However, despite this drive to standardization, there are few tools to help scientists manage this process. RightField was created to lower the barrier of uptake by providing a mechanism for scientists to produce ontology annotation from *within the software environments they already use*.

RightField was developed as part of the SysMO-DB project, which supports a consortium of more than 300 Systems Biologists with data management and exchange. SysMO is a pan-European project to study the Systems Biology of Micro-Organisms, which involves a mixture of high-throughput '*omics* experiments, such as microarray analysis or proteomics, as well as traditional molecular biology and enzyme reaction kinetics. In SysMO-DB, data is standardized by providing spreadsheet templates for different types of experiment to conform to the "Just Enough Results Model" (JERM). The JERM is the SysMO-DB internal structure that describes what type of experiment was performed, who performed it, and what was measured. For experiment types with an established minimum information model, the JERM also complies with this. By combining JERM templates and embedded ontology terms with RightField we provide an infrastructure that promotes and encourages compliance and standardization.

2 Data Generation, Annotation and Reuse

RightField was designed to support a community of laboratory scientists with little experience of metadata management, ontologies or standardization. The primary objective was to provide an application that would allow consistent annotation without changing working practices. Understanding the life-cycle of data generation, annotation and reuse is vital in this process. Capturing experimental metadata at the time of the experiment increases accuracy and increases the likelihood that the annotation is provided by the person performing the experiment. Using the same versions of ontologies for a series of experiments is also vital for accurate comparisons. RightField was designed to be a spreadsheet annotation tool because spreadsheets, particularly MS Excel, are ubiquitous in the laboratory science community for organizing and managing experimental data. Embedding annotation terms in the spreadsheets ensures that term selection occurs at the time of the experiment within the application already in use.

RightField is an open-source, cross-platform Java application which uses Apache-POI for interacting with Microsoft documents. It does not require any special macros, visual basic code or platform specific libraries to run it. It enables users to upload Excel spreadsheets along with ontologies from their local file systems, or from the BioPortal [2] (a repository of biological ontologies available at http://bioportal.bioontology.org/). RightField supports OWL, OBO and RDFS ontologies and RDF vocabularies. In the uploaded spreadsheet, individual cells, or whole columns or rows can be marked with the required ranges of ontology terms. For example, they could include all subclasses from a chosen class, direct subclasses only, all individuals, or only direct individuals. Each spreadsheet can be annotated with terms from multiple ontologies.

Once marked-up and saved, the RightField-enabled spreadsheet contains embedded worksheets with information concerning the origins and versions of ontologies used in the annotation. *This encapsulation stage is crucial*. With everything embedded in the spreadsheet, scientists do not require any new applications to use it and they can complete annotation offline should they wish. This also makes the spreadsheets readily exchangeable and enables a series of experiments to be annotated with the same versions of the same ontologies even if the live ontologies change during this time.

3 RightField Annotation – A Case Study

A research group is studying the impact of changes in flux for different nutrient limitation conditions in *Saccharomyces cerevisiae*. They perform transcriptomics, metabolomics and proteomics experiments and integrate the results. Each experiment is high-throughput; generating complex data which needs to be annotated with rich metadata concerning the experimental conditions, the methods and equipment.

The transcriptomics experiments represent a series which will be compared and analysed together. For publication, this data must conform to the MIAME standard and be deposited in a public microarray repository, such as ArrayExpress (http://www.ebi.ac.uk/microarray-as/ae/) or GEO (http://www.ncbi.nlm.nih.gov/geo/). ArrayExpress provides a service to auto-generate a MIAME compliant template suitable for a particular experiment, but this does not include annotation terms for use in the template. Uploading this auto-generated template into RightField enables an informatics expert to preset ranges of values for annotation.

Figure 1A shows the RightField annotation tool being used to preset annotation values from the MGED ontology (http://www.mged.org/) into the auto-generated MIAME-compliant template, and 1B shows the resulting template with drop-down lists of ontology annotation terms. The marked-up spreadsheet is distributed to the experimentalists to standardize the information that can be recorded and the terms that can be used for annotation. The result is ontology annotation by stealth. The experimentalists do not require specialist knowledge of the ontology resources used.

4 Discussion

RightField is a tool with *light-touch* use of semantic web technologies. The novel part of this work lies in disguising the use of semantics from the end users. Simplicity and unobtrusive embedding with a widely used application is its strength. To compare with similar efforts: ISA Creator (http://isatab.sourceforge.net/isacreator.html) is a bespoke spreadsheet tool designed for experts not end users; the Anzo platform is a commercial product with similar goals (http://www.cambridgesemantics.com/).

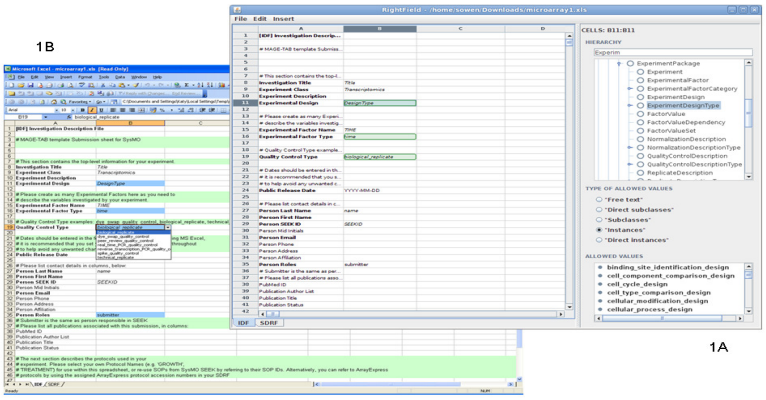


Figure 1 A&B: RightField and a resulting ontology term-embedded spreadsheet

Many experimental biologists have no interest or experience in the use of ontologies and terminologies, but the data they produce is difficult to interpret or reuse without a shared understanding that can be gained from the use of common vocabularies. RightField is the application that bridges this gap. Data can be annotated accurately and at source by the laboratory scientists. This reduces errors and it reduces the time it takes to annotate the data to comply with community standards. Crucially, RightField restricts the choices of annotation terms to a small and manageable set and to make that set accessible and understandable to the scientists.

The next steps for SysMO-DB are to develop methods to fully exploit the corpus of semantically annotated data from RightField Spreadsheets. Compliance with community ontologies means that we can already use Web Services from the BioPortal for term lookup and visualization. In addition, we are developing mechanisms to extract the structured Excel data in RDF to provide further means for searching across the content of spreadsheets and will allow SysMO data to be provided as open linked data. Early investigations using XLWrap [3] are promising.

Acknowledgments
This work was funded by the UK's BBSRC award BBG0102181.

References

1. Taylor, C.F., et al. (2008) Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. Nature biotechnology, 26, 889-896.
2. Noy, N.F., et al. (2009) BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic acids research, 37, W170-1733
3. Langegger A, Wöß W (2009): XLWrap – Querying and Integrating Arbitrary Spreadsheets with SPARQL. 8th Intl Semantic Web Conf, Washington D.C. LNCS 5823, Springer, 2009.

A Graphical Evaluation Tool for Semantic Web Service Matchmaking

Ulrich Lampe, Melanie Siebenhaar, Stefan Schulte, and Ralf Steinmetz

Multimedia Communications Lab (KOM),
Technische Universität Darmstadt, Germany
ulrich.lampe@kom.tu-darmstadt.de

Abstract. Semantic matchmaking – i.e., the task of finding matching (Web) services based on semantic information – has been a prominent field of research lately, and a wide range of supporting tools both for research and practice have been published. However, no suitable solution for the visualization of matchmaking results exists so far. In this paper, we present the *Matchmaking Visualizer*, an application for the visual representation and analysis of semantic matchmaking results. It allows for the comparing of matchmaking approaches for semantic Web services in a fine-grained manner and thus complements existing evaluation suites that are based on rather coarse-grained information retrieval metrics.

1 Introduction

In the envisioned Internet of Services (IoS), organizations will be able to realize business processes using not only internal (Web) services, but also external (Web) services from public marketplaces. One prerequisite to this vision is that functionally equivalent services can be effectively and efficiently identified in the potentially huge collection of available service offers. This service discovery process based on functional requirements is referred to as *matchmaking*. Recent approaches to matchmaking frequently utilize semantic information in addition to the purely syntactical service descriptions [1–3], based on the notion of Semantic Web services (SWS) [4].

In current research, the evaluation of matchmaking approaches is mostly based on standard Information Retrieval (IR) metrics such as precision and recall. These metrics are, for instance, employed in the popular *Semantic Matchmaking Evaluation Environment* (SME2) tool[1]. IR metrics, however, exhibit a rather coarse-grained representation of matchmaking performance, because the service rankings are merged into a selected number of (more or less representative) figures. This type of evaluation provides little insight into the actual matching process. For instance, the similarities that have been assigned to certain pairs of components in a service, such as operations, and the resulting assignments can not be assessed in detail. However, this type of information can be beneficial

[1] http://projects.semwebcentral.org/projects/sme2/

in order to comprehend a matchmaking process in detail. It can also help to analyze required modifications to a (Web) service for actual consumption within, e.g., a complex business workflow.

In this paper, we present the *Matchmaking Visualizer* (MV) as a complement to existing matchmaking evaluation tools. It facilitates the visual representation and analysis of individual service offers and requests and the respective matching results. This permits a much more fine-grained evaluation and assessment of matchmaking engines.

2 Prototype Description

2.1 Representation of Matchmaking Results

A matchmaking result consists of three parts in principle: The description of the initial (1) service request and (2) service offer, and (3) the collection of matching results between pairs of components in both services. The latter express the similarity between individual components in service request and offer, based on their functional and/or semantic description.

We assume that both service request and offer are represented in the identical form, namely as a complete service description file. Reflecting the variety of available service description formats for both WS-* and RESTful services, MV handles the Web Service Description Language (WSDL)[2] in conjunction with Semantic Annotations for WSDL and XML Schema (SAWSDL)[3], Web Ontology Language for Services (OWL-S)[4], Web Application Description Language (WADL) [5], and HTML for RESTful Services (hRESTS) [6].

Each individual matching result consists of the *qualified names* (QNames) of the two regarded components and a flag that indicates if the two components have been assigned to each other. The matching result further links to at least one similarity assessment, which consists of a numerical and textual similarity and a free-text comment. QNames comprise the namespace of a component as well as its local name and are unique to each component in a service description. Because some components, such as individual parameters, may be referenced multiple times, the QName of any parent of a component may optionally be provided as well. Both numerical and textual similarity have been included because they also appear in existing matchmaking approaches. For example, discrete *Degrees of Match* are used in [2], whereas [1, 3] apply continuous numerical measures. The data structure for the representation of matching results is made available as Java classes. This facilitates compatible external matchmakers generating the appropriate data at runtime.

[2] `http://www.w3.org/TR/wsdl`, `http://www.w3.org/TR/wsdl20/`
[3] `http://www.w3.org/TR/sawsdl/`
[4] `http://www.w3.org/Submission/OWL-S/`

2.2 Integration of External Matchmakers

Because MV constitutes an (unofficial) complement to SME2, we have decided to facilitate SME2's plug-in architecture for the integration of external matchmakers. It requires matchmakers to be provided as a JAR file (optionally with additional libraries) along with an XML-based description. As does SME2, MV requires each compatible matchmaker to implement a Java interface class. This interface, *IMatchmaker*, specifies one method, *matchServices*, which conducts the matchmaking process for a service request and offer file. The locations of the respective description files are provided as *URLs*. The matching result is returned using the Java data structure outlined in the previous section. Alternatively, complete matchmaking results may be provided to MV in the form of XML-based files. This can be useful to matchmakers that do not facilitate MV's plug-in architecture, e.g., if they are not implemented in Java.

2.3 Implementation

MV has been implemented in Java using different existing frameworks. For the visualization of graphs, the Java Universal Network/Graph Framework (JUNG)[5] is used. To read semantic service descriptions, we employ Woden4SAWSDL[6] and OWL-S API[7] to read (SA)WSDL and OWL-S documents respectively. For the processing of WADL documents, the generic JDOM[8] framework is utilized. hRESTS documents are first converted into an RDF model, using a XSLT stylesheet by Kopecký[9], and subsequently processed using the JENA framework[10].

2.4 User Interface

MV provides a platform-independent Graphical User Interface (GUI) based on the Java Swing framework. A portal site, including a demo video, screenshots, and additional example resources, is available at:

`http://www.kom.tu-darmstadt.de/~lampeu/iswc-2010/`

The GUI depicts a service request and service offer in the form of a tree structure, representing the increasing level of abstraction. The root, i.e., the whole service, branches into individual leaves, i.e., components, with parameters constituting the most fine-grained level of abstraction. The level of expansion for the tree may be globally selected, thus allowing to focus on the level of abstraction of highest interest. Between all components on the currently selected

[5] `http://jung.sourceforge.net/`
[6] `http://lsdis.cs.uga.edu/projects/meteor-s/opensource/woden4sawsdl/`
[7] `http://on.cs.unibas.ch/owls-api/index.html`
[8] `http://www.jdom.org/`
[9] `http://cms-wg.sti2.org/TR/d12/v0.1/20081202/xslt/hrests.xslt`
[10] `http://jena.sourceforge.net/`

level, the matching results are visualized in the form of edges. By selecting a component, the associated source code from the service description file may be displayed in the GUI. Further, by selecting a connecting edge, details about the respective matching result can be displayed.

MV allows conducting matchmaking on a selected service request and offer using any of the plug-in matchmaking engines. Results may also be combined for direct comparison. That way, users may assess in detail which similarities have been given to pairs of components and why certain components have been assigned to each other. Further, an existing XML result file may be loaded or an individual service may simply be displayed for analysis.

3 Summary

In the work at hand, we presented the Matchmaking Visualizer, a tool for the visualization of semantic Web service matchmaking results. It complements existing evaluation tools for semantic matchmakers that rely on rather coarse-grained information retrieval metrics. Matchmaking Visualizer permits a much more fine-grained, in-detail analysis of matchmaking results, down to the level of individual components and offers support for the direct comparison of different matchmaking approaches. For that matter, the tool is compatible with the plug-in architecture of the popular SME2 evaluation suite, offering the convenient integration of external matchmakers.

Acknowledgements. This work is supported in part by E-Finance Lab e. V., Frankfurt am Main, Germany (`www.efinancelab.de`).

References

1. Schulte, S., Lampe, U., Eckert, J., Steinmetz, R.: LOG4SWS.KOM: Self-Adapting Semantic Web Service Discovery for SAWSDL. In: IEEE 2010 Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems (SEASS '10), Los Alamitos, CA, USA, IEEE, IEEE Computer Society (Jul 2010)
2. Klusch, M., Kapahnke, P., Zinnikus, I.: Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer. In: The Semantic Web: Research and Applications, Proceedings of the 6th European Semantic Web Conference (ESWC 2009). Volume 5554 of Lecture Notes in Computer Science., Springer (2009) 550–564
3. Plebani, P., Pernici, B.: URBE: Web Service Retrieval Based on Similarity Evaluation. IEEE Transactions on Knowledge and Data Engineering **21**(11) (2009) 1629–1642
4. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic Web Services. IEEE Intelligent Systems **16**(2) (2001) 46–53
5. Hadley, M.: Web Application Description Language (WADL). Sun Microsystems, Inc. Technical Reports; Vol. SERIES13103; SMLI TR-2006-153 (2006)
6. Kopeckỳ, J., Gomadam, K., Vitvar, T.: hRESTS: an HTML Microformat for Describing RESTful Web Services. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE (2008) 619–625

RDF On the Go: An RDF Storage and Query Processor for Mobile Devices

Danh Le-Phuoc, Josiane Xavier Parreira, Vinny Reynolds, and Manfred Hauswirth

Digital Enterprise Research Institute,
National University of Ireland, Galway
Galway, Ireland
{danh.lephuoc,josiane.parreira,vinny.reynolds,manfred.hauswirth}@deri.org

Abstract. We present RDF On the Go, a full-fledged RDF storage and SPARQL query processor for mobile devices. Implemented by adapting the widely used Jena and ARQ Semantic Web Toolkit and query engine, it uses Berkeley DB for storing the RDF data, R-Trees for indexing spatial data indexing and a query processor that supports both standard and spatial queries.
By storing and querying RDF data locally at the user's mobile device, RDF On the Go contributes to improving scalability, decreasing transmission costs, and controlling access to user's personal information. It also enables the development of a next generation of mobile applications. RDF On the Go is available for the Android platform and can be downloaded at `http://rdfonthego.googlecode.com/`.

Keywords: RDF storage, SPARQL query processor, mobile devices

1 Introduction

Mobile devices nowadays are equipped with powerful hardware and software, as well as data connectivity and sensing functionalities such as location and orientation. At the same time, the Semantic Web has been evolving and becoming the preferable choice for representing information, with its Resource Description Framework (RDF) for representing semantic data, and query languages such as SPARQL.

While currently most of the processing of semantic data is done in centralized workstations, there are many advantages in executing this processing on mobile devices: i) by distributing the computation among the large number of existing mobile devices, a great level of scalability can be achieved; ii) if the data generated locally on the mobile device such as the sensors, e-mails, calendar appointments can be queried remotely and only the final results need to be sent to another machine for further processing, the data transmission costs can be dramatically reduced; iii) the local processing of user generated data also contributes to the privacy matter, since raw data no longer need to be shared in order to be analysed.

Having such RDF processing capability on the mobile device also enables a new range of applications such as integrating personal information with the

Linked data cloud, Semantic Mobile Augmented Reality [1] and Mobile Semantic Context-based applications.

Although the processing power is on the rise in mobile devices, it is still far behind workstations, and current implementations of RDF storage and query processors for standard computers can not be directly ported to mobile devices, where these resources are still constrained.

While most of the available semantic based mobile applications have to ship their queries to a remote SPARQL Endpoint, some applications, such as micro-Jena[2], Androjena[3] and i-MoCo[4], do store and query RDF data locally. However, they are tailored to specific application scenarios and offer only limited features.

RDF On the Go is the first application to offer a full-fledged RDF storage and SPARQL query processor. It adapts the widely used Jena and ARQ Semantic Web Toolkit and query engine to the constrained mobile device's environment. The RDF data is stored in the B-Trees provided by the lightweight version of the Berkeley DB for mobile devices[5]. Indexes are created for faster data access, where R-trees are used for spatial data indexing, and our lightweight query processor supports standard and spatial SPARQL queries. RDF On the Go is available for the Android platform and can be downloaded at `http://rdfonthego.googlecode.com/`.

The paper demonstrates our RDF On the Go system. Section 2 describes the implementation in more detail and a short demonstration of the system in action is given in Section 3. Section 4 concludes the paper and provide some thoughts for future work.

2 Implementation Details

For storage of the data, RDF on The Go uses a lightweight version of the Berkeley DB that is suitable for mobile devices, which provides a B-Tree implementation for accessing the RDF graphs. For each RDF node, the system employs *dictionary encoding* [2, 3] where node values are mapped to integer identifiers. This reduces the space required to store each RDF node, since the encoded version of the nodes are considerably smaller than the original ones. Moreover, dictionary encoding also allows faster processing, since integer comparisons are cheaper. Fast lookups are achieved in a two-step approach: first, each triple node is stored in multiple ways with different orderings of the triple elements, similar to [1, 6]. Then indexes are built for every ordering of the triple pattern, as proposed in [4]. To support spatial data, we also use R-Trees indexes for storing URIs that have spatial properties. These indexes will output the bindings for spatial graph patterns which are pipelined to the query execution plan.

Currently we support all standard SPARQL operators except aggregation and sorting operators, and the following three spatial operators: "nearby", "within"

[1] `http://www.w3.org/2010/06/w3car/exploiting_lod_for_ar.pdf`
[2] `http://poseidon.ws.dei.polimi.it/ca/?page_id=59`
[3] `http://code.google.com/p/androjena/`
[4] `http://www.cs.vu.nl/~pmika/swc-2008/i-MoCo-Mobile%20Conference%20Guide-weissEtAl_challenge08.pdf`
[5] `http://www.oracle.com/technetwork/database/berkeleydb/overview`

and "intersect". We plan to extend our SPARQL query processor to support most of the patterns described in [5].

To encourage developers to use RDF On the Go to build their applications, we have adapted the core APIs of Jena [6] and ARQ[7] to the Android environment. This allows the developers to manipulate RDF graphs in the same way as they do with the desktop versions of Jena and ARQ . We also reuse some of the Jena and ARQ packages such as the RDF parser and the SPARQL query parser.

3 Demonstration

In this section we provide a short demonstration of our RDF On the Go system. The current prototype loads sample RDF data set in the N3 format[8] to the RDF store on the mobile device. For demonstration purposes, the system loads a dataset that contains all RDF triples from the LinkedGeoData collection[9] that are about places in Galway, Ireland. The screenshots in figure 1 show the prototype running on a HTC Desire mobile device[10]. Figure 1(a) demonstrates the support for spatial SPARQL queries. It displays a map overlay containing all URIs in the dataset that have geo data properties within a particular area. This constrain is represented by the spatial graph pattern *"?uri spatial:within(lat1 lon1 lat2 lon2)"*, where lat1, lon1 are the latitude and longitude of the left upper corner of the area, and lat2, lon2 are the latitude and longitude of the right lower corner. The spatial query pattern used can also be seen this this figure. Each URI is rendered as a marker on the map. By clicking on a marker details of the URI are shown in figure 1(b).

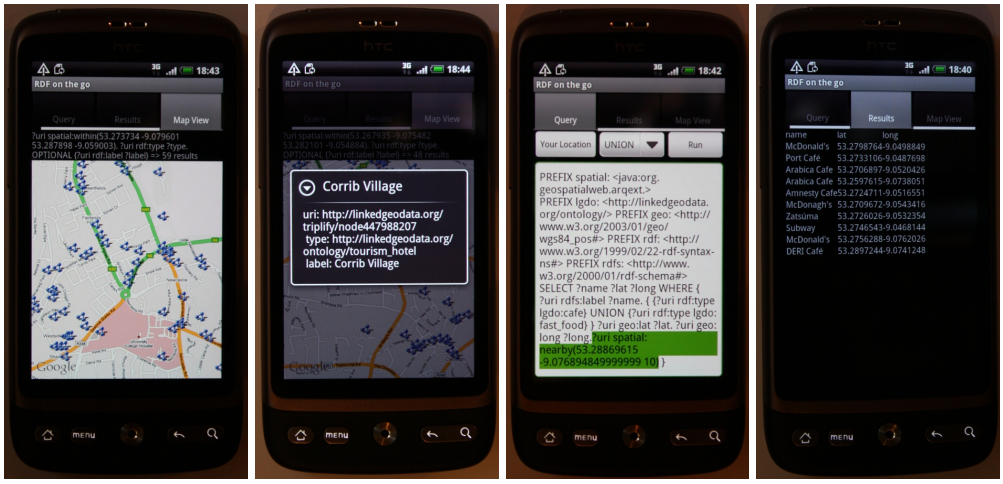
Figure 1(c) shows the interface for standard SPARQL queries. Here we are asking for the 10 nearest 'cafes' or 'fast food restaurants' to the current location. To facilitate entering the query, some common patterns such as OPTIONAL and UNION and the device's current location given by the device's GPS can be added by selecting the corresponding options from a dropdown menu. The results of this query are shown in figure 1(d).

RDF On the Go is available for the Android platform, version 2.1 or newer. The prototype, a video demonstrating the prototype in use, the data collection used in the demonstration and more information are available at `http://rdfonthego.googlecode.com/`.

4 Conclusion

This paper demonstrates the RDF On the Go system, a full-fledged RDF storage and SPARQL query processor for building semantic applications on mobile devices. It not only meets the demand of emerging semantic applications on these devices but also raises many interesting research problems in the areas of data storage and query processing in the context of mobile environments. In the next

[6] `http://jena.sourceforge.net/`
[7] `http://jena.sourceforge.net/ARQ/`
[8] `http://www.w3.org/DesignIssues/Notation3`
[9] `http://linkedgeodata.org`
[10] `http://www.htc.com/www/product/desire/overview.html`



(a) URIs shown on a map (b) Details of an URI (c) Example of a SPARQL query (d) Query results

Fig. 1. Screenshots of the RDF On the Go prototype.

steps, we will focus on efficiency issues, by building a mobile specific query optimizer and more efficient data storage machanism. In context of mobile devices for example, bandwidth and battery are a crucial elements that need to be taken into account in future work.

5 Acknowledgements

This work has been supported by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2), the European Union under Grant No. FP7-224342-ICT-2007-2 (PECES) and the Irish Research Council for Science, Engineering and Technology (IRCSET).

References

1. D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable semantic web data management using vertical partitioning. In *VLDB*, pages 411–422, 2007.
2. J. B. et al. Sesame: An architecture for storing and querying rdf data and schema information. In *Spinning the Semantic Web*, 2003.
3. K. W. et al. Efficient RDF storage and retrieval in Jena2. In *EXPLOITING HYPERLINKS 349*, pages 35–43, 2003.
4. A. Harth and S. Decker. Optimized index structures for querying rdf from the web. In *LA-WEB*, page 71, 2005.
5. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3):1–45, 2009.
6. C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *VLDB*, 1(1):1008–1019, 2008.

Using the Annotation Ontology in Semantic Digital Libraries

L. Jael García Castro[1], Olga X. Giraldo[2], Alexander García Castro[3]

[1] Universität der Bundeswehr München, Werner-Heinsenberg-Weg 39,
85779 Neubiberg, Germany
w31blega@unibw.de
[2] National University of Colombia
Palmira, Valle, Colombia
oxgiraldo@unal.edu.co
[3] University of Bremen, Bibliothekstrasse 1,
28359 Bremen, Germany
cagarcia@uni-bremen.de

Abstract. The Living Document Project aims to harness the collective knowledge within communities in digital libraries, making it possible to enhance knowledge discovery and dissemination as well as to facilitate interdisciplinary collaborations amongst readers. Here we present a prototype that allows users to annotate content within digital libraries; the annotation schema is built upon the Annotation Ontology; data is available as RDF, making it possible to publish it as linked data and use SPARQL and SWRL for querying, reasoning, and processing. Our demo illustrates how a social tagging system could be used within the context of digital libraries in life sciences so that users are able to better organize, share, and discover knowledge embedded in research articles. Availability: http://www.biotea.ws/videos/ld_ao/ld_ao.html

Keywords: Social semantic web, digital libraries, Web 3.0

1 Introduction

Semantic Digital Libraries (SDL) make extensive use of meta-data in order to support information retrieval and classification tasks. Within the context of SDLs, ontologies can be used to: (*i*) organize bibliographic descriptions, (*ii*) represent and expose document contents, and (*iii*) share knowledge amongst users [1]. There have been some efforts aiming to make use of ontologies and Semantic Web technology in digital libraries; for instance, JeromeDL (http://www.jeromedl.org) allows users to semantically annotate books, papers, and resources [2]. The Bricks project (http://www.brickscommunity.org/) aims to integrate existing digital resources into a shared digital memory; it relies on OWL-DL in order to support, organize and manage meta-data [1]. Digital libraries within the biomedical domain store information related to methods, biomaterial, research statements, hypotheses, results,

etc. Although the information is in the digital library, retrieving papers addressing the same topic and for which similar biomaterial has been used is not a trivial task [3]. Ontologies have shown to be useful for supporting the semantic annotation of scientific papers [4] –and thereby facilitating information retrieval tasks. However, as ontologies are often incomplete users should be able to provide additional metadata [3, 5]. Collaborative social tagging and annotation systems have recently gained attention in the research community [6, 7]; partly because of their rapid and spontaneous growth and partly because of the need for structuring and classifying information. Collaborative social tagging is considered exemplary of the WEB2.0 phenomena because such sites use the Internet to "harness" the collective intelligence. It has been observed that several users can tag a resource; tags used for individual resources tend to stabilize overtime [8]. Our implementation uses the Annotation Ontology (AO) [9] for supporting the automatic and manual annotation of research articles. Annotations may be rooted in existing ontologies or provided by users; we are supporting the tagging of atomic components within papers –*e.g.* words, tables, figures. The content of the paper and the corresponding tags are being presented as linked data, this facilitates the interoperability between the paper and external resources –*e.g.* databases, repositories for experimental data, etc. Our approach aims to facilitate sharing, linking, and integrating knowledge across digital libraries and online resources. It also aims to support concept-based collaboration.

2 Enhancing Digital Libraries with the Annotation Ontology

The AO is built upon the Annotea Project (http://www.w3.org/2001/Annotea/); it is also compatible with Newman's (http://www.holygoat.co.uk/projects/tags/), MOAT [7] , and SKOS (http://www.w3.org/2004/02/skos/) ontologies. The AO supports free and semantic annotation over the paper; it facilitates tagging the paper as a whole as well as portions of it, *i.e.* atomic annotation. It also provides facilities for curation, provenance, authoring and versioning. Annotations are not limit to tags but also include notes, comments, erratum, etc.

Our prototype, the LD, makes it possible for users to annotate papers as well as specific sections of them, *e.g.* words, sentences, images, tables, etc. It also interoperates with automatic annotation tools such as Whatizit (http://www.ebi.ac.uk/webservices/whatizit). Annotations are used to improve search and retrieval of papers; it also makes possible to find related papers and researchers. Within the LD, the AO is used to represent the network of concepts and related resources derived from the annotations; in this sense, the AO applied to papers plays a similar role to that played by FOAF in human-centric social networks. The LD facilitates discovering links and improving interaction across papers and researchers.

An atomic annotation is shown in Fig. 1. The document is internally represented by an XML as it is the format used by the publisher; however RDF is also possible. The annotated elements are identified by using XPointer technology (http://www.w3.org/TR/WD-xptr). The provenance is based on FOAF ontology while tagging reuses Newman's and MOAT ontology. The annotation states a related meaning for the term

"partial sequence on psy promoter" to the GeneBank (http://www.ncbi.nlm.nih.gov/genbank/) term AB005238, since the meaning is linked to a well established ontology, the type of the annotation is Qualifier.

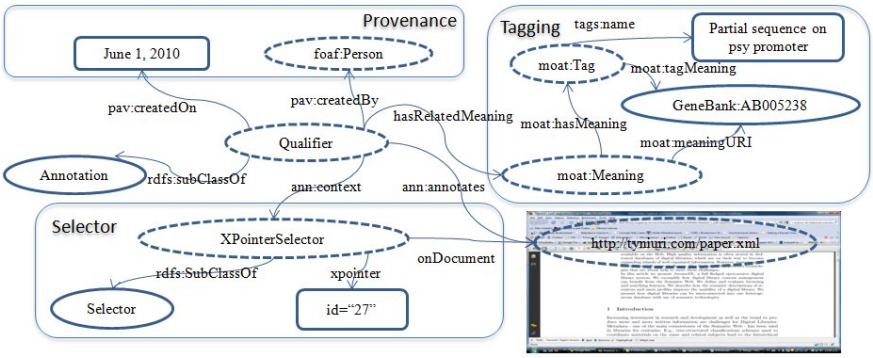


Fig. 1. LD and AO in action

The ***search & retrieval module*** is based on that one usually provided by digital libraries; it uses clouds of annotations and annotators to facilitate navigation and filtering. Once a paper is selected, the ***annotation module*** allows users to identify annotations on the paper, using different colors for different types of annotations, *i.e.* manual and automatic annotations, and also to distinguish amongst categories, *i.e.* species, proteins, genes, etc. It also allows users to manage their annotations and to link them to external resources. Additional information on automatic annotations is provided: links to specialized sources such as UniProt (http://www.uniprot.org). The ***contextual reading module*** allows easily navigating across the paper by jumping from one annotation to other. The ***linked open data module*** allows exporting annotations as RDF, making it possible to use query and reasoning languages such as SPARQL and SWRL. An overview of the LD modules is showed in Fig. 2.

3 Final Remarks

"Less is more" illustrates the collaboration dynamic that embodies the Long Tail (http://en.wikipedia.org/wiki/Long_Tail) principle within the Social Web; a huge number of people providing relatively small contributions that collectively are substantial and significant. Current available metadata in digital libraries is not enough as to support quires such as "*retrieve papers for which microarrays have been used in liver mice*". By making it possible for ontologies and free-provided terms to live together within the scaffold granted by the AO executing such complex queries is possible. It also facilitates the enrichment of the available metadata. In addition, presenting the paper as RDF allows going beyond the PDF without compromising the business model most publishers have –selling access to the full content of the document. The LD approach offers an environment in which researchers harness the

collective intelligence as they are building networks based on similar reading practices. Our future work includes: *i*) enhancing meta-data on authors and co-authors, *ii*) allowing users to organize networks, use social consensus mechanisms, and create relationships between annotations, and *iii*) better orchestrating the LD with existing biomedical ontologies, *e.g.* improving the user interface for large ontologies.

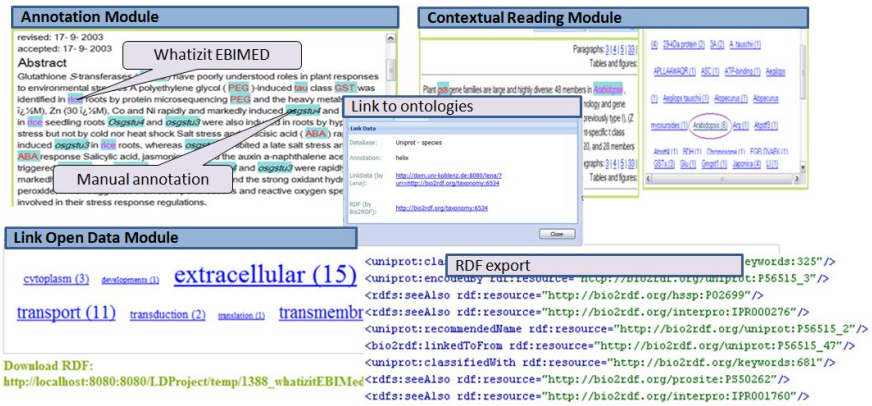


Fig. 2. LD: Modules and Characteristics

References

1. Kruk, S., Haslhofer, B., Piotr, P., Westerski, A., Woroniecki, T.: The Role of Ontologies in Semantic Digital Libraries. European Networked Knowledge Organization Systems (NKOS) Workshop, Spain (2006)
2. Kruk, S., Woroniecki, T., Gzella, A., Dabrowski, M.: JeromeDL -a Semantic Digital Library. International Semantic Web Conference -Semantic Web Challenge, Korea (2007)
3. Garcia-Castro, A., Labarga, A., Garcia, L., Giraldo, O., Montaña, C., Bateman, J.A.: Semantic Web and Social Web heading towards Living Documents in the Life Sciences. Web Semantics: Science, Services and Agents on the World Wide Web **8** (2010) 155-162
4. Shotton, D., Portwin, K., Klyne, G., Miles, A.: Adventures in semantic publishing: exemplar semantic enhancement of a research article. PLoS Computational Biology **5** (2009)
5. Pafilis, E., O'Donoghue, S.I., Jensen, L.J., Horn, H., Kuhn, M., Brown, N.P., Schneider, R.: Reflect: augmented browsing for the life scientist. Nat Biotech **27** (2009) 508-510
6. Kim, H.-L., Scerri, S., Breslin, J., Decker, S., Kim, H.-G.: The State of the Art in Tag Ontologies: A Semantic Model for Tagging and Folksonomies. International Conference on Dublin Core and Metadata Applications, Germany (2008)
7. Passant, A., Laublet, P.: Meaning Of A Tag: A Collaborative Approach to Bridge the Gap Between Tagging and Linked Data. International World Wide Web Conference - Linked Data on the Web Workshop, China (2008)
8. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science **32** (2006) 198-208
9. Ciccarese, P., Ocaña, M., Das, S., Clark, T.P.a.B.-o.: AO: An Open Annotation Ontology for Science on the Web. Bio-ontologies, USA (2010)

Towards Linked Data Services

Sebastian Speiser and Andreas Harth

Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany
lastname@kit.edu

Abstract. Large amounts of data reside in sources which are not web-accessible. Wrappers – small software programs that provide uniform access to data – are often used to transform legacy data sources for use on the Semantic Web. Wrappers, as well as links between data from wrapped sources and data that already exists on the Web, are typically created in an ad-hoc fashion. We propose a principled approach to integrating data-providing services with Linked Data. Linked Data Services (LIDS) can be used in various application scenarios to provide uniform access to legacy data and enable automatic interlinkage with existing data sets.

1 Introduction

The trend towards publishing data on the Web is gaining momentum, particularly spurred by the Linking Open Data (LOD) project[1] and several government initiatives publishing public sector data. Data publishers often use Linked Data principles[2] which leverage established Web standards such as Uniform Resource Identifiers (URIs), the Hypertext Transfer Protocol (HTTP) and the Resource Description Framework (RDF)[3].

Web services are often used to access frequently changing data sets or data which is computed based on supplied input parameters. Wrappers that provide uniform access to services have been created, e.g., in form of the book mashup [1] which returns RDF about books based on Amazon's API, or twitter2foaf[4], which provides access to the follower network of a given user based on Twitter's API. These are useful examples of exposing data from services in a common protocol (HTTP) and data format (RDF).

A lot of data – even data that is already accessible via a uniform access mechanism – still exists in form of unconnected data islands; interlinkage between data on the current Linked Data Web is low. Using ad-hoc wrappers, interlinkage has to be done manually or by service-specific algorithms. Users have to manually associate existing URIs with URIs from wrappers. For example, to establish a link between a person instance (e.g., described using the FOAF vocabulary[5]) and her Twitter account, one has to hard-code the relation between an existing

[1] http://linkeddata.org/
[2] http://www.w3.org/DesignIssues/LinkedData
[3] http://www.w3.org/TR/rdf-concepts/
[4] http://twitter2foaf.appspot.com/
[5] http://xmlns.com/foaf/0.1/

URI and the person's Twitter wrapper URI, which is created by appending the username to `http://twitter2foaf.appspot.com/id/`.

Legacy data should be published so that automated integration and processing is possible, which requires:

- uniform interfaces to all services and data sources, so that data can be easily accessed and integrated; and
- formal service descriptions, so that links between data from different sources can be established automatically.

We present preliminary work on such an approach to create what we call LInked Data Services (LIDS). Using LIDS, vast amounts of idle data can be brought to the Semantic Web via a standardised method for creating Linked Data interfaces to services. LIDS, in addition, enable (semi-)automatic service discovery and integration.

2 Scenario

Consider an investor who wants to assess the outlook of a potential investment target. The investor could vet the company by navigating an integrated dataset containing basic company data, key personnel, competitors, job openings, IP portfolio and previous VC investments in the company. All required information is available on the Web, but with three major drawbacks: i) data is accessible via several incompatible protocols ii) data is encoded in various character encodings and syntaxes and iii) data is sparsely interlinked.

Consider a description of a company office (e.g. `#deu-karlsruhe-`), which contains latitude and longitude attributes (e.g. 48.996 and 8.463), and a GeoNames service that finds nearby places described in Wikipedia[6]. Calling the service with the given latitude/longitude returns:

```
<geonames> <entry>
  <title>Turmbergbahn</title>
  <summary>The Turmbergbahn is a funicular railway near
      Karlsruhe in Germany...</summary>
  <wikipediaUrl>http://en.wikipedia.org/wiki/Turmbergbahn</wikipediaUrl>
  ...
 </entry>
 <entry>
  <title>Durlach</title>
  <summary>Durlach is a borough of the German city of Karlsruhe..</summary>
  ...
 </geonames>
```

Based on the available data one could establish a `foaf:based_near` connection between `#deu-karlsruhe-` and `http://dbpedia.org/resource/Turmbergbahn`, however, that step would require specialised code.

[6] `http://ws.geonames.org/findNearbyWikipedia`

3 Linked Data Services

In the following we describe the LIDS approach using an example. We focus on data-providing services, which return data that is related in a specific way to the given parameters. We extend that notion for Linked Data Services as follows:

A Linked Data Service (LIDS) provides HTTP URIs for entities representing service inputs that encode parameters as key-value pairs in the query string. Dereferencing the URI via HTTP GET returns an RDF description of the service input entity, its relation to the service output and the output data itself. Both input and output of a LIDS are formally described using SPARQL.

For example, a LIDS wrapper for the exemplary GeoNames service is available at `http://geowrap.openlids.org/findNearbyWikipedia`. Input parameters are encoded in the query string of the service URI, e.g. `http://geowrap.openlids.org/findNearbyWikipedia?lat=48.996&lng=8.463`. To establish an "non-information" URI we add a localname to derive the entity URI, e.g. `http://geowrap.openlids.org/findNearbyWikipedia?lat=48.996&lng=8.463#point`. Looking up the entity returns a "non-information" URI denoting the input point and the relation to URIs of nearby places from Wikipedia (we substitute Wikipedia URIs with those from DBpedia[7]):

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbp: <http://dbpedia.org/resource/> .
<http://geowrap...Wikipedia?lat=48.996&lng=8.643#point>
                                foaf:based_near dbp:Turmbergbahn ;
                                foaf:based_near dbp:Durlach .
```

A LIDS returning a URI with the input parameter allows for adding additional descriptions to that URI; e.g., we could use the `owl:sameAs` property to establish equivalence between `http://geowrap.openlids.org/findNearbyWikipedia?lat=48.996&lng=8.463#point` and `#deu-karlsruhe-`.

We propose a simple vocabulary for LIDS[8] that defines a class for LIDS and a description property relating a LIDS to a SPARQL query describing the service. The LIDS description of the geowrap service is as follows:

```
CONSTRUCT { ?point foaf:based_near ?feature . }
FROM <http://geowrap.openlids.org/findNearbyWikipedia>
WHERE {
    ?point geo:lat ?lat .
    ?point geo:long ?lng .
}
```

The use of unsafe variables and the meaning of the FROM clause are not completely adhering to the SPARQL standard, but have their intuitive meaning. The FROM clause encodes the base URI of the service. Please note the user of unsafe variables (here: `?feature`), which are bound by the service during execution. The variable appearing in both CONSTRUCT and WHERE clause (here: `?point`) denotes the input entity and is used for building entity URIs.

[7] `http://dbpedia.org/`
[8] `http://openlids.org/vocab`

4 Related Work

In contrast to our proposal, early Web service description formalisms, such as WSDL, do not model the relation between input and output data, which leaves space for ambiguities. General Semantic Web Services approaches include OWL-S[9] and WSMO [4] still lack practical applications, which can be partially explained by their complexity and their use of formalisms that are not familiar to all Semantic Web users. Semantic descriptions of stateless services (e.g. [3, 2, 5]) define service functionality in terms of input and output conditions, however, employ proprietary description formalisms and/or static definition of inputs and outputs. Our approach is more flexible and better integrates with the Linked Data principles.

5 Conclusions and Future Work

We have presented preliminary work on an approach to integrating data services with Linked Data. A uniform access method – compatible to Linked Data principles – in combination with a lightweight service description, enables the creation of LInked Data Services (LIDS). LIDS have formal, yet lightweight and flexible descriptions based on SPARQL, a language which is familiar to many Semantic Web users and developers.

LIDS can be used in a number of scenarios. In the future, we plan to develop and study algorithms that automatically enrich existing Linked Data with links to LIDS, which can happen in different settings, e.g.:

- processing of static RDF data, inserting links to LIDS, and storing the result;
- dynamically adding links to LIDS to the result of a Linked Data endpoint;
- locally augmenting retrieved data with data from LIDS in a data browser;
- using data from LIDS during SPARQL query processing.

For a formal treatment of LIDS, more example services and up-to-date information visit `http://openlids.org/`.

References

1. C. Bizer, R. Cyganiak, and T. Gauss. The RDF Book Mashup: From Web APIs to a Web of Data. In *Workshop on Scripting for the Semantic Web, at ESWC*, 2007.
2. D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding Semantic Matching of Stateless Services. *AAAI06*, pages 1319–1324, 2006.
3. K. Iqbal, M. L. Sbodio, V. Peristeras, and G. Giuliani. Semantic Service Discovery using SAWSDL and SPARQL. In *International Conference on Semantics, Knowledge and Grid*, 2008.
4. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.
5. W.-f. Zhao and J.-l. Chen. Toward Automatic Discovery and Invocation of Information-Providing Web Services. In *Asian Semantic Web Conference*, 2006.

[9] `http://www.w3.org/Submission/OWL-S/`

SPARQL Views: A Visual SPARQL Query Builder for Drupal

Lin Clark

Digital Enterprise Research Institute, National University of Ireland, Galway
lin.clark@deri.org

Abstract. Publishing Linked Data on the Web has become much easier with tools such as Drupal. However, consuming that data and presenting it in a meaningful way is still difficult for both Web developers and for Semantic Web practitioners. We demonstrate a module for Drupal which supports visual query building for SPARQL queries and enables meaningful displays of the query result.

Keywords: User Interfaces, End-user Programming, CMS

1 Introduction

Integrating Semantic Web technologies into mainstream Content Management Systems (CMSs) has been established as a way to increase adoption of the Semantic Web [3][2]. One example of incorporating these technologies in a mainstream system is the RDF in Drupal 7 core initiative, which enables over 300,000 sites[1] in joining the Semantic Web by exposing the content's structure as RDF.

While previous work in Drupal has made it easy for site administrators to *publish* RDF, modules that enable sites to *consume* RDF still require knowledge of SPARQL[2]. Because most Web developers do not meet this knowledge requirement, in practice the data is only directly accessible for Semantic Web practitioners. This means that one of the greatest potential benefits of RDF, the potential for data integration between sites, goes untapped for many sites.

Conversely, many Semantic Web practitioners do not have a full understanding of Web application development, which limits the displays they can create of the queried data. Where displays on the data are created, it is often by developing bespoke systems that cannot be reused by other Semantic Web practitioners.

We believe it is possible to design tools that support both the average Web developer and the Semantic Web practitioner's differing needs. We demonstrate a tool that supports drag-and-drop, visual query building over RDF data and that enables quick, easy, reusable presentation of that data.

[1] Drupal core usage statistics from `http://drupal.org/project/usage/drupal`

[2] Modules include SPARQL, http://drupal.org/project/sparql, and RDF SPARQL Proxy, http://drupal.org/project/rdfproxy

2 Use Case

For illustration of the tool and its usefulness, we imagine a research institute Web site such as `http://deri.ie/`. The site includes pages for researcher profiles. A researcher's profile page pulls the researcher's publication list from an dataset such as DBLP[3]. The profiles will be set up and administered by a Webmaster who is unfamiliar with SPARQL, but knows a small amount of PHP and is comfortable with HTML.

We walk through this use case in a video demonstration available at `http://lin-clark.com/iswc`. We give an analysis of the challenges faced in building such a Web site and our solutions to those challenges below.

3 Usability Challenges in Query Building

3.1 Figuring out where to start

Inexperienced users are overwhelmed when given too many interaction options or, even worse, when given a blank slate where they must enter a command language like SPARQL [5].

Interaction strategy—To assist users in finding where to start, we initially offer one single point of interaction. We make the assumption that most queries are centered around predicates, and we use the drag-and-drop predicate list as the first means of interaction. We then animate the addition of subject and object to that predicate in order to guide the user to the next step, using the strategy of sequential affordance[4].

3.2 Figuring out which predicates to use in the query

When accessing an arbitrary endpoint, end users have little support in understanding what data is contained in the dataset and how it is linked.

Interaction strategy—The endpoint is queried to determine which predicates are used in the dataset. Only predicates that are present in the dataset are displayed. An autocomplete search box allows the user to filter to appropriate terms.

3.3 Declaring prefix mappings

Declaring prefix mappings requires knowledge of both syntax and of standards and inexperienced users can have a hard time finding the appropriate namespace for a vocabulary.

Interaction strategy—The prefix declaration is automatically generated for the user from the predicates that are used. If a namespace mapping is not available in the system (which uses prefix.cc as it's source), then the full URI is used in the predicate display and the query.

[3] `http://dblp.l3s.de/d2r/sparql`

3.4 Understanding the logical structure of queries

While conjunction and disjunction based logic is ubiquitous in everyday life and is thus intuitive to a large degree, the syntax of SPARQL obscures that logic for the inexperienced user.

Interaction strategy—Where SPARQL syntax requires mental computation of the logic, visual representation of queries allows perceptual inference of the underlying logic[1]. We provide a visualization which makes the triple based logical patterns clear and offers affordances that reflect the possibilities of the graph structure. We currently only support basic conjunctive queries as these are the easiest to conceptualize[6], but are looking at ways to visually express other patterns.

4 Usability Challenges in Result Display

4.1 Displaying results in multiple display formats

SPARQL results are not useful in their raw format, but need to have tailored displays in order for the pattern of information within the data to be communicated. Semantic Web practitioners often create bespoke systems for this display, leading to duplication of effort within the Semantic Web community.

Interaction strategy—We integrated the query building tool with a widely deployed Drupal module, Views. This module offers a pluggable system for displaying query results. Switching from an HTML table view, to a Google API chart view, to a JavaScript Exhibit of the data is easy to do through the Views user interface. Because SV is integrated with this pluggable system, any style and display plugins that the Drupal community develops to display various kinds of data can also style SPARQL results.

4.2 Using context to rewrite the query

Information is most useful if it is tailored to the page where it is displayed. For instance, in the use case above, when a visitor visits a faculty member's profile, the query should return only publications authored by that faculty member.

Interaction strategy—We provide support for Drupal's Token API. This allows users to easily register tokens—small placeholder variables—and use these in their queries. The token is then evaluated at page load, so context, such as which user's profile is being viewed, can be assessed when creating the query.

5 Future Work

5.1 Predicate Preprocessing and Ranking

We currently use a DISTINCT query to get the predicates from a dataset. For certain endpoints, such as the one at `http://dbpedia.org/sparql`, this query

will timeout and not return results. We plan to create a Web service that can extract predicates from crawls preprocess the predicates for different endpoints. This service could provide additional data, such as the predicate definition and a ranking based on predicate usage within the dataset.

5.2 Dataset Selection

A barrier we have not addressed in the current work is the selection of appropriate datasets. We plan to create tools to guide users in finding an appropriate dataset.

5.3 Federated Queries

Currently, there can only be one endpoint or dataset defined per view. However, one of the most exciting potentials of the Semantic Web is to mix data from multiple sources and we plan to explore how this can be supported in SV.

5.4 Evaluation

We believe that it is possible to support and encourage the user's increasing understanding of SPARQL by lowering the barrier to entry and focusing on learnability. We plan to evaluate this assertion, to see whether interaction with SPARQL Views increases the acceptance of SPARQL and the understanding of the syntax itself, enabling users to incrementally improve their understanding of the query language.

Acknowledgments. This work was funded by Google Summer of Code, LOD Around-The-Clock (LATC) ICT-256975, and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). Thank you to Drupal contributors Daniel Wehner and Laura Scott for their guidance and feedback.

References

1. Tiziana Catarci, Maria F. Costabile, Stefano Levialdi, and Carlo Batini. Visual query systems for databases: A survey, 1995.
2. Stéphane Corlosquet, Renaud Delbru, Tim Clark, Axel Polleres, and Stefan Decker. Produce and consume linked data with drupal! In *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 763–778. Springer Berlin / Heidelberg, 2009.
3. Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic mediawiki. In *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942. Springer Berlin / Heidelberg, 2006.
4. Joanna Mcgrenere. Affordances: Clarifying and evolving a concept. In *Proceedings of Graphics Interface 2000*, pages 179–186, 2000.
5. Donald A. Norman. *The Design of Everyday Things*. Basic Books, New York, 2002.
6. Vladimir M. Sloutsky and Yevgeniya Goldvarg. Mental representation of logical connectives. *The Quarterly Journal of Experimental Psychology*, 57A(4):636–665, 2004.

The Polish interface for Linked Open Data

Aleksander Pohl

Computational Linguistics Department,
Jagiellonian University, Cracow, Poland
aleksander.pohl@uj.edu.pl
http://klon.wzks.uj.edu.pl/cycdemo

Abstract. This paper describes an application which aims at producing Polish descriptions for the data available as Linked Open Data, the MusicBrainz knowledge base contents in particular.

1 Introduction

It is trivial to say that the natural language is the most *natural* way of conveying information for people. No matter how many formal, unambiguous languages given person knows, when it comes to quick transfer of semantic content, natural language is always the best option. This fact is reflected in the recommendations for the user interface designers. For example one of the Nielsen's [2] usability heuristics[1] is: *The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.*

This observation stays in a contrast to the fact, that Linked (Open) Data[2], which seems to be the most visible part of the Semantic Web, is usually presented in a form reflecting the structure of RDF triples[3]. And even if the content is available in a more human-readable format[4], it still looks table-like and in most cases it is only available in English.

As a result, it is hard to imagine, that a person would prefer the strict and organized information from DBpedia (who reads abstracts in several languages?) over the same information found in Wikipedia. Obviously this is not an argument against the Semantic Web, or Linked (Open) Data in particular, this is only an observation, which shows, that the way the semantic data is presented to the (non-English speaking) end users could be improved.

2 The Idea

The general idea of our approach is as follows: provide an Internet service, which allows for translation of the data available as RDF triples into natural language

[1] http://www.useit.com/papers/heuristic/heuristic_list.html
[2] For reference see: http://linkeddata.org
[3] See examples in DBpedia: http://dbpedia.org/page/Cher and Citeseer http://citeseer.rkbexplorer.com/description/resource-CS107764.
[4] See examples in OpenCalais http://d.opencalais.com/er/company/ralg-tr1r/9e3f6c34-aa6b-3a3b-b221-a07aa7933633.html and MusicBrainz http://musicbrainz.org/artist/bfcc6d75-a6a5-4bc6-8282-47aec8531818.html

descriptions. In the basic scenario, the system should be able to translate subject-predicate-object structures, such as `http://example.com/John foaf:nick ,,Johnny1''`[5] into „John's nickname is «Johnny1»". In more sophisticated scenarios, it should allow for embedding links, pictures, videos, etc. thus should be able to translate predicates such as `foaf:homepage` or `foaf:img`. And in the most advanced scenario, it should be able to produce descriptions for complex structures, such as series of events, assuring that the chronology is correct, the mentions of people and places are not replicated too often, and so on.

The value added by such a system comes form the fact, that whenever the developer of some Linked (Open) Data base uses an ontology which is known to the system, he doesn't have to write the RDF-to-NL module, since, if the meaning of the predicates used is well-defined, their natural language paraphrases should stay intact. In fact – this solution is not bound to Linked (Open) Data. With some modifications it might be used in any application with natural language (probably multilingual) interface, provided that its data structures were mapped to the above-mentioned ontology.

3 The Problems

In the case of the basic scenario, the paraphrase based on simple templates connected with particular predicates and filled with the labels of the resources or the actual values (like numbers) seems to be easily achievable. Similar approach is used in many applications with multilingual interface – when the application is localized, the templates are translated and they are filled with the data independently of the language (e.g. in Gmail there is a message at the bottom of the page: „Obecnie używasz 8 MB (0%) z 7500 MB." – „You are currently using 8MB (0%) of your 7500MB.").

In fact, this solution is not as good, as it seems, at least for inflectional languages in general and Polish in particular. The first and most problematic issue is the necessity to accommodate the gender of the subject with the verb. Thus even for the simplest sentence to be fully sound, the information required is not directly present in the RDF triple. If we consider the fact, available in the MusicBrainz knowledge base, that Cher was born on the 20^{th} of May 1946, we have to know that Cher is a women, to properly construct the sentence „Cher *urodziła* się 20 maja 1946 roku". The same fact about Michael Jackson is paraphrased as „Michael Jackson *urodził* się 29 sierpnia 1958 roku". So the template has to be adjusted with respect to the gender of the subject, but MusicBrainz doesn't contain the necessary information.

Another problem tightly connected with inflectional nature of Polish, is the inflection of numerals. In English, when some application is localized, usually there is only room for two word forms: singular and plural and the inflectional scheme is trivial (e.g. „There is one *track* on the CD/There are several *tracks* on the CD."). But in Polish numerals influence the case of the subordinate nominal

[5] See `http://xmlns.com/foaf/spec/` for the definitions of the properties.

phrase and the gender of the nominal phrase influences the form of the numeral as well. The (partial) scheme is as follows: „1(jedn*o*) krzesł*o*" (one chair) – „2(dw*a*) krzesł*a*" – „5(pięć) krzes*eł*" – „12(dwanaście) krzes*eł*" – „22(dwadzieścia dw*a*) krzesł*a*" – „25(dwadzieścia pięć) krzes*eł*"; „1(jedn*a*) ścieżk*a*" (one track) – ... The scheme is further complicated, when the numeral phrase is an argument of a verb, like in the sentence „Przysłuchiwałem się 1(jedn*ej*) ścież*ce*" (I listened to 1 track).

4 The Partial Solution

We argue that the most reasonable solution for the above mentioned problems (as well others connected with RDF-to-NL translation) is the implementation of the logic-to-NL translation system for a general ontology. Such an ontology should distinguish men from women (distinction not always present in domain-specific ontologies, like Music Ontology[6]) and other things. It should also contain many other concepts and relations, for the logic-to-NL system to be at least partially complete. Being a part of Linked Open Data is also necessary. Such a system would be a firm base for the RDF-to-NL translation application.

We think that the Cyc ontology [1] is the best candidate. First of all – it has a logic-to-English translation module, so providing the English paraphrases for RDF triples should be easy. It is a general ontology, with very broad coverage (in terms of concepts – several hundreds of thousands and relations – more than 20 thousands) and provides a Semantic Web end-point[7]. It is also well connected with DBpedia and some other elements of Linked Open Data.

Still Cyc contains only the English lexicon, that is the mapping between concepts and English words. In our previous research we created an algorithm for mapping of Cyc concepts to Polish one-segment expressions [4]. This research is further carried out and the results will be presented on the International Multiconference on Computer Science and Information Technology in the Computational Linguistic – Applications track[8]. The important fact about the constructed Polish lexicon is that, it is based on the Polish inflectional dictionary described in [3]. This means that the information necessary for accurate paraphrases accommodating common nouns and verbs as well as numerals and common nouns will be available in it.

Still the inflectional dictionary doesn't contain most of the proper names which are so common in knowledge bases and as a result the accommodation of subject and verb has to be carried out differently[9]. This is the case where the Semantic Web plays its part – the information which is so easily available for people, namely the gender of a person, might be deduced by the system from Linked Open Data and the Cyc ontology. First of all the concepts representing men and

[6] `http://musicontology.com/` – the ontology used in MusicBrainz

[7] `http://sw.opencyc.org/`

[8] `http://www.imcsit.org/pg/358/281`

[9] The inflection of unknown proper names is not covered in this research. Thus the paraphrase is not accurate if the proper name occurs at the argument position.

women should be identified in Cyc (it is #$MaleHuman and #$FemaleHuman respectively). Then the categories of the object should be looked up in the source knowledge base. If any of it is a specialization[10] of one of the above mentioned classes, the gender could be determined. If not, the same procedure should be applied for the knowledge bases containing synonyms of the object in question. The procedure would stop if any knowledge base allowed for determining the gender or certain threshold (timeout, number of visited knowledge bases, etc.) were reached.

For example Cher, whose MusicBrainz address is `http://dbtune.org/musicbrainz/resource/artist/bfcc6d75a6a5-4bc6-8282-47aec8531818`, is linked to the DBpedia resource `http://dbpedia.org/page/Cher`, where one of her OpenCyc types is `http://sw.opencyc.org/2008/06/10/concept/Mx4rvVjW5ZwpEbGdrcN5Y29ycA` (female person) which directly indicates, that she is a *woman*.

5 The Application

An application was build which creates Polish paraphrases for portion of the knowledge available in the MusicBrainz knowledge base. It is available under the URL: `http://klon.wzks.uj.edu.pl/cycdemo` and is integrated with the tool used for mapping Cyc symbols to Polish words and expressions. The actual functionality is available when the user clicks the „search" (szukaj) button and selects the „Sparql" engine. When he enters the name of an artist or an album (case sensitive), the resources found in the base are presented[11]. If he clicks the white button on the right of the resource, he will see the table with properties describing the resource. The information is presented systematically, but it is not easy to understand. If the user clicks the yellow button, he will see the Polish paraphrase of the data. Not all the data which is available in the base is presented, but the text is much more appealing and intelligible.

References

1. Lenat, D.B.: CYC: A large-scale investment in knowledge infrastructure. Communications of the ACM 38(11), 33–38 (1995)
2. Nielsen, J.: Usability Inspection Methods, chap. Heuristic Evaluation, pp. 25–62. John Wiley & Sons (1994)
3. Pisarek, P.: Słowniki komputerowe i automatyczna ekstrakcja informacji z tekstu, chap. Słownik fleksyjny, pp. 37–68. Uczelniane Wydawnictwo Naukowo-Dydaktyczne AGH (2009)
4. Pohl, A.: Automatic Construction of the Polish Nominal Lexicon for the OpenCyc Ontology. In: Recent Advances in Intelligent Information Systems (2009)

[10] The equivalence of the classes in given knwoledge base and Cyc might be established via the `http://sameas.org` service.

[11] The query takes much time, since it consists of many sub-queries to the Semantic Web end-points.

HANNE - A Holistic Application for Navigational Knowledge Engineering

Sebastian Hellmann, Jörg Unbehauen, Jens Lehmann

AKSW Research Group, http://aksw.org, Universität Leipzig, Germany
lastname@informatik.uni-leipzig.de

Abstract. Although research towards the reduction of the knowledge acquisition bottleneck in ontology engineering is advancing, a central issue remains unsolved: Light-weight processes for collaborative knowledge engineering by a massive user base. In this demo, we present HANNE, a holistic application that implements all necessary prerequisites for Navigational Knowledge Engineering and thus reduces the complexity of creating expressive knowledge by disguising it as navigation. HANNE enables users and domain experts to navigate over knowledge bases by selecting examples. From these examples, formal OWL class expressions are created and refined by a scalable Iterative Machine Learning approach. When saved by users, these class expressions form an expressive OWL ontology, which can be exploited in numerous ways: as navigation suggestions for users, as a hierarchy for browsing, as input for a team of ontology editors.

1 Introduction

Over the past years, structured data has become widely available. Still, the retrieval of dedicated knowledge for given applications or research questions out of these data sources remains a tedious process. A domain expert might have a very precise idea of the concepts she would like to retrieve from a knowledge source. Yet, she faces a number of challenges when trying to retrieve corresponding examples out of a particular data set.

Due to their sheer size, users of these knowledge bases can hardly know which identifiers are used and are available for the construction of queries. Furthermore, domain experts might not be able to express their queries in a structured form at all, but they often have a very precise imagination what kind of results they would like to retrieve. A historian, for example, searching in DBpedia [2] for ancient Greek law philosophers influenced by Plato can easily name some examples and if presented a selection of prospective results she will be able to quickly identify false results. However, she might not be able to efficiently construct a formal query adhering to the large DBpedia knowledge base a priori.

The construction of queries asking for objects of a certain kind contained in an ontology, such as in the previous example, can be understood as a class construction problem: We are searching for a class expression which subsumes

exactly those objects adhering to our informal query (e.g. ancient Greek law philosophers influenced by Plato [1]).

In recent years, several methods have been proposed for constructing ontology classes by means of Machine Learning techniques from positive and negative examples (see [3] for an overview). Due to their dependency on reasoning methods, these techniques are tailored for small and medium size knowledge bases and cannot be directly applied to large knowledge bases. The scalability of the algorithms is ensured, however, by reasoning only over "interesting parts" of a knowledge base for a given task [1]. As a result users of large knowledge bases are empowered to construct queries by iteratively providing positive and negative examples to be contained in the prospective result set.

In this paper, we present HANNE - a Holistic Application for Navigational kNowledge Engineering. HANNE allows for the extraction of formal definitions of user-defined concepts and the corresponding examples out of arbitrary and possibly large RDF data sets. Based on initial examples given by the user, HANNE learns a formal OWL Class Expression of the concept that the user is interested in. This expression is converted into a SPARQL query[2] and passed to a triple store database with reasoning capabilities. The results are gathered and presented to the user to choose more examples, to refine the query, and to improve the formal definition at will.

Our tool, available online at `http://hanne.aksw.org`, addresses and circumvents the barriers to the acquisition of knowledge out of data sets: (1) it does not need any deployment and provides a user interface in a familiar surrounding, the browser, (2) the meaning of the identifiers used in the knowledge source is made explicit by the tool, and, finally, (3) the application uses OWL; the results are thus represented in a readable, portable and sustainable way.

2 Example Usage

At the time of writing, the DBpedia ontology class `http://dbpedia.org/ontology/Country` contained 2505 instances, including all current countries as well as all historic countries, most of which ceased to exist nowadays.

On April 27th, 2010, there has been a discussion on the DBpedia mailing list[3] on how to retrieve (via SPARQL) a list of current countries only, as the coverage of the OWL class was obviously too imprecise (or its definition was too broad). One suggested solution[4] by a DBpedia expert was to manually include a filter for *dbo:dissolutionYear*[5] within the SPARQL query.

[1] technically we mean OWL class expressions such as *AncientGreekPhilosopher and influencedBy value Plato* in Manchester OWL Syntax `http://www.w3.org/TR/owl2-manchester-syntax/`

[2] `http://www.w3.org/TR/rdf-sparql-query/`

[3] `http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg01652.html`

[4] `http://www.mail-archive.com/dbpedia-discussion@lists.sourceforge.net/msg01658.html`

[5] `http://prefix.cc/dbo`

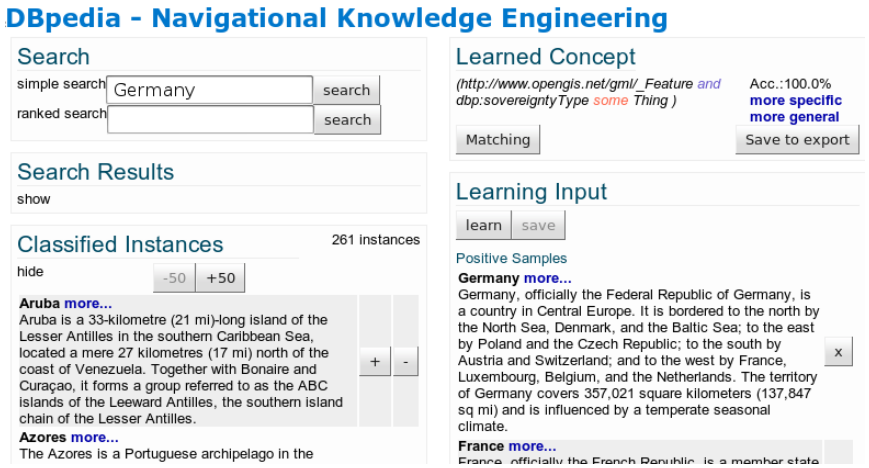


Fig. 1. Screenshot of the left and middle part of `http://hanne.aksw.org`: Real countries in DBpedia. (The right part containing a list of stored concepts and additional features is omitted for a larger image and readability)

Although, the request (originally posted by a DBpedia user) was answered, two shortcomings remain: 1. The answer was not recorded or documented in a sustainable way (e.g. incorporated as OWL class within the ontology) 2. The process of finding the answer was very tedious for the user. He had to wait several days and required the help of an ontology expert that was familiar with the existing vocabulary.

In the following, we will explain step by step, how an OWL class (named e.g. *Real_Country*) can be created without hardly any effort and previous knowledge with HANNE. On the left side of Figure 1, a full text search over the DBpedia data set can be conducted. This represents the entry point, as initial examples have to be chosen to bootstrap the learning process. In our case, a user could start by searching for "Germany". From the search result, she picks *Germany* as a positive example and *East Germany*, *West Germany*, *Nazi Germany* as negatives. After she has pressed the *learn* button (middle, above given examples) a formal OWL definition (in Manchester OWL Syntax) is presented in the top middle (*Learned Concept*) in this case *http://www.opengis.net/gml/_Feature and dbp:sovereigntyType some Thing* . She now has two options on how to proceed: 1. if she finds the learned concept adequate, she can label (e.g. *Real_Country*), comment (*Countries, which are officially accepted and still exist*) and save it to export a complete list of instances 2. Alternatively, she can retrieve instances matching the learned OWL class, which are then displayed on the left side *Classified Instances*. These instances can be further evaluated and more positive and negative examples can be chosen to iterate the process. In our case, a total of 261

instances adhere to the class definition, a quite accurate list (manually checked, including some cases, such as the Azores or the Isles of Man, which are arguable).

3 Overview of the Application

The application[6] realizes a holistic approach to Navigational Knowledge Engineering, as it combines navigational features with knowledge engineering capabilities. It is implemented in Java based on the Google Web Toolkit[7] and is made up of highly configurable and extensible Spring components, so that it can be customized and tailored for certain data sets. The default implementation of the component interfaces is held generic and works on arbitrary SPARQL endpoints with RDFS-Reasoning capabalities[8]. The full text search (Figure 1 left side) is based on a configurable SPARQL template engine. For learning OWL class expressions, DL-Learner[9][3] is used. [1] describes the underlying technique of machine learning on large knowledge bases and contains performance measurements (especially on DBpedia) with acceptable speed for a web scenario.

To help users understand the meaning of learned class expression, labels and comments are displayed in a tooltip, when hovering over a named class or property. Advanced users or ontology editors can also manually alter the class expression by selecting suggested classes, which are either more special or more general than the currently learned example. Whenever the learned class expression changes an additional reasoner is queried and shows related concepts from the formerly saved class expressions, which are either sub-, super-, or sibling classes. All saved classes can be browsed and loaded by all users to further refine searches. If all classes are exported in bulk, they form a class hierarchy, which can be utilized as additional schema for browsing or as input for a team of ontology engineers.

At the time of writing, we configured the Web demo for DBpedia and a Linguistic data set, but plan to increase the number of available knowledge bases.

References

1. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *IJSWIS*, 5(2):25–48, 2009.
2. Jens Lehmann, Chris Bizer, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.
3. Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning journal*, 78(1-2):203–250, 2010.

[6] source code available at `http://nlp2rdf.googlecode.com`

[7] `http://code.google.com/webtoolkit`

[8] our local mirror of DBpedia used in the demo is Virtuoso based `http://virtuoso.openlinksw.com/`

[9] `http://dllearner.org`

Automated Mapping Generation for Converting Databases into Linked Data

Simeon Polfliet

Ensimag engineering school
Grenoble Institute of Technology (INPG)
Grenoble, France
simeon.polfliet@ensimag.imag.fr

Ryutaro Ichise

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

Abstract. Most of the data on the Web is stored in relational databases. In order to make the Semantic Web grow we need to provide easy-to-use tools to convert those databases into linked data, so that even people with little knowledge of the semantic web can use them. Some programs able to convert relational databases into RDF files have been developed, but the user still has to link manually the database attribute names to existing ontology properties and this generated "linked data" is not actually linked with external relevant data. We propose here a method to associate automatically attribute names to existing ontology entities in order to complete the automation of the conversion of databases. We also present a way - rather basic, but with low error rate - to add links automatically to relevant data from other data sets.

Keywords: Database, Linked Data, Semantic Integration, Semantic Web

1 Introduction

Even though significant research and development efforts have been made, the achievement of the vision of the Semantic Web remains remote. The amount of data on the Semantic Web remains marginal in comparison with the traditional Web. The importance of revealing relational data and making it available as RDF and as Linked Data[1] has been already acknowledged. Most notably, Virtuoso RDF views[4] and D2RQ[2] are production-ready tools for generating RDF representations from relational database contents. But the main restriction to their deployment is the complexity of generating a mapping, which is the last non-automated part of these programs.

In this paper, we will present a method to generate automatically the mapping between attribute names and existing ontology entities, completed with a method to add links automatically to external data. Then, we will present the application of this method on relational databases applied on the D2RQ Mapping system and the D2R Server[3], and the tests and results on different kind of relational databases.

A presentation of our software AuReLi (**Au**tomatic **Re**lational Database to **Li**nked Data Converter) can be found at `http://ri-www.nii.ac.jp/AuReLi/`

2 Method

In ontology matching, there are three types of methods to compare two entities: string-based, structure-based and knowledge-based methods. In the present problem, there

is on one side a database and on the other side we have several ontology descriptions. Thus, structure-based method are not relevant here. We are seeking to compare the name of an attribute in a database with the name of an ontology property. These names can be composed by one or several words: the first step is to split the name into a set of words in order to compare the words of each set. The success of the matching depends on the correctness of the word decomposition. The words composing names of ontology properties and of attributes in a database are usually either separated by special characters, for instance `product_name`, or by a change of case, e.g. `ProductName`. As sometimes it is not the case, we completed this simple splitting method with a method based on the presence of the words in a dictionary such as the WordNet dictionary[1] used here. After doing the previous splitting, it is necessary to check if the resulting words exist in the dictionary. If that is not the case, then we try to split it into words that are in the dictionary. However, because it is possible that the word is not in the dictionary but some part of it is, we will only keep the result if all the decomposed parts are present in the dictionary. With this method, even `productname` will be correctly split. The second step is to compare the resulting set of words of the attribute name with the sets of words of all the ontology entities, and then return the best match. In order to compare the words, we use string-based similarity measures[2], especially Jaro-Winkler, and WordNet similarity measures[3]: Lin[5] and Wu and Palmer[6] measures. We use WordNet measures if the words exist in the WordNet dictionary, otherwise we use the string-based ones.

Once the mapping is done, in order to have true linked data, we want to add links to relevant data. The idea is to make a SPARQL query on a given data set. If you know the target data and its ontology entities, you can specifically build SPARQL queries for this data set to get links. But here, in a more general setting, we do not have this information. However, there is a property common to most of the data sets: rdfs:label. Even better, this property is especially good because it is usually at the same time short and clearly defining the data. Therefore, if the rdfs:label property was correctly set on your data, the SPARQL query based on this property should not return wrong links and has good chances to find a result if there is a related data in the target data set.

3 Implementation

We produced a reusable Java library and used the D2RQ Map and the D2R Server[3] as a basis to implement and test our method. A Java graphical user interface was produced for the mapping generation, in order to simplify its use as much as possible. First, the user has to define the parameters to connect to the relational database, and to give to the program the ontology descriptions he wants to use, as shown in Fig. 1. We already provide some of the most common generic ontology descriptions along with some more specialized ones, but the user can add any other ontology by providing a file with its OWL definition. Then, the program generates the mapping of the table and attribute names with the ontology entities. It presents the resulting mapping to

[1] Princeton University: WordNet, Version 3.0:
`http://wordnet.princeton.edu/wordnet/download/`
[2] S. Chapman: SimMetrics Java library:
`http://www.dcs.shef.ac.uk/~sam/simmetrics.html`
[3] D. Hope: Java WordNet::Similarity:
`http://www.cogs.susx.ac.uk/users/drh21/`

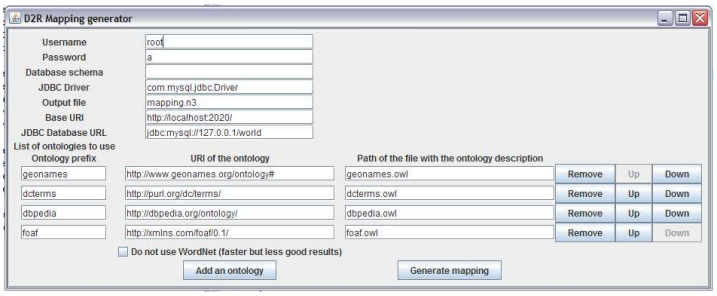


Fig. 1. Mapping generation graphical interface

the user so that he can check and make changes if necessary. It also allows the user to choose which attributes to use as labels for the rdfs:label property.

The D2R Server was also modified to add links automatically in the generated data. If the feature is activated, it makes a SPARQL query on DBpedia for each request of the user and add the link to the data if there was a result. We used DBpedia because it is currently one of the biggest and the most general linked database.

4 Test and Results

Five databases from different sources, with different size and about different topics were used for the tests: Northwind[4], World and Sakila[5], Automobile[6], World Development Indicator[7]

There are approximately three hundred attributes in those five databases: after a manual check of the mappings, 79,66% of the attributes were correctly mapped. The wrong mappings are explained by the fact that some attributes were too specific and consequently could not match any existing ontology property in the ontology descriptions used in the experiment. Another limit is the use of acronyms or short abbreviations, which did not produce a correct mapping either. The generation time was around one minute for each database. The mapping generated automatically can be seen in Fig. 2 for the World database. On the left and the middle are the table names and the attribute names, on the right are the matched ontology entities. We can observe for instance that the attribute *GNPOld* do not have good corresponding property and thus is mapped with *foaf:OnlineAccount* which is obviously irrelevant. But on the other hand, *Percentage* becomes *dbpedia:part*, which is quite good since a percentage is a part of something. This matching is due to WordNet because it would not have been found by a string-based similarity measure.

For the server, the use of the feature to automatically add links is slightly slowing down each request of the user because it needs the answer of the SPARQL query. It

[4] Example database for the Microsoft SQL Server:
`http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46a0-8da2-eebc53a68034`

[5] Two example database from the MySQL website:
`http://dev.mysql.com/doc/index-other.html`

[6] Data set from the UCI Machine Learning Repository:
`http://archive.ics.uci.edu/ml/datasets.html`

[7] database from the World Bank Data Catalog:
`http://data.worldbank.org/data-catalog`

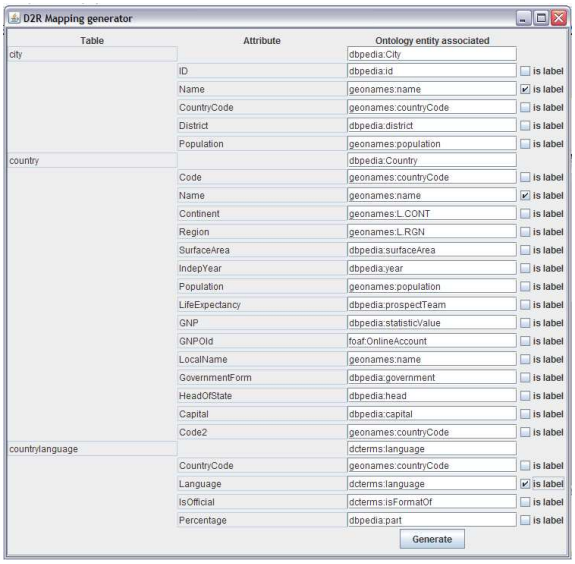


Fig. 2. Mapping generation result for the World database

becomes problematic if the external data set is slow or do not answer to the query. The results on the rdfs:label property on DBpedia are usually good, providing the labels in the mapping are correct. The principal case where the added links are wrong is in the case of homonyms, e.g. cities such as London, England and London, Canada.

5 Conclusion

The automatic mapping generation is a difficult problem which renders almost impossible the automatic production of a 100% correct mapping. Nevertheless, even if the user still needs some knowledge of the Semantic Web, we managed to simplify the process with a user-friendly interface where the user only has to check the correctness of the proposed mapping. The automatic addition of links in the generated RDF is simple and functional, and can easily be extended to add a greater variety of links.

References

1. T. Berners-Lee. Design issues: Linked data, 2006. `http://www.w3.org/DesignIssues/LinkedData.html`
2. C. Bizer and A. Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In ISWC2004 (posters), November 2004.
3. C. Bizer and R. Cyganiak: D2R Server, Version 0.7 `http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/`
4. O. Erling and I. Mikhailov. RDF support in the Virtuoso DBMS. In Proceedings of the 1st Conference on Social Semantic Web, volume P-113 of GI-Edition - Lecture Notes in Informatics (LNI), ISSN 1617-5468. Bonner Kollen Verlag, September 2007.
5. Lin, D. An information-theoretic definition of similarity. In Proceedings of the International Conference on Machine Learning, 1998
6. Z. Wu and M. Palmer. Verb semantics and lexical selection. In 32nd Annual Meeting of the Association for Computational Linguistics, 133-138, 1994

Avalanche: Putting the Spirit of the Web back into Semantic Web Querying

Cosmin Basca and Abraham Bernstein

DDIS, Department of Informatics, University of Zurich, Zurich, Switzerland
{lastname}@ifi.uzh.ch

Abstract. Traditionally Semantic Web applications either included a web crawler or relied on external services to gain access to the Web of Data. Recent efforts, have enabled applications to query the entire Semantic Web for up-to-date results. Such approaches are based on either centralized indexing of semantically annotated metadata or link traversal and URI dereferencing as in the case of Linked Open Data. They pose a number of limiting assumptions, thus breaking the openness principle of the Web. In this demo we present a novel technique called Avalanche, designed to allow a data surfer to query the Semantic Web transparently. The technique makes no prior assumptions about data distribution. Specifically, Avalanche can perform "live" queries over the Web of Data. First, it gets on-line statistical information about the data distribution, as well as bandwidth availability. Then, it plans and executes the query in a distributed manner trying to quickly provide first answers.

1 Introduction

With the rapid growth of the Web of Data, unexplored avenues for application development are emerging. While some application designs include a Semantic Web (SW) data crawler, others rely on services that facilitate access to the Web of Data (WoD) either through the SPARQL protocol or various API's (i.e. Sindice or Swoogle). As the mass of data continues to grow – currently LOD [2] accounts for 4.7 billion triples – the scalability factor will give raise to a new set of challenges. Marginally addressed today is the question: How to query the Web of Data on-demand, without hindering the flexible openness principle of the Web – seen as the ability to query independent un-cooperative semantic databases, not controlling their distribution, their availability or having to adhere to fixed publishing guidelines (i.e. LOD). Currently, some approaches maintain a global index over all SW data, striving to deliver up-to date results. They become expensive as the quantity of data grows and demand rises. Derived from traditional distributed database systems, they offer high performance, but break the flexibility and openness principle by assuming perfect knowledge about participating data. Instance level federation (i.e. subjects are distributed to hosts according to a known scheme) as in the case of SemWIQ [5] relaxes the centralization constraints by increasing the flexibility of the system while still holding some constraints over data distribution. As proposed by Hartig et al. [4] other

approaches are based on the principle of link traversal and URI dereferencing. These algorithms are driven by LOD principles, offering high flexibility at the cost of potentially expensive query processing (due to network latency) and the impossibility to issue certain types of queries as pointed out by the authors.

This demo proposes to address the issue of preserving the openness and flexibility of WoD while being able to query it "live". Our motivation lies in the belief that such flexibility is paramount for future development of the Semantic Web – as it was for the WWW. Considering the Web's uncontrollable nature, AVALANCHE [1] strives to find the *first K* results as fast as possible.

2 Avalanche — System Design and Implementation

The system consists of six major components working together in a parallelized pipeline: the AVALANCHE *endpoints Web Directory* or *Search Engine*, the *Statistics Requester*, the *Plan Generator*, *Plan Executor* instances, *Plan Materializer* instances and the *Query Stopper* component as seen in Figure 1.

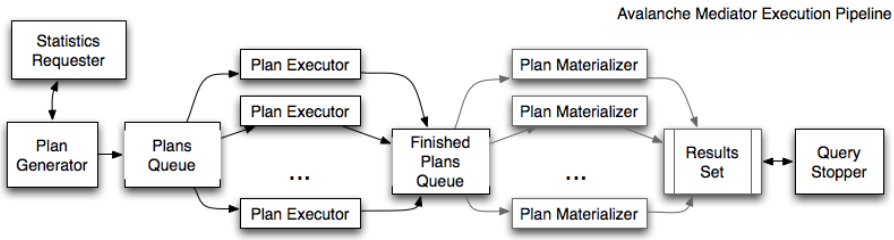


Fig. 1. The Avalanche execution pipeline

The algorithm is comprised of two steps: the *Query Preprocessing* step and the parallel *Query Execution* step. During *Query Preprocessing*, participating hosts are selected via means of a lightweight endpoint-schema inverted index. Ontological prefix (the shorthand notation of the schema – i.e. `foaf`) and schema invariants (i.e. predicates, classes, labels, etc) are appropriate candidate entries to index. After query parsing, this information is immediately available and used to quickly trim down the number of potential endpoints. Further all selected AVALANCHE endpoints are queried for unbounded variables instance counts.

Query Execution follows: first the query is broken down into the superset of all **molecules**, where a molecule is a subgraph of the overall query graph. A combination of minimally overlapping molecules, covering the query graph, is referred to as a **solution**. Binding all molecules in a given solution to physical hosts that may resolve them, transforms a solution into a **plan**. It is the *Plan Generator's* job to issue plans for execution, as soon as assembled. An emergent challenge from preserving the openness of the query process and the flexibility of semantic data publishing, is denoted by the exponential complexity class of the plan composition space. It is thus crucially important to issue "good" plans first (plans that aid in finding the answers fast) while pruning and stopping undesired plans as soon as possible. To this end, the molecule space is trimmed empirically and plans are generated ordered given their *objective* function. The objective of a plan is represented as the ratio between the *utility* (number of results produced)

and the *cost* of execution (time spent to execute subqueries, distributed joins and bandwidth consumption). Consider for example the following query over RDF data describing publications:

```
SELECT ?name ?title WHERE {
?paper akt:has-author ?author; akt:has-author ?jim; akt:has-title ?title.
?author akt:full-name ?name. ?jim akt:full-name "James A. Hendler". }
```

AVLANACHE's goal is to return the list of all authors that wrote papers with "James A. Hendler" (bound variable) and their titles, given that the required data is spread with an unknown distribution over several independent hosts.

At a given moment during the execution of a plan, a *Plan Executor* may find itself in the following state: molecule `M1` (see Figure 2) was reported to be highly selective on host A, while the remainder of the query (molecule `M2`) is a low selectivity molecule on host B. Given that bandwidth and latency cost can be high, partial results [1] are not sent to one central site to be joined. Instead we start the execution with the highly selective molecule `M1` (host A) and then filter results on host B by sending over results from host A as in Figure 2.

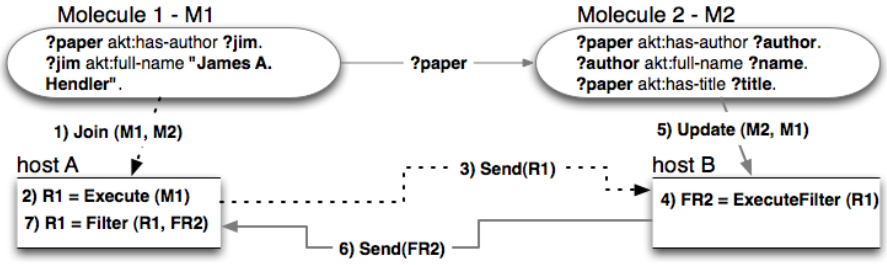


Fig. 2. Distributed Join and Update operations for a Simple Plan

Similarly the *Plan Materializer*, materializes out-of-order finished plans, by merging partial results and fetching their actual string representations. Since we have no control over the distribution and availability of RDF data and data providers (SPARQL endpoints), getting a complete answer to the query is an unreasonable assumption. Instead the *Query Stopper* monitors for the following *stopping conditions*: a global timeout, a *first K* results policy – all execution stops as soon as K (unique) results are returned to the caller – and finally, to avoid waiting for the timeout when the number of results is $\ll$ K we measure relative result-saturation (i.e. we stop at 90% saturation).

The cost of expensive distributed n-way joins can be reduced via bloom-joins [6]. We extend the objective function with a qualitative join estimation, as described in [3] and since constructing *bloom filters* for large sets is expensive, we only consider the highly selective triple patterns (a threshold set empirically).

[1] It is important to note that to execute plans, hosts will need to share a common id space – a given in Semantic Web via URIs. Naturally, using RDF strings can be prohibitively expensive. To limit bandwidth requirements we, chose to employ a single global id space in the form of the `SHA` family of hash functions on the URIs.

3 Preliminary Evaluation

To demonstrate the AVALANCHE algorithm and evaluate its performance we conducted a preliminary evaluation on the combined IEEE, ACM and DBLP LOD datasets, totaling 35 million triples. We distributed the datasets over a five-node cluster, splitting by dataset and chronological order (i.e. ACM articles till 2003 on host A). Each machine had 2GB RAM and an Intel Core 2 Duo E8500 @ 3.16GHz. AVALANCHE was able to successfully execute query plans and retrieve many up-to-date results without having any prior knowledge of the data distribution. Furthermore, we observed that different objective functions have a significant influence on the outcome and should play a critical role when deployed on the Semantic Web. *The demo*[2] *will mirror this setup and let people pose arbitrary queries live.*

4 Conclusion

In order to preserve the openness of the Web of Data, one has to make shallow or no prior assumptions to data distribution and availability. Also given the uncontrollable nature of the Web, a **first K** results policy makes sense. For this purpose we developed and present AVALANCHE a novel approach for querying the Web of Data that: (1) makes no assumptions about data distribution, availability, or partitioning, (2) provides up-to-date results, and (3) is flexible as it makes no assumption about the structure of participating triple stores. To our knowledge, AVALANCHE [3] is the first Semantic Web query system that makes no assumptions about the data distribution whatsoever. Whilst, it is a first prototype with a number of drawbacks it represents a first important step towards bringing the spirit of the Web back to triple-stores — a major condition to fulfill the vision of a truly global and open Semantic Web.

References

1. C. Basca and A. Bernstein. Avalanche: Putting the Spirit of the Web back into Semantic Web Querying. In *Proceedings of the 6th International Workshop on Scalable Semantic Web Knowledge Base Systems*, November 2010.
2. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - The story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.
3. A. Broder, M. Mitzenmacher, and A. B. I. M. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.
4. O. Hartig, C. Bizer, and J.-C. Freytag. Executing SPARQL queries over the Web of linked data. In *8th International Semantic Web Conference (ISWC)*, 2009.
5. A. Langegger, W. Wöß, and M. Blöchl. A Semantic Web middleware for virtual data integration on the web. In *5th European Semantic Web Conference*, 2008.
6. S. Ramesh, O. Papapetrou, and W. Siberski. Optimizing distributed joins with bloom filters. In *ICDCIT '08: Proceedings of the 5th International Conference on Distributed Computing and Internet Technology*, 2009.

[2] Video available at: http://www.ifi.uzh.ch/ddis/fileadmin/basca/avalanche-demo.mov

[3] This work was partially supported by the Swiss National Science Foundation under contract number 200021-118000

Displaying email-related contextual information using Contextify

Gregor Leban, Marko Grobelnik

Jožef Stefan Institute, Ljubljana, Slovenia
gregor.leban@ijs.si, marko.grobelnik@ijs.si

Abstract. Contextify is a tool for maximizing user productivity by showing email-related contextual information. The contextual information is determined based on the currently selected email and includes related emails, people, attachments and web links. This content is displayed in a sidebar in Microsoft Outlook and in a special dialog that can display an extended context.

Keywords: email, optimization, productivity, visualization

1 Introduction

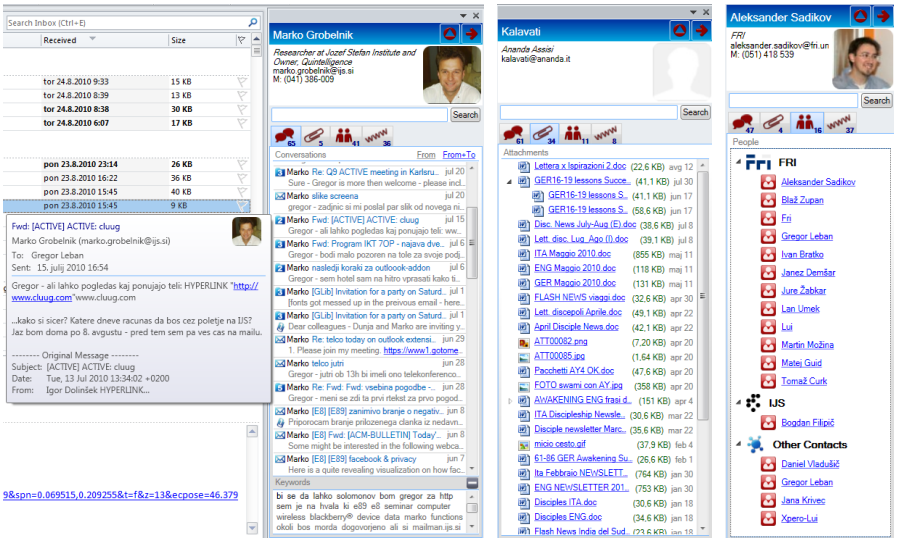
Despite the popularity of numerous web services, emails still play a crucial role in todays information exchange. It is very common for people to receive tens or even hundreds of emails per day. In this sea of information it is often difficult to stay organized and to find the right information when one needs it.

To help people perform mail-related tasks faster and with greater ease we developed Contextify. Contextify is an add-on for Microsoft Outlook. Its goal is to maximize users productivity by unobtrusively finding and showing the relevant contextual information for the currently selected email. The main functions of the add-on are displayed in two windows which will be described next.

2 Contextify sidebar

One way how Contextify displays the relevant contextual information is in a sidebar of Microsoft Outlook. The goal of the sidebar is to show important information related to the sender of an email being currently selected and shown within Outlook. This information is mainly gathered from past emails from this person but also from various online services such as Facebook, LinkedIn and Twitter.

An example of the sidebar is shown in Figure 1. The top of the sidebar displays the person's photo together with the relevant personal information collected from Outlook contacts and online social services (e.g. Facebook, LinkedIn). Other contextual information is organized into several tabs. In the first tab we show the list of recent emails from this person together with several email details (Figure 1.a). For emails that are a part of a thread, the whole thread can be



(a) (b) (c)

Fig. 1. Information displayed in different tabs of the Contextify sidebar. The Email tab (a) shows recent emails from the selected contact, the Attachments tab (b) shows the exchanged files, and the People tab (c) shows the people included in the conversations.

brought into view by clicking on the email. Below the list of emails there is also a tag cloud of keywords that best describe the content of these emails. The second tab displays the list of exchanged attachments together with various details (Figure 1.b). Attachments that were exchanged several times, probably because being updated, are grouped together. They can be opened by clicking on the filename. The third tab lists the people who are participating in these emails (Figure 1.c) while the fourth tab displays the web links that were exchanged in the emails. The sidebar also provides searching capabilities where the context is determined based on the search terms and not the sender of the selected email. The found search terms are also highlighted for easier recognition. In the future we'll add a tab with a summary with persons appearance on various social services.

3 Contextify dialog

An expanded context for the selected email can be displayed in the Contextify dialog. The goal of the dialog is to provide a visual display of the contextual information together with additional highlighting and filtering options. The context in this dialog consists of those emails where the participants (the sender and all the recipients) sufficiently match the participants in the currently selected email. The context is defined here as all emails sent to a similar social group.

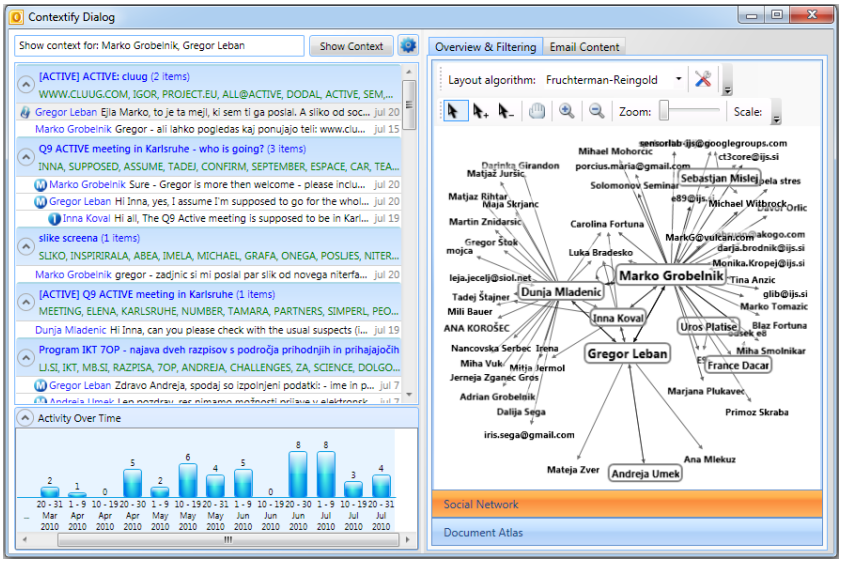


Fig. 2. Contextify dialog displays an expanded version of the context.

Such a definition of a context helps the user to see emails that are related to a particular aspect of his/her life.

An example of this dialog is shown in Figure 2. All emails that belong to the context are grouped into threads and displayed in the top left part of the dialog. Below the topic of each thread is a list of keywords that best describe the content of the thread. The bottom left part of the dialog shows a visualization of email activity for the computed context over time – each bar shows the number of emails that were received in a particular time period. By selecting or unselecting specific bars the users can display only emails from a specific time period. The right side of the dialog shows the social network for the participants in the context. There is a directed edge between persons A and B if there is at least one email sent from A to B. The font size for nodes depends on how often the person is present in the emails this helps quickly identifying the most relevant participants. Selecting a person in the graph also highlights emails sent by this person in the list of emails.

4 Contact management

Accessing and modifying information about contacts can be done in the Contact management dialog. Here, we can manually group different email addresses that represent the same person. In this way, when showing contextual information, emails from all person's email addresses will be displayed. By clicking the "Import contact information" button additional information about contacts can

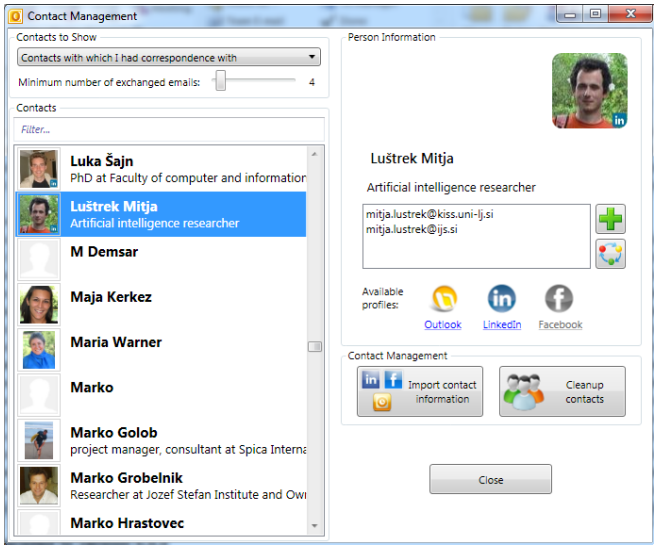


Fig. 3. Contact management dialog displays information about the contacts collected from emails. Additional information about the contacts can be imported from social web-services such as Facebook and LinkedIn.

be automatically imported from Facebook, LinkedIn and Outlook contacts. The "Cleanup contacts" button provides a fast way to tidy-up contact names and to merge contacts which are likely to represent the same person. An example of the Contact management dialog is shown in Figure 3.

5 Conclusion and future work

The goal of Contextify is to determine and show the relevant contextual information that should help the user to more efficiently perform his email-related tasks. We are currently working on text-mining algorithms to provide additional functionality such as email summarization, automatic categorization of emails into folders, improved extraction of information from email contents and more advanced contact management.

A video demonstration of the add-on can be seen at http://www.youtube.com/watch?v=hYpxhUvYM10 (lower resolution) and at http://www.screencast.com/t/ODg5MzM1YWEt (higher resolution).

6 Acknowledgements

This work was supported by the Slovenian Research Agency, the European Social Fund, the IST Programme of the EC under PASCAL2 (IST-NoE-216886) and ACTIVE (IST-2008-215040).

KiWi - A Platform for building Semantic Social Media Applications

Thomas Kurz, Sebastian Schaffert, Tobias Bürger, Stephanie Stroka, Rolf Sint, Mihai Radulescu and Szabolcs Grünwald

Salzburg Research Forschungsgesellschaft
Jakob Haringer Str. 5/3, 5020 Salzburg, Austria
firstname.lastname@salzburgresearch.at

Abstract. The combination of semantic technologies and social software has become more and more popular in the last few years as can be seen by the emergence of Semantic Wikis or the popularity of vocabularies such as FOAF or SIOC. The KiWi project is based upon these principles and offers features required for Social Media applications such as versioning, (semantic) tagging, rich text editing, easy linking, rating and commenting, as well as advanced "smart" services such as recommendation, rule-based reasoning, information extraction, intelligent search and querying, a sophisticated social reputation system, vocabulary management, and rich visualization. KiWi can be used both, as a platform for building custom Semantic Media applications, and as a Semantic Social Index, integrating content and data from a variety of different sources, e.g. Wikis, blogs and content management systems in an enterprise internet. Third-party applications can access the KiWi System using simple-to-use web services. The demo presents the whole functionality of the Open Source development platform KiWi in its final version within one integrated project management scenario. Furthermore it shows different KiWi-based Social Media projects to illustrate its various fields of application.

1 Introduction

Wikis are Web-based applications which allow all users to edit content online. In the same sense, the term "Wiki" refers to a new philosophy of working with web content with the principles of everyone contributing, ease of use, easy linking of information, versioning and web-based media. Semantic Wikis (e.g. Semantic MediaWiki[1] or IkeWiki[2]), which have been developed in the research community in the last years [1,2], combine this philosophy with the intelligence and methods of the Semantic Web. This philosophy holds for other applications as well: Most other social software systems such as blogs, photo sharing sites or social networking platforms share the same basic principles. To leverage on these principles and the power of semantic technologies, the EU-funded project KiWi

[1] Semantic MediaWiki: http://semantic-mediawiki.org
[2] IkeWiki: http://ikewiki.salzburgresearch.at

(Knowledge in a Wiki) builds a generic framework for Semantic Social Media. The demo presents a number of KiWi-based applications (e.g., an artwork portal, an idea management application, a community-based news portal etc.) and demonstrates the added value of the KIWI framework for these applications.

2 The KIWI Framework

The KIWI framework follows a service oriented architecture[3] with a small core and an extension mechanism for developing application-specific actions and program logic on top of it. The core services are usable from any extension and generic jQuery widgets communicating over RESTfull webservices and thus its functionality can easily be reused or adapted by application developers. The KiWi core involves a fully versioned triplestore, a full text and metadata search index (based on Apache SOLR[4]) and services to manage content, users and components.
KiWi follows the resource concept, meaning that everything such as a wiki page, person, tag, ontology class, etc. is a resource itself, a so called ContentItem. A ContentItem can be extended with RDF Relations via a facading mechanism to fulfill specific requirements. The system uses popular ontologies such as FOAF, HGTags, IPTC Newscodes and it is possible to import self-created ontologies as well. The KiWi framework builds heavily on Java Enterprise Edition[5] (Java EE 5) and JBoss Seam[6] (currently version 2.1).

3 Enabling Technologies

3.1 Information Extraction

As described in [3], KiWi combines semantic annotations directly with the text content of the ContentItems and provides advanced user interfaces supporting the annotation process with the help of suggestions coming from an information extraction component. The service uses natural language processing and machine learning algorithms to provide suggestions for annotations.

3.2 Reasoning

KiWi offers a rule-based inconsistency tolerant reasoning that can be explained to users and that also allows for efficient knowledge base updates by the means of reason maintenance, as described in [4]. The reasoner is able to run programs implemented in sKWRL, a simple KiWi rule language.

[3] Link to KiWi Architecture:
`http://www.kiwi-community.eu/display/DOC/KiWi+Architecture`
[4] Apache SOLR: `http://lucene.apache.org/solr/`
[5] Java Enterprise Edition:
`http://www.oracle.com/technetwork/java/javaee/overview/index.html`
[6] JBoss Seam:`http://seamframework.org/`

3.3 Semantic Search

KiWi includes two search engines, both based on a SOLR search index. The first one allows searching on text and RDF metadata as well as facetting on authors, tags, types, RDF literal properties and RDF object properties. Furthermore it is possible to personalize search. The second search engine is able to interpret KWQL queries [5]. KWQL is a rule-based query language based on a label-keyword query paradigm. KWQL allows rich combined queries of full text, document structure, and informal to formal semantic annotations. In addition KiWi provides visKWQL[6], a visual interface for the KWQL language aimed at supporting users in the query construction process.

3.4 Personalization

In KiWi there are several recommender services that use personal activities to enhance content recommendations. As described in [7], the recommenders use traditional tag-based retrieval, external factors such as tag popularity, tag representativeness and the affinity between users and tags. It is also possible to personalize the search by using users personal tagcloud and users personal interests extracted from his or her published content.

3.5 Content Versatility

As described in [8], every piece of information is a combination of human-readable content and associated meta-data, and the same piece of information can be presented to the user in many different forms, even in parallel: as a wiki page, as a blog post, as a comment to a blog, as a photo, or even in a bubble in a map-based application. The decision how the information is displayed is taken based on the context of the content and the user. This is what we call "Content Versatility". In the demo we will show the same content (a meeting in our case) in the KiWi wiki application as well as in TagIt, a KiWi-based application for map based content retrieval. We will also show that different services (e.g. search services) and widgets (e.g. recommender widgets) can be reused by different applications.

4 Demo Outline

Since the last demonstration at European Semantic Web Conference 2009, the project has made considerable progress, which will be presented in this demo. It shows the exploitation of the KiWi system within a project management scenario in which the system is used by a company as a Semantic Wiki to handle its project management. The Wiki is used to manage users, project pages, meetings, meeting minutes, ideas etc. The storyline followed in the demo defines a simple workflow, i.e., how a project can be collaboratively created, structured and edited in KiWi using its enabling technologies. Furthermore it shows how

existing content can versatilely be searched, integrated and displayed. To demonstrate the ability to implement different types of applications on top of KiWi as framework a number of other KiWi-based applications will be presented.

5 Conclusion

The potential of KiWi for building semantic social applications has been demonstrated in several projects that build upon its framework. This includes the idea management software Ideator[9] or the artwork portal ArtAround[7]. Further information about the KiWi project as well as source code and developer support can be found at `http://www.kiwi-community.eu` and the showcase of the current KiWi version at `http://showcase.kiwi-project.eu`.

Acknowledgements *The research leading to these results is part of the project "KiWi - Knowledge in a Wiki" and has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement No. 211932. The authors would like to express their gratitude to all other KiWi developers, particularly Klara Weiand, Fred Durao, Jakub Kotowski and Marek Schmidt.*

References

1. Völkel, M., Schaffert, S., eds.: 1st Workshop "From Wiki to Semantics"(SemWiki'06) – colocated with ESWC'06, Budva, Montenegro, 2006.
2. Lange, C., Reutelshöfer, J., Schaffert, S., Skaf–Molli, H., eds.: Fifth Workshop "Semantic Wikis – Linking Data and People" (SemWiki'10) – colocated with ESWC'10,Hersonissos, Crete, Greece, 2010.
3. Schmidt, M., Smrž, P.: Annotation component for a Semantic Wiki, Fifth Workshop "Semantic Wikis – Linking Data and People", Hersonissos, Greece, 2010
4. Kotowski, J. and Bry, F.: A Perfect Match for Reasoning, Explanation, and Reason Maintenance: OWL 2 RL and Semantic Wikis, Fifth Workshop "Semantic Wikis – Linking Data and People", Hersonissos, Greece, 2010.
5. Bry, F., Weiand, K.: Flavors of KWQL, a Keyword Query Language for a Semantic Wiki, Proceedings of the 36th Conference on Current Trends in Theory and Practice of Computer Science, Czech Republic, 2010.
6. Hartl, A., Weiland, K., Bry, F.: visKQWL, a visual renderer for a semantic web query language, Proceedings of the 19th international conference on World wide web (demo), North Carolina, USA, 2010.
7. Durao, F., Dolog, P.: Analysis of Tag-Based Recommendation Performance for a Semantic Wiki, Fourth Workshop "Semantic Wikis – Linking Data and People", Hersonissos, Greece, 2009.
8. Schaffert, S., Eder, J., Grünwald, S., Kurz, T., Radulescu, M.: KiWi - A Platform for Semantic Social Software, Demo on 6th European Semantic Web Conference, Hersonissos, Greece, 2009.
9. Sint, R., Markus, M., Schaffert, S., Kurz, T.: Ideator - a collaborative enterprise idea management tool powered by KiWi, Fifth Workshop "Semantic Wikis – Linking Data and People", Hersonissos, Greece, 2010

[7] ArtAround: `www.artaround.at`

Enterprise Data Classification Using Semantic Web Technologies

David Ben-David[1], Tamar Domany[2], and Abigail Tarem[2]

[1] Technion – Israel Institute of Technology, Haifa 32000, Israel,
davidbd@cs.technion.ac.il
[2] IBM Research – Haifa, University Campus, Haifa 31905, Israel,
{tamar,abigailt}@il.ibm.com

Abstract. Organizations today collect and store large amounts of data in various formats and locations, however they are sometimes required to locate all instances of a certain type of data. Data classification enables efficient retrieval of information when needed. This work presents a reference implementation for enterprise data classification using Semantic Web technologies. We demonstrate automatic discovery and classification of Personally Identifiable Information (PII) in relational databases, using a classification model in RDF/OWL describing the elements to discover and classify. At the end of the process the results are also stored in RDF, enabling simple navigation between the input model and the findings in different databases.
Recorded demo link: https://www.research.ibm.com/haifa/info/demos/piidiscovery_full.htm

Keywords: Semantic Techniques, RDF, Classification, modeling, NeON, RelationalOWL

1 Introduction

Organizations today collect and store large amounts of data in various formats and locations. When an organization is required to meet certain legal or regulatory requirements, for instance to comply with regulations or perform discovery during civil litigation, it needs to find all the places where the required data is located. Data discovery and classification is about finding and marking enterprise data in a way that enables quick and efficient retrieval of the relevant information when needed. Most existing approaches either require re-classification of the data each time the organization's policies change, can only be applied to a single data type or format, or only identify predefined sets of known fields.

In this work we demonstrate the concept of enterprise data classification using Semantic Web technologies described in [2]. The goal of the solution is to provide organizations with a tool that automatically locates and annotates valuable information, provides manageable results and enables quick and easy access

to the data when needed. For example, if in order to comply with a privacy regulation an organization is required to mask all Social Security Numbers (SSN), all the occurrences of SSN must be found.

This reference implementation demonstrates the automatic discovery and classification of Personally Identifiable Information (PII) stored in relational databases. The classification process starts with creating a model described using the Resource Description Framework (RDF), containing the entities to discover and classify as well as additional information that can help the discovery process (e.g., type and format); this is referred to as a *classification model*. In this demo we used a model representing PII, but any model that follows the meta-model described in [2] can be used. The result of the classification process is a set of RDF triples linking between entities in the classification model and locations in the data stores, in this case database tables and columns. Using RDF to define the classification model makes it easy to expand, merge and combine existing models and generate new models for different purposes. The fact that the classification results are also represented in RDF that follow the same shema as the classification model, enables us to unify the results from different classifiers, navigate easily between the model entities and the data sources (thanks of the use of URIs), annotate, reason, and query the classified data, and more. The classification process is composed of four stages:

1. Creating or loading an existing classification model.
2. Importing database schemas.
3. Discovering and classifying the data according to the classification model, using SPARQL and various classification algorithms.
4. Representing the results in a way that allows navigation between the classification model and the specific columns where the information was found.

A high-level view of this process is depicted in Figure 1.

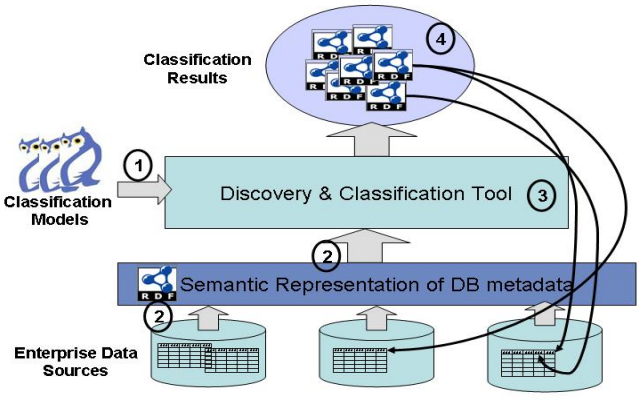


Fig. 1. The classification process

2 Implementation

Our reference implementation is based on the NeOn toolkit [4], an open-source, Eclipse-based ontology engineering environment. In addition we used the Eclipse Data Tools Platform (DTP)[1] to define connections with local and remote databases. We chose RelationalOWL [5] as the basis to create an RDF representation of the database metadata. We also used the Jena framework [3] to access and query the different RDF representations. To those we added a set of "home-made" plug-ins that perform the discovery and integrate between the different components in the system. The discovery component uses the syllabifying techniques described in [2], as well as type checking. Future extensions are planned to include additional linguistic techniques (such as stemming) and the use of sample content to verify the data's format.

Using our classification tool users can create projects, build and edit models, import existing models (in both cases, the models are validated against the metamodel) and import or create database metadata RDF representations. Users can perform the discovery process on any combination of models and databases. The results, as well as the models and database metadata, can be viewed in both a hierarchical view and a graph view, as depicted in Figure 2. Figure 2 shows part of the discovery results (in this case - all table columns in which a first name was discovered).

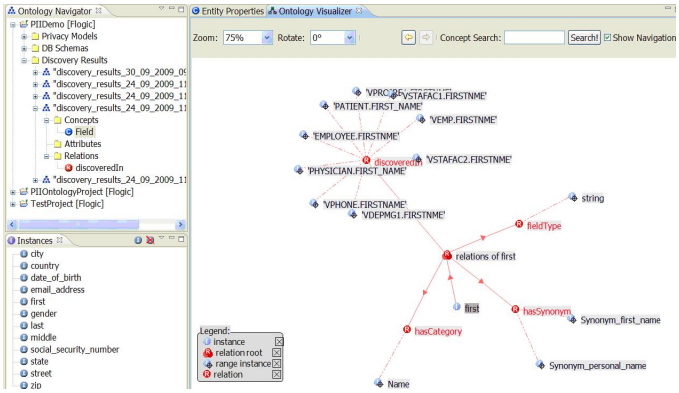


Fig. 2. A partial view of the discovery results in the NeOn-based tool

For the purpose of this demonstration, we execute our classification on two externally available databases: one representing employee records in an organization (taken from the sample database created by the DB2® software installation) and the other representing medical records of patients (taken from "Avitek Medical Records Development Tutorials" by BEA Systems, Inc. [3]).

[3] `http://download.oracle.com/docs/cd/E13222_01/wls/docs100/medrec_tutorials/index.html`

As noted previously, we use RDF to represent the discovery results, making it possible to navigate from any node in the result back to both the classification field in the model, and to the data field (column) in the database representation. This easy navigation allows verifying the classification results, refining them (adding or removing triples), and enriching the model so it is more accurate in subsequent runs.

3 Summary

In this demonstration we exhibited the main advantages of our approach. By combining different discovery techniques and extracting most of the search logic to external files, we created a highly flexible and adaptable solution. Using RDF to represent both the ontologies and the results maximizes the modularity and extensibility of the classification input and facilitates easy navigation between the results, the models and the data sources. The ontology can thus serve as a centralized point to manage all valuable information in the organization and enables easy location of all related pieces of data in one click. In addition, all of the information created and used by the system (models, metadata RDF representations, results) can be exposed to existing and evolving Semantic Web tools, such as semantic query languages, reasoning engines and rule languages.

References

1. Eclipse Data Tools Platform (DTP) Project. data sheet, `http://www.eclipse.org/datatools/`
2. Ben-David, D., Domany, T., Tarem, A.: Enterprise Data Classification using Semantic Web Technologies. In: ISWC (2010)
3. Carroll, J.J., Dickinson, I., Dollin, C., Seaborne, D.R.A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations. Tech. rep., HP Laboratories (2003), `http://www.hpl.hp.com/techreports/2003/HPL-2003-146.pdf`
4. Holger, P.H., Studer, L.R., Tran, T.: The NeOn Ontology Engineering Toolkit. In: ISWC (2009), `http://www.aifb.uni-karlsruhe.de/WBS/pha/publications/neon-toolkit.pdf`
5. de Laborda, C.P., Conrad, S.: RelationalOWL: a data and schema representation format based on OWL. In: Conferences in Research and Practice in Information Technology. pp. 89–96 (2005)

The eCloudManager Intelligence Edition
Semantic Technologies for Enterprise Cloud Management

Peter Haase, Tobias Mathäß, Michael Schmidt,
Andreas Eberhart, Ulrich Walther

fluid Operations, D-69190 Walldorf, Germany
firstname.lastname@fluidops.com

Abstract. Enterprise clouds apply the paradigm of cloud computing to enterprise IT infrastructures, with the goal of providing easy, flexible, and scalable access to both computing resources and IT services. Realizing the vision of the fully automated enterprise cloud involves addressing a range of technological challenges. In this demonstration, we show how semantic technologies can help to address the challenges related to intelligent information management in enterprise clouds. In particular, we address the topics of *data integration*, *collaborative documentation and annotation* and *intelligent information access and analytics* and demonstrate solutions that are implemented in the newest addition to our eCloudManager product suite: The Intelligence Edition.

1 Introduction

Cloud computing has emerged as a model in support of "everything-as-a-service" (XaaS). Cloud services have three distinct characteristics that differentiate them from traditional hosting. First, they are sold on demand, typically by the minute or the hour; second, they are elastic – users can have as much or as little of a service as they want at any given time; and third, cloud services are fully managed by the provider [3]. While the paradigm of cloud computing is best known from so called public clouds, its promises have also caused significant interest in the context of running enterprise IT infrastructures as private clouds [2]. A private cloud is a network or a data center that supplies hosted services to a limited number of people, e.g. as an enterprise cloud. Like public clouds, enterprise clouds provide easy, scalable access to computing resources and IT services.

Realizing the vision of the fully automated data center – the enterprise cloud – involves addressing a range of technological challenges, touching the areas of infrastructure management, virtualization technologies, but also distributed and service-oriented computing. In our conference paper [1], we have described the challenges related to intelligent information management in enterprise clouds and discussed how semantic technologies can help to address them. In particular, we have addressed the topics of *data integration*, *documentation and annotation*, and *intelligent information access and analytics*. Summarizing the main contributions, our RDF-based approach to data integration allows us to deal with the highly heterogeneous and changing set of resources encountered in enterprise

data centers. Semantic wikis provide an end-user oriented interface for creating structured and unstructured annotations, supporting the main use cases for documentation and knowlegde management, seamlessly integrating automatically obtained data with user-generated content. This data can be searched, explored, and analyzed without system boundaries, supported by state-of-the-art techniques of semantics-based information access.

This demonstration complements the conference paper with a live demo of the implementation of our solution in the eCloudManager Intelligence Edition[1]. In the remainder, we present a brief solution overview of the eCloudManager, followed by a description of the demonstration scenario.

2 Solution Overview

The eCloudManager Product Suite is a Java-based software solution that is targeted at the management of enterprise cloud environments. The eCloudManager's overall architecture is depicted in Figure 1. The bottom of the figure shows the two dimensions of information relevant to the eCloudManager, namely *Data Center Resources* and *Business Resources*. The data center resources are divided along the IT stack into (i) a *Hardware Layer* that consists of physical storage, network and compute infrastructure, (ii) a *Virtualization Layer* built on top of the hardware layer that is made up of hypervisors with appropriate management capabilities, and finally (iii) the *Application Layer* built on top of the virtualization layer, comprising applications on top of the virtualized resources. These data center resources are complemented by associated business resources, like customer data, hardware catalogs, or related project information.

Built on top of this infrastructure, the eCloudManager comes with four complementary editions for *Infrastructure Management*, *Virtual Landscape Management*, and *Self-Service*. In the demonstration, we will focus on the features of the fourth edition, namely the *Intelligence Edition*, which makes use of innovative semantic technologies to integrate available resources into a semantic store, investigate this data, and collaboratively interact with the integrated data.

At the bottom of the Intelligence Edition is the *Data Integration Layer*, which relies on the concept of so-called *data providers* that extract data from a physical or logical resource, convert it into RDF and integrate the resulting RDF data into the central repository. The central repository where the provider data is stored is settled in the *Data Management Layer*. Technically, it is realized as a Sesame triple store that adheres to a predefined (yet extendable) OWL ontology. In addition to the repository, the layer provides components for search and intelligent, semantics-based information access. A central component in this layer are also semantic wiki pages that are associated with the resources in the repository; they offer an entry point to the eCloudManager users, allowing to add new and complement existing information. The uppermost layer in the Intelligence Edition is the *Presentation Layer*. Located on top of the Data Management Layer, it

[1] The product including additional material such as screencams is available at `http://fluidops.com/eCM_INT.html`

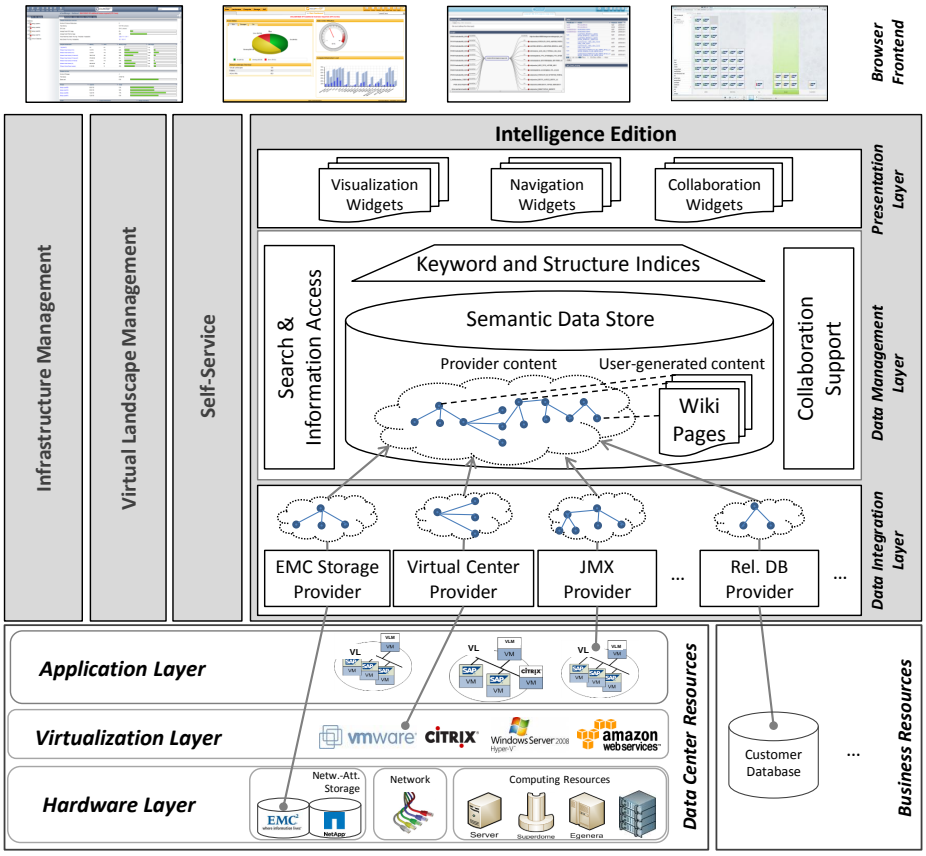


Fig. 1. eCloudManager Architecture

comes with a predefined set of widgets with varying functional focus, e.g. offering support to display wiki pages, visualize the underlying data using charts, navigate through the underlying RDF graph, and collaboratively annotate resources using both semantic annotations as well as free-text documentation.

3 Demonstration Scenario

We now give a brief overview of the demonstration, in which we show how the Intelligence Edition addresses the challenges in the administration of a real enterprise cloud. It is structure along the main features of the system.

Data Integration. Being able to automate data center operations via low level APIs is the prerequisite for achieving the vision of a fully automated enterprise cloud. Many layers play a role in this picture and one is faced with a large set of provider APIs ranging from storage to application levels. In the demo, we will show an enterprise cloud with heterogeneous multi-vendor resources. We show how we use RDF as a data model for integrating semantically heterogeneous information to obtain a unified view on the entire data center, both horizontally – across different product versions and vendors – and vertically – across storage, compute units, network, operating systems, and applications.

Collaborative Documentation and Annotation. In order to have a complete picture, organizational and business aspects need to be added to the technical data. Consider the following examples: The decision whether to place a

Fig. 2. Visual exploration of query results in the Intelligence Edition

workload on a redundant cluster with highly available storage is strongly affected by the service level the system needs to meet, data center planning tools must take expiring warranties of components into account, and having a relatively mild punishment for SLA violations may lead a cloud operator to take a chance and place workloads on less reliable infrastructure. In the demonstration, we will show how administrators can extend the data fed from infrastructure providers by documenting and annotating the respective items. The administrator can e.g attach best practices for error handling to storage resources, connect infrastructure level resources with project and customer information etc.

Intelligent Information Access and Analytics. Efficient management of a data center requires providing data center managers with the information they need to make intelligent, timely and precise decisions. We will demonstrate specific information needs, including the generation of reports about status and utilization of data center resources over time, the visualization of key performance metrics in dashboards, the search for specific resources etc. Many of these information needs require multi-dimensional queries that span across both IT-related and business aspects, and therefore cannot be answered by a single data source alone. As an example, Figure 2 shows an intermediate step in the visual exploration process for customers with gold service level who are affected by the failure of a storage filer. Similar in spirit, we will demonstrate different queries and result visualizations that overcome the borders of data sources.

References

1. Peter Haase, Tobias Mathäß, Michael Schmidt, Andreas Eberhart, and Ulrich Walther. Semantic technologies for enterprise cloud management. In *ISWC*, 2010.
2. Borja Sotomayor, Rubén S. Montero, Ignacio M. Llorente, and Ian Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13(5):14–22, 2009.
3. Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, 2009.

WebProtégé: Supporting the Creation of ICD-11

Sean M. Falconer, Tania Tudorache, Csongor Nyulas, Natalya F. Noy, Mark A. Musen

Stanford Center for Biomedical Informatics Research, Stanford University, US
{sfalc , tudorache, nyulas, noy, musen}@stanford.edu

Abstract. WebProtégé is a highly customizable Web interface for browsing and editing ontologies, which provides support for collaboration. We have created a customized version to support the World Health Organization with the collaborative development of the 11th revision of the International Classification of Diseases (ICD-11). Our demo will present this customized version and focus on how content creation and collaboration is being supported in WebProtégé for the development of ICD-11.

1 Introduction

The International Classification of Diseases (ICD) is a public global standard that organizes and classifies information about diseases and related health problems [4]. Health officials use ICD in all United Nations member countries to compile basic health statistics, to monitor health-related spending, and to inform policy makers. In the United States, use of the ICD is also a requirement for all medical billing. ICD has therefore a major impact on many aspects of health care all over the world.

In 2007, the WHO initiated the 11th revision of ICD. Several ambitious goals were set for this version (details in [2]). One such goal is to allow the ICD to become a multi-purpose classification for a much larger number of usages. Previous versions of ICD were strictly classification hierarchies used for statistical purposes. To meet the new revision goals, ICD-11 will use OWL to create a rich formal representation. Another key difference between ICD-11 and previous versions is that the development process of ICD-11 will use a Web-based open process powered by collaboration and social features. That is, similar to Wikipedia, the WHO hopes that a large number of medical experts will contribute to the content of ICD-11.

Our group has been working closely with the WHO to provide the technical support for these ambitious goals. We have created a customized version of WebProtégé specifically designed to support the ICD authoring process. In [2], we discuss in detail the use of Semantic Web technologies for the revision of ICD. Our demo will showcase features of the customized WebProtégé such as content creation and collaboration. For the remainder of this paper, we present the architecture and highlight features of the user interface.

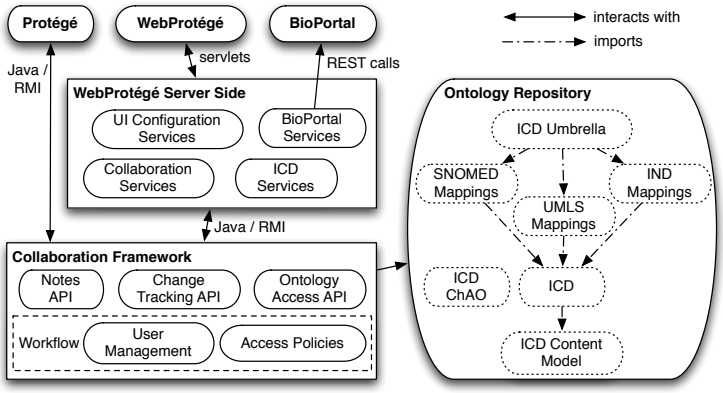


Fig. 1. An architecture diagram of WebProtégé used in the ICD context. WebProtégé front-end uses the services in the WebProtégé server side to display information to the user. The WebProtégé server side and the desktop client Protégé connect to the *Collaboration Framework* to access the ontology and the collaboration services. The *Ontology Repository* stores the ontologies available to the clients.

2 WebProtégé— Authoring Platform for ICD-11

WebProtégé is a pluggable and extensible platform for authoring ontologies. The tool was designed such that it could be customized for each project according to its own requirements. This is particularly important for the ICD revision project, as it is still in its infancy, many fundamental issues and requirements are undefined. Our tools need to be adaptable on the fly when changes are made to the content model, user-interface requirements, and workflow. In the following sections, we briefly describe the architecture and highlight some of the features of the user interface.

2.1 Architecture

Figure 1 shows a high level architecture diagram and the interaction of the software components for the customized version of WebProtégé. The core functionality of the application is supported by the Protégé server, which provides access to the ontology content, such as retrieving and changing classes, properties and individuals in the ontology. The ontologies that the server accesses are stored in a database on the server side. The Collaboration Framework [3] provides the collaboration and ontology access services: *Ontology Access API*, *Notes and Discussions API*, and *Change Tracking API*.

Both WebProtégé and the "traditional" Protégé desktop client can connect through the Collaboration Framework to interact with the ontology. Support for importing terms from existing terminologies was one of the requirements for the ICD-11 revision. In order to search external biomedical terminologies and to import terms from these terminologies, WebProtégé accesses BioPortal, a repository of about 200 biomedical ontologies and terminologies [1]. BioPortal provides REST service access that enables search across different ontologies and access to information about specific terms.

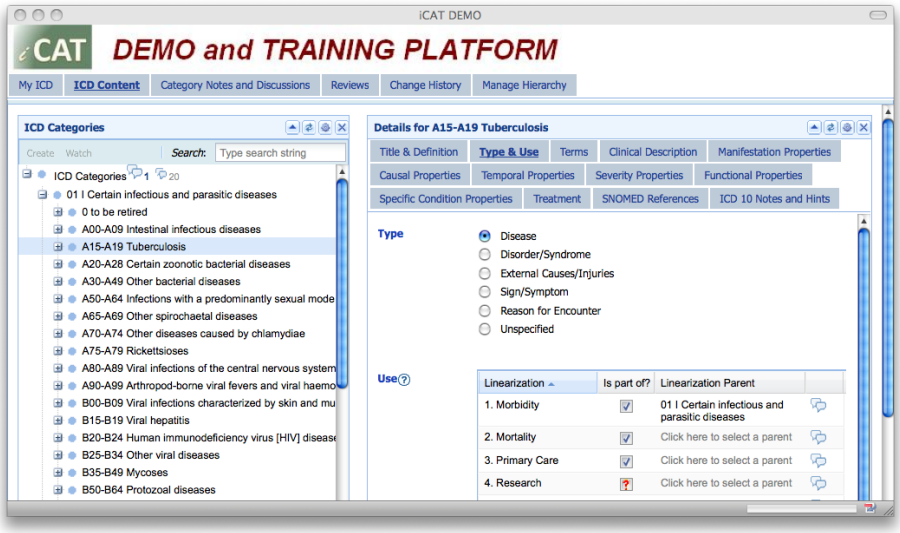


Fig. 2. The ICD authoring tool using WebProtégé. Each tab contains one or more panels, called *portlets* that can be arranged by drag-n-drop. The left hand-side portlet shows the disease class hierarchy of the ICD ontology. The right panel shows the uses (linearizations) of the selected disease in the tree, in this case *Tuberculosis*.

2.2 The WebProtégé User Interface

WebProtégé is a web portal, inspired by other portals, such as myYahoo or iGoogle. Our vision is to enable users to build a custom user interface by combining existing components in a form that is appropriate for their project. The user interface is composed of tabs, either predefined ones or user-defined. A new tab is an empty container in which users can add and arrange by drag-n-dropping portlets. A portlet is a user interface component that provides some functionality. For example, the Class tree portlet displays the class hierarchy in an ontology and has support for class level operations (create and delete class, move class in hierarchy, etc.). WebProtégé is extensible and has a plugin infrastructure.

For the ICD-11 revision project, we had to take into consideration that most of the ICD editors are domain experts, mainly medical doctors, who do not have backgrounds in ontology engineering. We configured the customized version of WebProtégé to use simple Web-based entry forms with fields that correspond to the attributes of the ICD content model. We have tried to ensure that the user interface does not look like an ontology editing environment, but is a customized experience for the domain experts.

The customized WebProtégé user interface is shown in Figure 2. The user interface is organized in a series of tabs: MyICD, ICD Content, Category Notes and Discussion, Reviews, Change History and Manage Hierarchy tab. Each tab presents a certain piece of functionality to the user.

The main functionality of WebProtégé is the support for browsing and editing ontologies on the Web. Editing support is available in the ICD Content tab.

Medical experts from all over the world are using the system to edit ICD-11 simultaneously. Each change by an editor is immediately committed and visible to other editors of ICD-11.

Another key piece of functionality is collaborative support. This is critical in such a large distributed project. Editors can use WebProtégé to add notes to classes, properties, and individuals in the ontology. This allows authors to raise questions and discuss different issues that arise during editing. The WHO plans to use a peer review process to ensure quality of the ICD content model. In the current version, WebProtégé supports a prototypical implementation of this feature. A user with appropriate permissions can request the review of a disease description.

WebProtégé uses a declarative user interface where the user interface components and layout is specified in an XML configuration file. The configuration can be changed on the fly, allowing the interface to be updated without recompiling or re-deploying the application. This feature provides great flexibility for customizing WebProtégé.

3 Summary

We have briefly presented a customization of WebProtégé, a web-based tool for distributed collaborative development of ontologies. The WHO is using WebProtégé as the primary development environment for ICD-11. Developers of other terminologies within the WHO Family of International Classifications (WHO-FIC) are beginning to use WebProtégé as well. In the demo session, we will demonstrate the main features and benefits of WebProtégé, specifically how to create and edit the ICD content model, how to create notes and discussion threads, and import terms from Bioportal.

References

1. N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute, and M. A. Musen. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic Acids Research*, 10.1093/nar/gkp440, 2009.
2. T. Tudorache, S. Falconer, C. Nyulas, and M. A. Musen. Will semantic web technologies work for the development of icd-11? In *International Semantic Web Conference (ISWC2010)*, 2010.
3. T. Tudorache, N. F. Noy, and M. A. Musen. Supporting collaborative ontology development in Protégé. In *Seventh International Semantic Web Conference, ISWC 2008*, Karlsruhe, Germany, 2008.
4. WHO. International classification of diseases, manual of the international statistical classification of diseases, injuries and causes of death: 10th revision. Technical report, 1993.

Building Linked Data Applications with Fusion: A Visual Interface for Exploration and Mapping

Samur Araujo[1], Geert-Jan Houben[1], Daniel Schwabe[2], Jan Hidders[1]

[1]Delft University of Technology, PO Box 5031, 2600 GA Delft, the Netherlands
[2]PUC-Rio, Rua Marques de Sao Vicente, 225, Rio de Janeiro, Brazil

{s.f.cardosodearaujo, g.j.p.m.houben, a.j.h.hidders}@tudelft.nl
dschwabe@inf.puc-rio.br

Abstract. Building applications over Linked Data often requires a mapping between the application model and the ontology underlying the source dataset in the Linked Data cloud. Explicitly formulating these mappings demands a comprehensive understanding of the underlying schemas (RDF ontologies) of the source and target datasets. This task can be supported by integrating the process of schema exploration into the mapping process and help the application designer with finding the implicit relationships that she wants to map. This demo describes Fusion - a framework for closing the gap between the application model and the underlying ontologies in the Linked Data cloud. Fusion simplifies the definition of mappings by providing a visual user interface that integrates the exploratory process and the mapping process. Its architecture allows the creation of new applications through the extension of existing Linked Data sources with additional data.

Keywords: semantic web, data interaction, data management, RDF mapping, Linked Data

1 Introduction

Nowadays, the Linked Data[1] cloud provides a new environment for building applications where many datasets are available for consumption. Although data in this cloud is ready to use, applications over the Linked Data cloud have currently an intrinsic characteristic: they consume RDF[2] data "as is", since designers do not have write permission over the data in the cloud which would enable them to change the data in any way. This fact raises an important issue concerning the development of applications over Linked Data: how to fill the gap between the ontology associated with the application model and the ontology used to represent the underlying data from the Linked Data cloud? The main benefit of mapping these two models is that

[1] Linked Data - http://linkeddata.org/
[2] http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Linked Data can then be accessed through properties defined in the application model, which is more convenient for the designer, consequently simplifying considerably the development and maintenance of the application.

This demo presents Fusion [1], a lightweight framework to support application designers in building applications over Linked Data. It supports designers in mapping the ontology of the used Linked Data sources to their application model by integrating the process of exploration of the target schema with the task of expressing a mapping rule itself. Fusion features a visual user interface that guides the designer in the process of specifying a mapping rule. It uses a standard RDF query language and allows Linked Data to be accessed using properties defined in the application model, consequently simplifying the use of Linked Data in a specific context.

2 Architecture Overview

The main aim of Fusion is to help the designer in discovering relationships in RDF graphs that exist in the Linked Data cloud and specifying rules for the derivation of new properties for these relationships. Fusion's architecture provides a complete environment to specify and execute a derivation rule. An overview of Fusion's architecture is shown in Fig. 1. The Fusion server engine is responsible for executing the derivation rule itself. During the process of executing a rule, it queries a source endpoint in the Linked Data, processes the results, and produces a set of new triples that will be added to the Fusion repository. Any RDF data store can be used as a Fusion repository. Currently, Fusion implements adapters for Sesame[6] and Virtuoso[7] data stores, although other adapters can be easily added to its architecture. All derived triples in Fusion contain as subject a resource whose URI belongs to the queried dataset, so the derived data is intrinsically interlinked with the Linked Data cloud. For this reason, a query over a federation of endpoints that includes the Fusion repository endpoint will allow the designer to have a view over the Linked Data that also includes the properties defined in her application model.

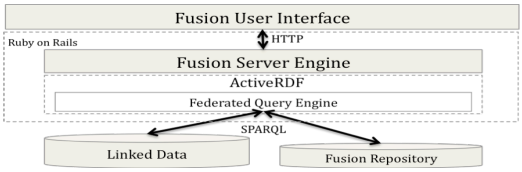


Fig. 1 – Fusion's architecture overview.

Fusion is implemented in Ruby on Rails[8] as a web application. It uses the ActiveRDF[9] API that allows an RDF graph to be accessed in the *object-oriented*

[6] http://www.openrdf.org/
[7] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/
[8] http://rubyonrails.org/
[9] http://www.activerdf.org/

paradigm. By using this API the properties of an RDF *resource* can be accessed as an attribute of its corresponding Ruby[10] object. This architecture allows the designer to write complex functions for computing a new *datatype* property value using the full power of the Ruby language, which cannot be achieved simply by using the SPARQL language.

Derivation rules could be executed on demand, by any rule engine associated to the databases in the federation. However, even if theoretically possible, executing inference rules, or even instantiating a *virtual view*, over the Linked Data is still an open problem, since it raises many performance issues. Indeed, querying data that is already materialized is always faster than querying data that needs to be processed at runtime. Fusion avoids this problem by materializing the result of the rules as new triples in the Fusion repository when the derivation rules are defined.

3 Example of Use

This section presents a scenario that illustrates the use of Fusion to create an application by extending Linked Data sources with additional properties. Suppose that the designer wants to establish the relationship between US senators and the US state that they represent. Therefore she needs to construct a derivation rule that will find and define such a correspondence between politicians and states in the GovTrack.Us's Linked Data. In the first step in the process, the designer provides an example of two resources in GovTrack.Us that she knows in advance that are actually related, for instance, the politician Christopher Bond and the state of Missouri. Also, she needs to declare the GovTrack.Us endpoint to be queried and the maximum depth of the path. As the result of this first step, Fusion shows all the paths that connect these two example resources satisfying the maximum path length. This result is shown in Fig. 2. In this example, the paths found have a maximum length of 3.

Fig. 2 – Fusion's interface showing the discovered paths.

[10] http://www.ruby-lang.org/en/

In this view, the designer can now look for the path that has the intended semantics. Note that with this view the tool assists the designer in this discovery process, since she does not need to query the schema manually in order to find these paths. The first path shown in Fig. 2 indicates that the politician Christopher Bond has a role as senator representing the state Missouri, and in our example case the designer can now infer that this is an instance of the path that she is looking for. After this conclusion, the designer chooses that instance to be the template for the rule.

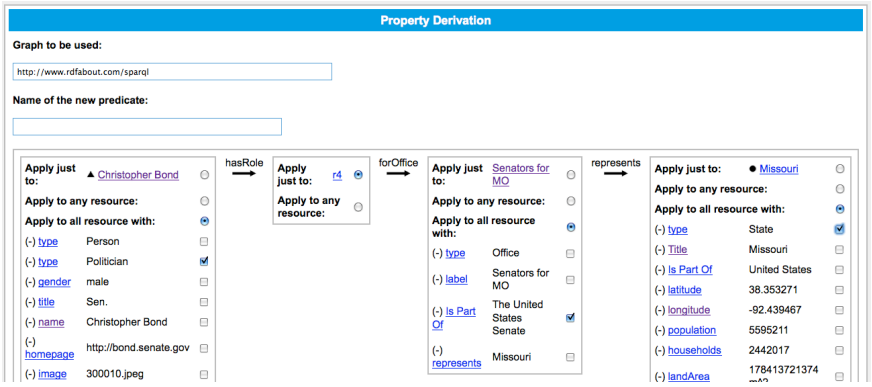


Fig. 3 – Generalizing the path for the property isSenatorOf in GovTrack.Us.

In the next step, shown in Fig. 3, the designer will define the derivation rule itself, which means that she visually formulates a query, which generalizes the selected path from the first step into a query that selects the elements to be connected through the property *isSenatorOf*. To complete this operation she also needs to define the graph where the derived triples will be stored and a specific URI to be used as the predicate of the new triples, which in this example will be the URI *http://example.org/isSenatorOf*. Note that in this example 3 nodes were generalized such that only paths between resources of the RDF type *Politician* and RDF type *State* that contain an intermediate node that *is part of* the United States Senate will be considered during the derivation process. Consequently, Fusion will derive the new property *isSenatorOf* for all instances of the class Politician that are connected to an instance of the class State through the designated path. The whole process ends with Fusion adding new triples to Fusion repository.

References

1. Araujo S., Houben G., Schwabe D., Hidders J. Fusion – Visually Exploring and Eliciting Relationships in Linked Data. In Proceedings of the 9th International Semantic Web Conference (ISWC2010). Shanghai, China. Nov 07-11, 2010.

STEREO: a SaT-based tool for an optimal solution of the sERvice selEctiOn problem

Daniel Izquierdo, María-Esther Vidal, and Blai Bonet

Departamento de Computación
Universidad Simón Bolívar
Caracas 89000, Venezuela
{idaniel,mvidal,bonet}@ldc.usb.ve

Abstract. We present STEREO, a system that offers an expressive formalism and implements techniques firmly grounded on logic to solve the Service Selection Problem (SSP). STEREO adopts the Local-As-View approach (LAV) to represent services' functionality as views on ontology concepts, while user requests are expressed as conjunctive queries on these concepts. Additionally, users can describe their preferences, which are used to rank the solutions. We discuss the LAV formulation of SSP; then, we illustrate the encoding of SSP as a logical theory whose models are in correspondence with the problem solutions, and in presence of preferences, the best models are in correspondence with the best-ranked solutions. We demonstrate STEREO and the properties of modern SAT solvers that provide an efficient and scalable solution to SSP.

1 Introduction

Nowadays, several annotation tools like the one proposed by Ambite et al. [1], are able to label and convert existing Web data sources into Semantic Web Services. Once this tremendous amount of services becomes available, users will more than ever require approaches to precisely express their selection needs and to effectively identify the services that best meet their requirements. To achieve this goal, we developed STEREO, a system that offers an expressive formalism for describing Web services and user requests, and implements a logic-based solution to efficiently select the services that best meet the user requirements. The STEREO formalism and techniques have been reported in [4].

STEREO is tailored to constantly changing Web service datasets and a relatively stable set of ontology concepts. STEREO adopts the recent approach of Ambite et al. [1] that describes services as views on ontology concepts following the LAV approach that is widely used in integration systems [5]. Thus, every time a service changes or a new one becomes available, only a tiny fraction of the mappings must be updated. Additionally, STEREO models user requests as conjunctive queries on the ontology concepts and casts SSP as the problem of rewriting a query in terms of a set of views, the so-called Query Rewriting Problem (QRP). Thus, STEREO exploits existing scalable and efficient techniques that have been proposed in the data integration area [2, 5]. Furthermore,

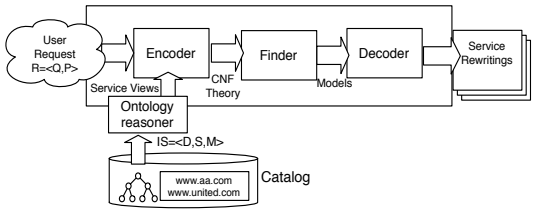


Fig. 1. The STEREO Architecture

STEREO offers a simple yet expressive language for preferences and supports users' preferences and constraints on the set of the selected services to satisfy a given request. These preferences and constraints further refine and rank the set of valid rewritings of the posed query, in a way that the best solutions to SSP corresponds to the best-ranked valid rewritings of the corresponding QRP. STEREO constructs a propositional logical theory that captures the features associated with SSPs; each model of the theory encodes a valid combination of services, and these models are ordered by their rank and the best-ranked models are the models with minimum rank. Knowledge encoded in the ontology is used to extend the rules in the logical theory to permit the cover of ontology symbols in the query with symbols in the views according to subsumption relationships represented in the ontology. We demonstrate the benefits of the approach and show the following key issues: the expressiveness by modeling a real-world domain; the scalability by demonstrating how STEREO is able to enumerate realistic instances of SSP in a few seconds; and finally, the performance by enumerating in the same logical theory, the optimal models based on different cost/rewards of the users' preferences. The demo is published at `http://stereo.ldc.usb.ve/STEREO`.

2 STEREO Architecture

The STEREO architecture is comprised of a Catalog of service descriptions, an Ontology Reasoner, the Encoder, the best model Finder, and the Decoder. Figure 1 depicts the overall architecture. An instance of SSP consists of an integration framework IS and a user request R. Formally, IS is a tuple $\langle D, S, M\rangle$ where D is an ontology[1], S is the set of services, and M is the set of LAV mappings. R is a tuple $\langle Q, P\rangle$ where Q is a conjunctive query on the ontology relational symbols and a set of preferences P. The STEREO Catalog is populated with the integration system components. We developed an Ontology Reasoner to compute the transitive closure of the subsumption relationships, and the Encoder module makes use of this information in conjunction with the LAV mappings, constant symbols, and users' preferences to encode an input instance of SSP

[1] An ontology is comprised of a set of relational and constant symbols from which logical formulas can be constructed, and a collection of axioms describing the ontology.

as a CNF theory. STEREO uses c2d (`http://reasoning.cs.ucla.edu/c2d`) to compute all the best-ranked models of the CNF theory in a certain normal form called deterministic and decomposable negation normal form (d-DNNF) [3]; however, off-the-shelf SAT solvers can be also used. The Finder computes a best model by enumerating the models and it can reuse the same d-DNNF for different cost/rewards associated with the preferences to calculate the best service rewritings. The Decoder translates the model(s) into the input instance.

3 Demonstration of Use Cases

We illustrate our approach on a simple travel domain that contains information about flight and train trips between cities and information about which cities are in the US. The ontology is comprised of the predicates *trip*, *flight*, *train* and *uscity*. The first predicate relates cities (x, y) if there is a direct trip either by plane or train between them. The *flight* predicate relates (x, y, t) whenever there is a direct flight from x to y operated by airline t, and similarly for *train*, and *uscity* indicates if a given city is or not a US city. The ontology axioms capture two subsumption relations:

$$\mathit{flight}(x, y, t) \sqsubseteq \mathit{trip}(x, y).\ \mathit{train}(x, y, t) \sqsubseteq \mathit{trip}(x, y).$$

For the services, we assume the following type of data sources:

- *nat-flight*(x, y, t) relates two US cities that are connected by a direct flight,
- *one-way-flight*(x, y) relates two cities that are connected by a one-way flight,
- *nat-train*(x, y) relates US cities that are connected by a direct train,
- *to-pa*(x) tells if there is a direct flight from x to Paris operated by American,
- *from-la*(x) tells if there is a direct flight from LA to x operated by United.

Based on the ontology concepts the services are described using LAV views:

$$\begin{aligned}
\mathit{nat\text{-}flight}(x, y) &:\!- \mathit{flight}(x, y, t),\ \mathit{uscity}(x),\ \mathit{uscity}(y)\,. \\
\mathit{one\text{-}way\text{-}flight}(x, y) &:\!- \mathit{flight}(x, y, t)\,. \\
\mathit{nat\text{-}train}(x, y) &:\!- \mathit{train}(x, y, t),\ \mathit{uscity}(x),\ \mathit{uscity}(y)\,. \\
\mathit{to\text{-}pa}(x) &:\!- \mathit{flight}(x, \mathrm{Paris}, \mathrm{AA})\,. \\
\mathit{from\text{-}la}(x) &:\!- \mathit{flight}(\mathrm{LA}, x, \mathrm{UA})\,.
\end{aligned}$$

Consider a user that needs to select the services able to retrieve one-stop round trips from a US city x to any city y in the world, and the user prefers flying than traveling by train. This request is represented as follows:

$$Q(x, y) :\!- \mathit{uscity}(x),\ \mathit{trip}(x, u),\ \mathit{trip}(u, y),\ \mathit{trip}(y, v),\ \mathit{trip}(v, x)\,.$$

This preference is modeled by assigning a high reward to the symbol *flight*. Any rewriting of the ontology predicates in terms of the services defined by these predicates, implements the request. In addition, rewritings comprised of services

defined by the symbol *flight* will have higher scores than those with services defined by the symbol *train*. Thus, this rewriting is valid and highly ranked:

$$I(x, \text{Paris}) \ :- \ \textit{nat-flight}(x, u),\ \textit{to-pa}(u),\ \textit{one-way-flight}(\text{Paris}, v),\ \textit{nat-flight}(v, x)\,.$$

But, the following rewriting is not a valid solution because it maps the query variable y into two different constants Paris and LA that denote different cities:

$$I'(x, y) \ :- \ \textit{nat-flight}(x, u),\ \textit{to-pa}(u),\ \textit{from-la}(v),\ \textit{nat-flight}(v, x)\,.$$

Using this travel domain, we consider the following scenarios:

- We run STEREO in three different benchmarks. We start with a benchmark of queries with 2 to 5 sub-goals and sets of 10 to 100 services; we show that the sets of 100 airlines with 5-stop flights can be compiled in 328 secs, the best model can be computed in 0.29 secs, and the enumeration of all models in 0.47 secs. We add a second service for each airline and show that STEREO behavior remains similar. Finally, we try services with multiple sub-goals which are randomly generated and show that the compilation time does not grow monotonically with the number of views. The time to find the best model is 0.46 secs while the enumeration of all models is about 17 hours.
- We test the system with ontologies of different sizes and show reusability by assigning different cost/reward to the preferences; we also demonstrate that the same d-DNNF can be used to compute the best model, while the execution time remains almost the same.

4 Conclusions

We present a logical-based approach that relies on the LAV formalization to solve SSP; the approach is expressive, efficient and scalable. We illustrate the formalization of service functionalities in terms of simple conjunctive rules. In addition, we demonstrate a propositional logic-based formalization of SSP instances, and different scenarios where the properties STEREO can be observed by users. We show how the approach can be applied to real-sized problems, while the whole approach is only possible when the compilation of the CNF theory into d-DNNF succeeds.

References

1. J. L. Ambite, S. Darbha, A. Goel, C. A. Knoblock, K. Lerman, R. Parundekar, and T. A. Russ. Automatically constructing semantic web services from online sources. In *ISWC*, pages 17–32, 2009.
2. Y. Arvelo, B. Bonet, and M.-E. Vidal. Compilation of query-rewriting problems into tractable fragments of propositional logic. In *AAAI*, 2006.
3. A. Darwiche. Decomposable negation normal form. *J. ACM*, 48(4):608–647, 2001.
4. D. Izquierdo, M.-E. Vidal, and B. Bonet. An expressive and efficient solution to the service selection problem. In *ISWC-To Appear*, 2010.
5. R. Pottinger and A. Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001.

The *Catalogus Professorum Lipsiensis* – Semantics-based Collaboration and Exploration for Historians

Thomas Riechert*, Ulf Morgenstern+, Sören Auer*, Sebastian Tramp*, and Michael Martin*

* AKSW, Institut für Informatik, Universität Leipzig, Pf 100920, 04009 Leipzig
{lastname}@informatik.uni-leipzig.de, http://aksw.org
+ Historisches Seminar, Universität Leipzig, Pf 100920, 04009 Leipzig
{lastname}@uni-leipzig.de, http://www.uni-leipzig.de/histsem

1 Introduction

The World Wide Web (WWW), as an ubiquitous medium for publication and exchange, already significantly influenced the way how historians work: the availability of public catalogs and bibliographies enable efficient research of relevant content for a certain investigation; the increasing digitization of works from historical archives and libraries, in addition, enables historians to directly access historical sources remotely. The capabilities of the WWW as a medium for collaboration, however, are only starting to be explored. Many historical questions are only answerable by combining information from different sources, from different researchers and organizations. Furthermore, after analyzing original sources, the derived information is often more comprehensive than can be captured by simple keyword indexing.

In [3] we report about the application of an adaptive, semantics-based knowledge engineering approach for the development of a prosopographical knowledge base. In this demonstration we will showcase the comprehensive prosopographical knowledge base and its potential for applications. In prosopographical research, historians analyze common characteristics of historical groups by studying statistically relevant quantities of individual biographies. Untraceable periods of biographies can be determined on the basis of such accomplished analyses in combination with statistically examinations as well as patterns of relationships between individuals and their activities. In our case, researchers from the Historical Seminar at the University of Leipzig aimed at creating a prosopographical knowledge base about the life and work of professors in the 600 years history of University of Leipzig ranging from the year 1409 till 2009 - the *Catalogus Professorum Lipsiensis* (CPL).

2 Architectural Overview

The system architecture of CPL comprises a combination of different applications, which interact using standardized interfaces as illustrated in Figure 1. We divided the architecture into two separated zones (public and protected zone) due to technical constraints and in order to prevent security problems.

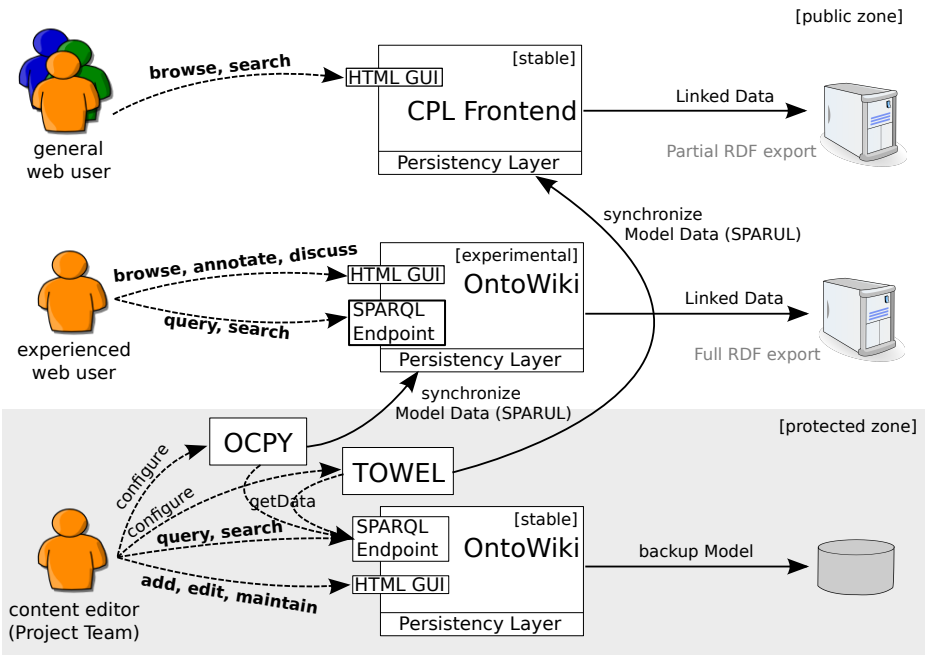


Fig. 1: Architectural overview about the project platforms

The semantic data wiki OntoWiki [1] located in the protected layer uses the *Catalogus Professorum Model*[1] (CPM), which comprises several ontologies and vocabularies for structuring the prosopographical information. The project team is working collaboratively and spatially distributed (e.g. in archives or libraries) to collect, structure and validate information about persons and institutions relevant to this knowledge domain.

For general web users the catalog is integrated in the public website of the University of Leipzig[2]. A simplified user interface consisting of plain HTML and Linked Data[3] resources is generated nightly from the knowledge base. The historians are able to interact with CPL via an experimental version[4] of OntoWiki. The version of the catalog available there is synchronized with the protected OntoWiki installation, transforms the exported data considering any linked knowledge bases and imports the changed data into this experimental installation. The project platform is usable for humans accessing web interfaces for reading or editing data. OntoWiki also provides a number of generic access interfaces, which include SPARQL and Linked Data endpoint and a Semantic Pingback server. On top of these generic access interfaces application specific access interfaces are deployed. These are described in more detail in the remainder and will be presented in the demonstration.

[1] Available at: `http://catalogus-professorum.org/cpm/`
[2] `http://www.uni-leipzig.de/unigeschichte/professorenkatalog/`
[3] `http://www.w3.org/DesignIssues/LinkedData.html`
[4] `http://catalogus-professorum.org/`

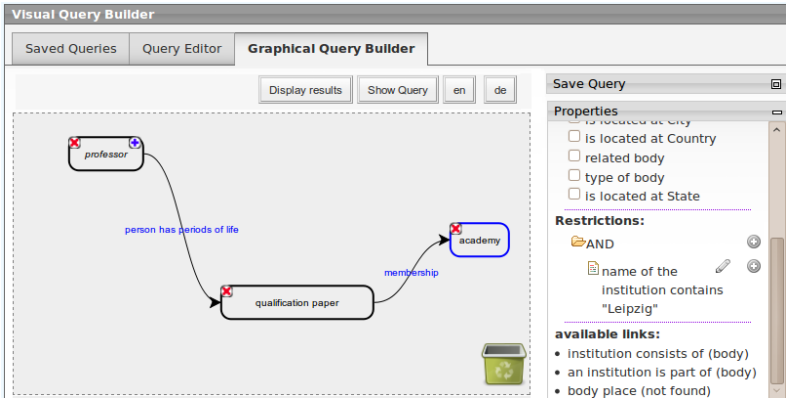


Fig. 2: *Visual Query Builder.*

3 Public CPL website and Linked Data

CPL is not just a tool for historians, but aims to showcase the results of historic research to the wider general public. For that purpose a special public website was created. The user interface of the public website is geared towards simplicity. The knowledge base can be explored by epochs, faculties, functions of professors (i.e. rector or dean) or alphabetically. Professors of the day are automatically selected based on important days in the life of a professor (i.e. birth or death). Furthermore, the public website comprises a full-text search, which searches within all literals stored in the CPL knowledge base.

Using OntoWiki's build-in endpoint functionality CPL is immediately available as Linked Data. Within the Linking Open Data effort, hundreds of data sets have already been connected to each other via `owl:sameAs` links. By interlinking CPL with other related datasets, we aim at establishing CPL as a linked data crystallization point for academic prosopographical knowledge.

4 Visual Query Builder.

OntoWiki also serves as a SPARQL endpoint, however, it quickly turned out that formulating SPARQL queries is too tedious for the historian domain experts. In order to simplify the creation of queries for the historians, we developed the *Visual Query Builder*[5] (VQB) as an OntoWiki extension, which is implemented in JavaScript and communicates with the triple store using the SPARQL language and protocol. VQB allows to visually create queries to the stored knowledge base and supports domain experts with an intuitive visual representation of query and data. Developed queries can be stored and added via drag-and-drop to the current query. This enables the reuse of existing queries as building blocks for more complex ones. VQB also supports the *set-based browsing* paradigm by visualizing different connectives, such as join, union,

[5] `http://aksw.org/Projects/OntoWiki/Extension/VQB`

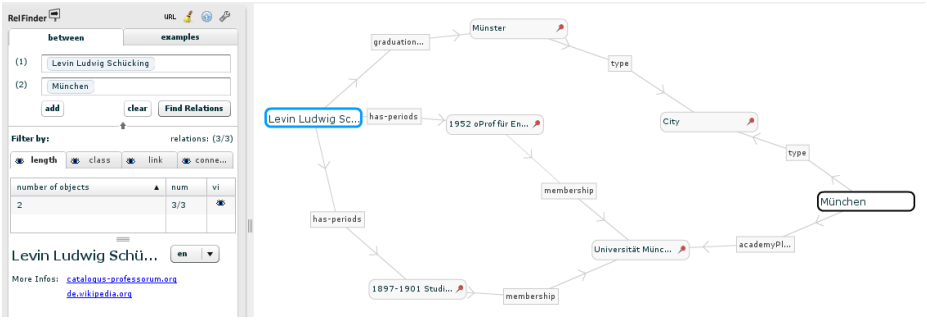


Fig. 3: Visualization of relationships in *RelFinder*.

intersection, difference between queries. The incremental query building is facilitated by displaying results already during query creation. The VQB user interface is visualized in Figure 2. The user interface can be adjusted by scaling or deactivating unused panels.

5 Relationship Finder.

An important aspect of historical investigations is the search for relationships between different persons or entities of interest. An application supporting such investigations within RDF knowledge knowledge bases is *RelFinder*[6] [2]. With the help of *RelFinder* relationships between individual entities can be easily discovered and visualized. Figure 3, for example, visualizes the relationship between the entities *Schücking* and *München*[7]. In this example, three connections were found and visualized as paths through the knowledge base. *RelFinder* is a generic tool and can be used in conjunction with arbitrary SPARQL endpoints.

References

1. Sören Auer, Sebastian Dietzold, and Thomas Riechert. OntoWiki – A Tool for Social, Semantic Collaboration. In *Proceedings of the 5th International Semantic Web Conference, ISWC2006*, Athens, GA, USA, 2006.
2. Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. RelFinder: revealing relationships in RDF knowledge bases. In *Proceedings of the 4th International Conference on Semantic and Digital Media Technologies*, 5887 of LNCS, pages 182–187. Springer, 2009.
3. Thomas Riechert, Ulf Morgenstern, Sören Auer, Sebastian Tramp, and Michael Martin. Knowledge engineering for historians on the example of the catalogus professorum lipsiensis. In *Proceedings of the 9th International Semantic Web Conference, ISWC2010*, Shanghai, China, 2010.

[6] Online at: `http://relfinder.semanticweb.org`

[7] More interesting relationships obtained from RelFinder are listed at: `http://catalogus-professorum.org/tools/relfinder`